# K21 Sub-Family Reference Manual

Supports: MK21FX512VLQ12, MK21FN1M0VLQ12,
MK21FX512VMD12, MK21FN1M0VMD12

Document Number: K21P144M120SF5RM
Rev. 4, November 2014

freescale™

# Contents

**Section number** **Title** **Page**

## Chapter 1
## About This Document



## Chapter 2
## Introduction



## Chapter 3
## Chip Configuration

## Chapter 4
## Memory Map

## Chapter 5
## Clock Distribution

## Chapter 6
## Reset and Boot



## Chapter 7
## Power Management



## Chapter 8
## Security

## Chapter 9
## Debug

## Chapter 10
## Signal Multiplexing and Signal Descriptions



## Chapter 11
## Port control and interrupts (PORT)

## Chapter 12
## System Integration Module (SIM)

## Chapter 15
## Power Management Controller (PMC)



## Chapter 16
## Low-Leakage Wakeup Unit (LLWU)

## Chapter 17
## Miscellaneous Control Module (MCM)

## Chapter 18
## Crossbar Switch (AXBS)



## Chapter 19
## Memory Protection Unit (MPU)

## Chapter 20
## Peripheral Bridge (AIPS-Lite)



## Chapter 21
## Direct Memory Access Multiplexer (DMAMUX)

## Chapter 22
## Direct Memory Access Controller (eDMA)

## Chapter 23
## External Watchdog Monitor (EWM)



## Chapter 24
## Watchdog Timer (WDOG)

## Chapter 25
## Multipurpose Clock Generator (MCG)

## Chapter 26
## Oscillator (OSC)

## Chapter 27
## RTC Oscillator (OSC32K)

## Chapter 28
## Flash Memory Controller (FMC)

## Chapter 29
## Flash Memory Module (FTFE)

## Chapter 30
## EzPort



## Chapter 31
## External Bus Interface (FlexBus)

## Chapter 32
## Cyclic Redundancy Check (CRC)

## Chapter 33
## Memory-Mapped Cryptographic Acceleration Unit (MMCAU)



## Chapter 34
## Random Number Generator Accelerator (RNGA)

## Chapter 35
## Analog-to-Digital Converter (ADC)

## Chapter 36
## Comparator (CMP)

## Chapter 37
## 12-bit Digital-to-Analog Converter (DAC)



## Chapter 38
## Voltage Reference (VREFV1)

## Chapter 39
## Programmable Delay Block (PDB)

## Chapter 40
## FlexTimer Module (FTM)

## Chapter 41
## Periodic Interrupt Timer (PIT)

## Chapter 42
## Low-Power Timer (LPTMR)

## Chapter 43
## Carrier Modulator Transmitter (CMT)

## Chapter 44
## Real Time Clock (RTC)

## Chapter 45
## Universal Serial Bus OTG Controller (USBOTG)

## Chapter 46
## USB Device Charger Detection Module (USBDCD)

## Chapter 47
## USB Voltage Regulator



## Chapter 48
## CAN (FlexCAN)

## Chapter 49
## Serial Peripheral Interface (SPI)

## Chapter 50
## Inter-Integrated Circuit (I2C)

## Chapter 51
## Universal Asynchronous Receiver/Transmitter (UART)

## Chapter 52
## Secured digital host controller (SDHC)

## Chapter 53
## Integrated Interchip Sound (I2S) / Synchronous Audio Interface (SAI)

## Chapter 54
## General-Purpose Input/Output (GPIO)



## Chapter 55
## JTAG Controller (JTAGC)

# Chapter 1
# About This Document

## 1.1 Overview

### 1.1.1 Purpose

This document describes the features, architecture, and programming model of the Freescale K21 microcontroller.

### 1.1.2 Audience

This document is primarily for system architects and software application developers who are using or considering using the microcontroller in a system.

## 1.2 Conventions

### 1.2.1 Numbering systems

The following suffixes identify different numbering systems:

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

## 1.2.2 Typographic notation

The following typographic notation is used throughout this document:

| Example | Description |
|---|---|
| *placeholder*, x | Items in italics are placeholders for information that you provide. Italicized text is also used for the titles of publications and for emphasis. Plain lowercase letters are also used as placeholders for single letters and numbers. |
| `code` | Fixed-width type indicates text that must be typed exactly as shown. It is used for instruction mnemonics, directives, symbols, subcommands, parameters, and operators. Fixed-width type is also used for example code. Instruction mnemonics and directives in text and tables are shown in all caps; for example, BSR. |
| SR[SCM] | A mnemonic in brackets represents a named field in a register. This example refers to the Scaling Mode (SCM) field in the Status Register (SR). |
| REVNO[6:4], XAD[7:0] | Numbers in brackets and separated by a colon represent either:<br>• A subset of a register's named field<br>For example, REVNO[6:4] refers to bits 6–4 that are part of the COREREV field that occupies bits 6–0 of the REVNO register.<br>• A continuous range of individual signals of a bus<br>For example, XAD[7:0] refers to signals 7–0 of the XAD bus. |

## 1.2.3 Special terms

The following terms have special meanings:

| Term | Meaning |
|---|---|
| asserted | Refers to the state of a signal as follows:<br>• An active-high signal is asserted when high (1).<br>• An active-low signal is asserted when low (0). |
| deasserted | Refers to the state of a signal as follows:<br>• An active-high signal is deasserted when low (0).<br>• An active-low signal is deasserted when high (1).<br><br>In some cases, deasserted signals are described as *negated*. |
| reserved | Refers to a memory space, register, or field that is either reserved for future use or for which, when written to, the module or chip behavior is unpredictable. |
| w1c | Write 1 to clear: Refers to a register bitfield that must be written as 1 to be "cleared." |

# Chapter 2
# Introduction

## 2.1 Overview

This chapter provides high-level descriptions of the modules available on the devices covered by this document.

## 2.2 Module Functional Categories

The modules on this device are grouped into functional categories. The following sections describe the modules assigned to each category in more detail.

**Table 2-1. Module functional categories**

| Module category | Description |
|---|---|
| ARM® Cortex™-M4 core | • 32-bit MCU core from ARM's Cortex-M class adding DSP instructions and single-precision floating point unit, 1.25 DMIPS/MHz, based on ARMv7 architecture |
| System | • System integration module<br>• Power management and mode controllers<br>  • Multiple power modes available based on run, wait, stop, and power-down modes<br>• Low-leakage wakeup unit<br>• Miscellaneous control module<br>• Crossbar switch<br>• Memory protection unit<br>• Peripheral bridge<br>• Direct memory access (DMA) controller with multiplexer to increase available DMA requests<br>• External watchdog monitor<br>• Watchdog |

*Table continues on the next page...*

**Table 2-1. Module functional categories (continued)**

| Module category | Description |
|---|---|
| Memories | • Internal memories include:<br>  • Program flash memory<br>  • On devices with FlexMemory: FlexMemory<br>  • FlexNVM<br>  • FlexRAM<br>  • On devices with program flash only: Programming acceleration RAM<br>  • SRAM<br>• External memory or peripheral bus interface: FlexBus<br>• Serial programming interface: EzPort |
| Clocks | • Multiple clock generation options available from internally- and externally-generated clocks<br>• System oscillator to provide clock source for the MCU<br>• RTC oscillator to provide clock source for the RTC |
| Security | • Cyclic Redundancy Check module for error detection<br>• Hardware encryption, along with a random number generator<br>• Tamper detect and secure storage |
| Analog | • High speed analog-to-digital converter<br>• Comparator<br>• Digital-to-analog converter<br>• Internal voltage reference<br>• Bandgap voltage reference |
| Timers | • Programmable delay block<br>• FlexTimers<br>• Periodic interrupt timer<br>• Low power timer<br>• Carrier modulator transmitter<br>• Independent real time clock |
| Communications | • USB OTG controller with built-in FS/LS transceiver<br>• USB device charger detect<br>• USB voltage regulator<br>• CAN<br>• Serial peripheral interface<br>• Inter-integrated circuit ($I^2C$)<br>• UART<br>• Secured Digital host controller<br>• Integrated interchip sound ($I^2S$) |
| Human-Machine Interfaces (HMI) | • General purpose input/output controller |

## 2.2.1 ARM® Cortex™-M4 Core Modules

The following core modules are available on this device.

**Table 2-2. Core modules**

| Module | Description |
|---|---|
| ARM Cortex-M4 | The ARM Cortex-M4 is the newest member of the Cortex M Series of processors targeting microcontroller cores focused on very cost sensitive, deterministic, interrupt driven environments. The Cortex M4 processor is based on the ARMv7 Architecture and Thumb®-2 ISA and is upward compatible with the Cortex M3, Cortex M1, and Cortex M0 architectures. Cortex M4 improvements include an ARMv7 Thumb-2 DSP (ported from the ARMv7-A/R profile architectures) providing 32-bit instructions with SIMD (single instruction multiple data) DSP style multiply-accumulates and saturating arithmetic. |
| Floating point unit (FPU) | A single-precision floating point unit (FPU) that is compliant to the *IEEE Standard for Floating-Point Arithmetic* (IEEE 754). |
| NVIC | The ARMv7-M exception model and nested-vectored interrupt controller (NVIC) implement a relocatable vector table supporting many external interrupts, a single non-maskable interrupt (NMI), and priority levels.<br>The NVIC replaces shadow registers with equivalent system and simplified programmability. The NVIC contains the address of the function to execute for a particular handler. The address is fetched via the instruction port allowing parallel register stacking and look-up. The first sixteen entries are allocated to ARM internal sources with the others mapping to MCU-defined interrupts. |
| AWIC | The primary function of the Asynchronous Wake-up Interrupt Controller (AWIC) is to detect asynchronous wake-up events in stop modes and signal to clock control logic to resume system clocking. After clock restart, the NVIC observes the pending interrupt and performs the normal interrupt or event processing. |
| Debug interfaces | Most of this device's debug is based on the ARM CoreSight™ architecture. Four debug interfaces are supported:<br>• IEEE 1149.1 JTAG<br>• IEEE 1149.7 JTAG (cJTAG)<br>• Serial Wire Debug (SWD)<br>• ARM Real-Time Trace Interface |

## 2.2.2 System Modules

The following system modules are available on this device.

**Table 2-3. System modules**

| Module | Description |
|---|---|
| System integration module (SIM) | The SIM includes integration logic and several module configuration settings. |
| System mode controller | The SMC provides control and protection on entry and exit to each power mode, control for the Power management controller (PMC), and reset entry and exit for the complete MCU. |
| Power management controller (PMC) | The PMC provides the user with multiple power options. Ten different modes are supported that allow the user to optimize power consumption for the level of functionality needed. Includes power-on-reset (POR) and integrated low voltage detect (LVD) with reset (brownout) capability and selectable LVD trip points. |
| Low-leakage wakeup unit (LLWU) | The LLWU module allows the device to wake from low leakage power modes (LLS and VLLS) through various internal peripheral and external pin sources. |

*Table continues on the next page...*

**Table 2-3. System modules (continued)**

| Module | Description |
|---|---|
| Miscellaneous control module (MCM) | The MCM includes integration logic and embedded trace buffer details. |
| Crossbar switch (XBS) | The XBS connects bus masters and bus slaves, allowing all bus masters to access different bus slaves simultaneously and providing arbitration among the bus masters when they access the same slave. |
| Memory protection unit (MPU) | The MPU provides memory protection and task isolation. It concurrently monitors all bus master transactions for the slave connections. |
| Peripheral bridges | The peripheral bridge converts the crossbar switch interface to an interface to access a majority of peripherals on the device. |
| DMA multiplexer (DMAMUX) | The DMA multiplexer selects from many DMA requests down to a smaller number for the DMA controller. |
| Direct memory access (DMA) controller | The DMA controller provides programmable channels with transfer control descriptors for data movement via dual-address transfers for 8-, 16-, 32- and 128-bit data values. |
| External watchdog monitor (EWM) | The EWM is a redundant mechanism to the software watchdog module that monitors both internal and external system operation for fail conditions. |
| Software watchdog (WDOG) | The WDOG monitors internal system operation and forces a reset in case of failure. It can run from an independent 1 KHz low power oscillator with a programmable refresh window to detect deviations in program flow or system frequency. |

## 2.2.3 Memories and Memory Interfaces

The following memories and memory interfaces are available on this device.

**Table 2-4. Memories and memory interfaces**

| Module | Description |
|---|---|
| Flash memory | • Program flash memory — non-volatile flash memory that can execute program code<br>• FlexMemory — encompasses the following memory types:<br>  • For devices with FlexNVM: FlexNVM — Non-volatile flash memory that can execute program code, store data, or backup EEPROM data<br>  • For devices with FlexNVM: FlexRAM — RAM memory that can be used as traditional RAM or as high-endurance EEPROM storage, and also accelerates flash programming<br>  • For devices with only program flash memory: Programming acceleration RAM — RAM memory that accelerates flash programming |
| Flash memory controller | Manages the interface between the device and the on-chip flash memory. |
| SRAM | Internal system RAM. Partial SRAM kept powered in VLLS2 low leakage mode. |
| SRAM controller | Manages simultaneous accesses to system RAM by multiple master peripherals and core. |
| System register file | 32-byte register file that is accessible during all power modes and is powered by VDD. |
| VBAT register file | 32-byte register file that is accessible during all power modes and is powered by VBAT. |

*Table continues on the next page...*

**Table 2-4. Memories and memory interfaces (continued)**

| Module | Description |
|---|---|
| Serial programming interface (EzPort) | Same serial interface as, and subset of, the command set used by industry-standard SPI flash memories. Provides the ability to read, erase, and program flash memory and reset command to boot the system after flash programming. |
| FlexBus | External bus interface with multiple independent, user-programmable chip-select signals that can interface with external SRAM, PROM, EPROM, EEPROM, flash, and other peripherals via 8-, 16- and 32-bit port sizes. Configurations include multiplexed or non-multiplexed address and data buses using 8-bit, 16-bit, 32-bit, and 16-byte line-sized transfers. |

## 2.2.4 Clocks

The following clock modules are available on this device.

**Table 2-5. Clock modules**

| Module | Description |
|---|---|
| Multi-clock generator (MCG) | The MCG provides several clock sources for the MCU that include:<br>• Phase-locked loop (PLL) — Voltage-controlled oscillator (VCO)<br>• Frequency-locked loop (FLL) — Digitally-controlled oscillator (DCO)<br>• Internal reference clocks — Can be used as a clock source for other on-chip peripherals |
| System oscillator | The system oscillator, in conjunction with an external crystal or resonator, generates a reference clock for the MCU. |
| Real-time clock oscillator | The RTC oscillator has an independent power supply and supports a 32 kHz crystal oscillator to feed the RTC clock. Optionally, the RTC oscillator can replace the system oscillator as the main oscillator source. |

## 2.2.5 Security and Integrity modules

The following security and integrity modules are available on this device:

**Table 2-6. Security and integrity modules**

| Module | Description |
|---|---|
| Cryptographic acceleration unit (CAU) | Supports DES, 3DES, AES, MD5, SHA-1, and SHA-256 algorithms via simple C calls to optimized security functions provided by Freescale. |
| Random number generator (RNG) | Supports the key generation algorithm defined in the Digital Signature Standard. |
| Cyclic Redundancy Check (CRC) | Hardware CRC generator circuit using 16/32-bit shift register. Error detection for all single, double, odd, and most multi-bit errors, programmable initial seed value, and optional feature to transpose input data and CRC result via transpose register. |
| DryIce (tamper detection and secure storage) | The DryIce module includes a 32-byte secure memory that is asynchronously erased on any tamper detect. In addition, it can optionally force a System Reset and/or invalidate the Real Time Clock. |

## 2.2.6 Analog modules

The following analog modules are available on this device:

**Table 2-7. Analog modules**

| Module | Description |
|---|---|
| 16-bit analog-to-digital converters (ADC) | 16-bit successive-approximation ADC |
| Analog comparators | Compares two analog input voltages across the full range of the supply voltage. |
| 6-bit digital-to-analog converters (DAC) | 64-tap resistor ladder network which provides a selectable voltage reference for applications where voltage reference is needed. |
| 12-bit digital-to-analog converters (DAC) | Low-power general-purpose DAC, whose output can be placed on an external pin or set as one of the inputs to the analog comparator or ADC. |
| Voltage reference (VREF) | Supplies an accurate voltage output that is trimmable in 0.5 mV steps. The VREF can be used in medical applications, such as glucose meters, to provide a reference voltage to biosensors or as a reference to analog peripherals, such as the ADC, DAC, or CMP. |

## 2.2.7 Timer modules

The following timer modules are available on this device:

**Table 2-8. Timer modules**

| Module | Description |
|---|---|
| Programmable delay block (PDB) | • 16-bit resolution<br>• 3-bit prescaler<br>• Positive transition of trigger event signal initiates the counter<br>• Supports two triggered delay output signals, each with an independently-controlled delay from the trigger event<br>• Outputs can be OR'd together to schedule two conversions from one input trigger event and can schedule precise edge placement for a pulsed output. This feature is used to generate the control signal for the CMP windowing feature and output to a package pin if needed for applications, such as critical conductive mode power factor correction.<br>• Continuous-pulse output or single-shot mode supported, each output is independently enabled, with possible trigger events<br>• Supports bypass mode<br>• Supports DMA |

*Table continues on the next page...*

**Table 2-8. Timer modules (continued)**

| Module | Description |
|---|---|
| Flexible timer modules (FTM) | • Selectable FTM source clock, programmable prescaler<br>• 16-bit counter supporting free-running or initial/final value, and counting is up or up-down<br>• Input capture, output compare, and edge-aligned and center-aligned PWM modes<br>• Operation of FTM channels as pairs with equal outputs, pairs with complimentary outputs, or independent channels with independent outputs<br>• Deadtime insertion is available for each complementary pair<br>• Generation of hardware triggers<br>• Software control of PWM outputs<br>• Up to 4 fault inputs for global fault control<br>• Configurable channel polarity<br>• Programmable interrupt on input capture, reference compare, overflowed counter, or detected fault condition<br>• Quadrature decoder with input filters, relative position counting, and interrupt on position count or capture of position count on external event<br>• DMA support for FTM events |
| Periodic interrupt timers (PIT) | • Four general purpose interrupt timers<br>• Interrupt timers for triggering ADC conversions<br>• 32-bit counter resolution<br>• DMA support |
| Low-power timer (LPTimer) | • Selectable clock for prescaler/glitch filter of 1 kHz (internal LPO), 32.768 kHz (external crystal), or internal reference clock<br>• Configurable Glitch Filter or Prescaler with 16-bit counter<br>• 16-bit time or pulse counter with compare<br>• Interrupt generated on Timer Compare<br>• Hardware trigger generated on Timer Compare |
| Carrier modulator timer (CMT) | • Four CMT modes of operation:<br>  • Time with independent control of high and low times<br>  • Baseband<br>  • Frequency shift key (FSK)<br>  • Direct software control of CMT_IRO pin<br>• Extended space operation in time, baseband, and FSK modes<br>• Selectable input clock divider<br>• Interrupt on end of cycle with the ability to disable CMT_IRO pin and use as timer interrupt<br>• DMA support |
| Real-time clock (RTC) | • Independent power supply, POR, and 32 kHz Crystal Oscillator<br>• 32-bit seconds counter with 32-bit Alarm<br>• 16-bit Prescaler with compensation that can correct errors between 0.12 ppm and 3906 ppm |

## 2.2.8 Communication interfaces

The following communication interfaces are available on this device:

**Table 2-9. Communication modules**

| Module | Description |
|---|---|
| USB OTG (low-/full-speed) | USB 2.0 compliant module with support for host, device, and On-The-Go modes. Includes an on-chip transceiver for full and low speeds. |
| USB Device Charger Detect (USBDCD) | The USBDCD monitors the USB data lines to detect a smart charger meeting the USB Battery Charging Specification Rev1.2. This information allows the MCU to better manage the battery charging IC in a portable device. |
| USB voltage regulator | Up to 5 V regulator input typically provided by USB VBUS power with 3.3 V regulated output that powers on-chip USB subsystem, capable of sourcing 120 mA to external board components. |
| Controller Area Network (CAN) | Supports the full implementation of the CAN Specification Version 2.0, Part B |
| Serial peripheral interface (SPI) | Synchronous serial bus for communication to an external device |
| Inter-integrated circuit (I2C) | Allows communication between a number of devices. Also supports the System Management Bus (SMBus) Specification, version 2. |
| Universal asynchronous receiver/ transmitters (UART) | Asynchronous serial bus communication interface with programmable 8- or 9-bit data format and support of ISO 7816 smart card interface |
| Secure Digital host controller (SDHC) | Interface between the host system and the SD, SDIO, MMC, or CE-ATA cards. The SDHC acts as a bridge, passing host bus transactions to the cards by sending commands and performing data accesses to/from the cards. It handles the SD, SDIO, MMC, and CE-ATA protocols at the transmission level. |
| I2S | The $I^2S$ is a full-duplex, serial port that allows the chip to communicate with a variety of serial devices, such as standard codecs, digital signal processors (DSPs), microprocessors, peripherals, and audio codecs that implement the inter-IC sound bus ($I^2S$) and the Intel® AC97 standards |

### 2.2.9 Human-machine interfaces

The following human-machine interfaces (HMI) are available on this device:

**Table 2-10. HMI modules**

| Module | Description |
|---|---|
| General purpose input/output (GPIO) | All general purpose input or output (GPIO) pins are capable of interrupt and DMA request generation. All GPIO pins have 5 V tolerance. |

## 2.3 Orderable part numbers

The following table summarizes the part numbers of the devices covered by this document.

## Table 2-11. Orderable part numbers summary

| Freescale part number | CPU frequency | Pin count | Package | Total flash memory | Program flash | EEPROM | SRAM | GPIO |
|---|---|---|---|---|---|---|---|---|
| MK21FX512VLQ12 | 120 MHz | 144 | LQFP | 640 KB | 512 KB | 4 KB | 128 KB | 104 |
| MK21FN1M0VLQ12 | 120 MHz | 144 | LQFP | 1 MB | 1 MB | — | 128 KB | 104 |
| MK21FX512VMD12 | 120 MHz | 144 | MAPBGA | 640 KB | 512 KB | 4 KB | 128 KB | 104 |
| MK21FN1M0VMD12 | 120 MHz | 144 | MAPBGA | 1 MB | 1 MB | — | 128 KB | 104 |

# Chapter 3
# Chip Configuration

## 3.1 Introduction

This chapter provides details on the individual modules of the microcontroller. It includes:

- module block diagrams showing immediate connections within the device,
- specific module-to-module interactions not necessarily discussed in the individual module chapters, and
- links for more information.

## 3.2 Core modules

### 3.2.1 ARM Cortex-M4 Core Configuration

This section summarizes how the module has been configured in the chip. Full documentation for this module is provided by ARM and can be found at **arm.com**.

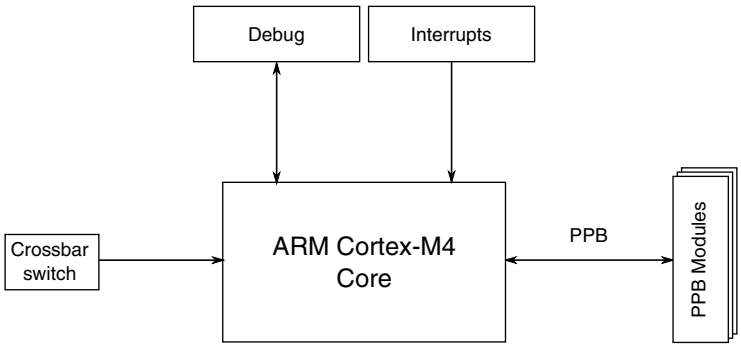



**Figure 3-1. Core configuration**

**Table 3-1. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | ARM Cortex-M4 core | ARM Cortex-M4 Technical Reference Manual |
| System memory map | | System memory map |
| Clocking | | Clock distribution |
| Power management | | Power management |
| System/instruction/data bus module | Crossbar switch | Crossbar switch |
| Debug | IEEE 1149.1 JTAG<br>IEEE 1149.7 JTAG (cJTAG)<br>Serial Wire Debug (SWD)<br>ARM Real-Time Trace Interface | Debug |
| Interrupts | Nested Vectored Interrupt Controller (NVIC) | NVIC |
| Private Peripheral Bus (PPB) module | Miscellaneous Control Module (MCM) | MCM |
| Private Peripheral Bus (PPB) module | Memory-Mapped Cryptographic Acceleration Unit (MMCAU) | MMCAU |

### 3.2.1.1 Buses, interconnects, and interfaces

The ARM Cortex-M4 core has four buses as described in the following table.

| Bus name | Description |
|---|---|
| Instruction code (ICODE) bus | The ICODE and DCODE buses are muxed. This muxed bus is called the CODE bus and is connected to the crossbar switch via a single master port. |
| Data code (DCODE) bus | |

*Table continues on the next page...*

| Bus name | Description |
|---|---|
| System bus | The system bus is connected to a separate master port on the crossbar. |
| Private peripheral (PPB) bus | The PPB provides access to these modules:<br>• ARM modules such as the NVIC, ETM, ITM, DWT, FBP, and ROM table<br>• Freescale Miscellaneous Control Module (MCM)<br>• Memory-Mapped Cryptographic Acceleration Unit (MMCAU) |

#### 3.2.1.2 System Tick Timer

The System Tick Timer's clock source is always the core clock, FCLK. This results in the following:

- The CLKSOURCE bit in SysTick Control and Status register is always set to select the core clock.
- Because the timing reference (FCLK) is a variable frequency, the TENMS bit in the SysTick Calibration Value Register is always zero.
- The NOREF bit in SysTick Calibration Value Register is always set, implying that FCLK is the only available source of reference timing.

#### 3.2.1.3 Debug facilities

This device has extensive debug capabilities including run control and tracing capabilities. The standard ARM debug port that supports JTAG and SWD interfaces. Also the cJTAG interface is supported on this device.

#### 3.2.1.4 Core privilege levels

The ARM documentation uses different terms than this document to distinguish between privilege levels.

| If you see this term... | it also means this term... |
|---|---|
| Privileged | Supervisor |
| Unprivileged or user | User |

### 3.2.2 Nested Vectored Interrupt Controller (NVIC) Configuration

This section summarizes how the module has been configured in the chip. Full documentation for this module is provided by ARM and can be found at **arm.com**.

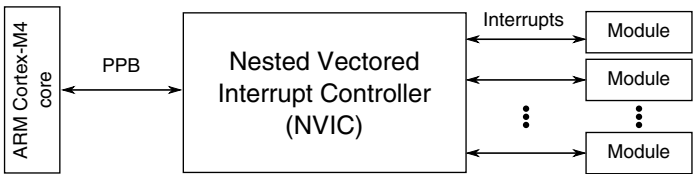



**Figure 3-2. NVIC configuration**

**Table 3-2. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | Nested Vectored Interrupt Controller (NVIC) | ARM Cortex-M4 Technical Reference Manual |
| System memory map | | System memory map |
| Clocking | | Clock distribution |
| Power management | | Power management |
| Private Peripheral Bus (PPB) | ARM Cortex-M4 core | ARM Cortex-M4 core |

### 3.2.2.1 Interrupt priority levels

This device supports 16 priority levels for interrupts. Therefore, in the NVIC each source in the IPR registers contains 4 bits. For example, IPR0 is shown below:

| | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | IRQ3 | | | | 0 | 0 | 0 | 0 | IRQ2 | | | | 0 | 0 | 0 | 0 | IRQ1 | | | | 0 | 0 | 0 | 0 | IRQ0 | | | | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

### 3.2.2.2 Non-maskable interrupt

The non-maskable interrupt request to the NVIC is controlled by the external $\overline{\text{NMI}}$ signal. The pin the $\overline{\text{NMI}}$ signal is multiplexed on, must be configured for the $\overline{\text{NMI}}$ function to generate the non-maskable interrupt request.

### 3.2.2.3 Interrupt channel assignments

The interrupt source assignments are defined in the following table.

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

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

**Table 3-4. Interrupt vector assignments**

| Address | Vector | IRQ[1] | NVIC non-IPR register number [2] | NVIC IPR register number [3] | Source module | Source description |
|---|---|---|---|---|---|---|
| **ARM Core System Handler Vectors** | | | | | | |
| 0x0000_0000 | 0 | – | – | – | ARM core | Initial Stack Pointer |
| 0x0000_0004 | 1 | – | – | – | ARM core | Initial Program Counter |
| 0x0000_0008 | 2 | – | – | – | ARM core | Non-maskable Interrupt (NMI) |
| 0x0000_000C | 3 | – | – | – | ARM core | Hard Fault |
| 0x0000_0010 | 4 | – | – | – | ARM core | MemManage Fault |
| 0x0000_0014 | 5 | – | – | – | ARM core | Bus Fault |
| 0x0000_0018 | 6 | – | – | – | ARM core | Usage Fault |
| 0x0000_001C | 7 | – | – | – | — | — |
| 0x0000_0020 | 8 | – | – | – | — | — |
| 0x0000_0024 | 9 | – | – | – | — | — |
| 0x0000_0028 | 10 | – | – | – | — | — |
| 0x0000_002C | 11 | – | – | – | ARM core | Supervisor call (SVCall) |
| 0x0000_0030 | 12 | – | – | – | ARM core | Debug Monitor |
| 0x0000_0034 | 13 | – | – | – | — | — |
| 0x0000_0038 | 14 | – | – | – | ARM core | Pendable request for system service (PendableSrvReq) |
| 0x0000_003C | 15 | – | – | – | ARM core | System tick timer (SysTick) |
| **Non-Core Vectors** | | | | | | |
| 0x0000_0040 | 16 | 0 | 0 | 0 | DMA | DMA channel 0 transfer complete |
| 0x0000_0044 | 17 | 1 | 0 | 0 | DMA | DMA channel 1 transfer complete |
| 0x0000_0048 | 18 | 2 | 0 | 0 | DMA | DMA channel 2 transfer complete |
| 0x0000_004C | 19 | 3 | 0 | 0 | DMA | DMA channel 3 transfer complete |
| 0x0000_0050 | 20 | 4 | 0 | 1 | DMA | DMA channel 4 transfer complete |
| 0x0000_0054 | 21 | 5 | 0 | 1 | DMA | DMA channel 5 transfer complete |
| 0x0000_0058 | 22 | 6 | 0 | 1 | DMA | DMA channel 6 transfer complete |
| 0x0000_005C | 23 | 7 | 0 | 1 | DMA | DMA channel 7 transfer complete |
| 0x0000_0060 | 24 | 8 | 0 | 2 | DMA | DMA channel 8 transfer complete |
| 0x0000_0064 | 25 | 9 | 0 | 2 | DMA | DMA channel 9 transfer complete |
| 0x0000_0068 | 26 | 10 | 0 | 2 | DMA | DMA channel 10 transfer complete |
| 0x0000_006C | 27 | 11 | 0 | 2 | DMA | DMA channel 11 transfer complete |

*Table continues on the next page...*

## Table 3-4. Interrupt vector assignments (continued)

| Address | Vector | IRQ[1] | NVIC non-IPR register number [2] | NVIC IPR register number [3] | Source module | Source description |
|---|---|---|---|---|---|---|
| 0x0000_0070 | 28 | 12 | 0 | 3 | DMA | DMA channel 12 transfer complete |
| 0x0000_0074 | 29 | 13 | 0 | 3 | DMA | DMA channel 13 transfer complete |
| 0x0000_0078 | 30 | 14 | 0 | 3 | DMA | DMA channel 14 transfer complete |
| 0x0000_007C | 31 | 15 | 0 | 3 | DMA | DMA channel 15 transfer complete |
| 0x0000_0080 | 32 | 16 | 0 | 4 | DMA | DMA error interrupt channels 0-15 |
| 0x0000_0084 | 33 | 17 | 0 | 4 | MCM | MCM interrupt |
| 0x0000_0088 | 34 | 18 | 0 | 4 | Flash memory | Command complete |
| 0x0000_008C | 35 | 19 | 0 | 4 | Flash memory | Read collision |
| 0x0000_0090 | 36 | 20 | 0 | 5 | Mode Controller | Low-voltage detect, low-voltage warning |
| 0x0000_0094 | 37 | 21 | 0 | 5 | LLWU | Low Leakage Wakeup<br>**NOTE:** The LLWU interrupt must not be masked by the interrupt controller to avoid a scenario where the system does not fully exit stop mode on an LLS recovery. |
| 0x0000_0098 | 38 | 22 | 0 | 5 | WDOG or EWM | Both watchdog modules share this interrupt |
| 0x0000_009C | 39 | 23 | 0 | 5 | RNG | Randon Number Generator |
| 0x0000_00A0 | 40 | 24 | 0 | 6 | $I^2C0$ | — |
| 0x0000_00A4 | 41 | 25 | 0 | 6 | $I^2C1$ | — |
| 0x0000_00A8 | 42 | 26 | 0 | 6 | SPI0 | Single interrupt vector for all sources |
| 0x0000_00AC | 43 | 27 | 0 | 6 | SPI1 | Single interrupt vector for all sources |
| 0x0000_00B0 | 44 | 28 | 0 | 7 | $I^2S0$ | Transmit |
| 0x0000_00B4 | 45 | 29 | 0 | 7 | $I^2S0$ | Receive |
| 0x0000_00B8 | 46 | 30 | 0 | 7 | — | — |
| 0x0000_00BC | 47 | 31 | 0 | 7 | UART0 | Single interrupt vector for UART status sources |
| 0x0000_00C0 | 48 | 32 | 1 | 8 | UART0 | Single interrupt vector for UART error sources |
| 0x0000_00C4 | 49 | 33 | 1 | 8 | UART1 | Single interrupt vector for UART status sources |
| 0x0000_00C8 | 50 | 34 | 1 | 8 | UART1 | Single interrupt vector for UART error sources |
| 0x0000_00CC | 51 | 35 | 1 | 8 | UART2 | Single interrupt vector for UART status sources |
| 0x0000_00D0 | 52 | 36 | 1 | 9 | UART2 | Single interrupt vector for UART error sources |
| 0x0000_00D4 | 53 | 37 | 1 | 9 | UART3 | Single interrupt vector for UART status sources |

*Table continues on the next page...*

## Table 3-4. Interrupt vector assignments (continued)

| Address | Vector | IRQ[1] | NVIC non-IPR register number [2] | NVIC IPR register number [3] | Source module | Source description |
|---|---|---|---|---|---|---|
| 0x0000_00D8 | 54 | 38 | 1 | 9 | UART3 | Single interrupt vector for UART error sources |
| 0x0000_00DC | 55 | 39 | 1 | 9 | ADC0 | — |
| 0x0000_00E0 | 56 | 40 | 1 | 10 | CMP0 | — |
| 0x0000_00E4 | 57 | 41 | 1 | 10 | CMP1 | — |
| 0x0000_00E8 | 58 | 42 | 1 | 10 | FTM0 | Single interrupt vector for all sources |
| 0x0000_00EC | 59 | 43 | 1 | 10 | FTM1 | Single interrupt vector for all sources |
| 0x0000_00F0 | 60 | 44 | 1 | 11 | FTM2 | Single interrupt vector for all sources |
| 0x0000_00F4 | 61 | 45 | 1 | 11 | CMT | — |
| 0x0000_00F8 | 62 | 46 | 1 | 11 | RTC | Alarm interrupt |
| 0x0000_00FC | 63 | 47 | 1 | 11 | RTC | Seconds interrupt |
| 0x0000_0100 | 64 | 48 | 1 | 12 | PIT | Channel 0 |
| 0x0000_0104 | 65 | 49 | 1 | 12 | PIT | Channel 1 |
| 0x0000_0108 | 66 | 50 | 1 | 12 | PIT | Channel 2 |
| 0x0000_010C | 67 | 51 | 1 | 12 | PIT | Channel 3 |
| 0x0000_0110 | 68 | 52 | 1 | 13 | PDB | — |
| 0x0000_0114 | 69 | 53 | 1 | 13 | USB OTG | — |
| 0x0000_0118 | 70 | 54 | 1 | 13 | USB Charger Detect | — |
| 0x0000_011C | 71 | 55 | 1 | 13 | — | — |
| 0x0000_0120 | 72 | 56 | 1 | 14 | DAC0 | — |
| 0x0000_0124 | 73 | 57 | 1 | 14 | MCG | — |
| 0x0000_0128 | 74 | 58 | 1 | 14 | Low Power Timer | — |
| 0x0000_012C | 75 | 59 | 1 | 14 | Port control module | Pin detect (Port A) |
| 0x0000_0130 | 76 | 60 | 1 | 15 | Port control module | Pin detect (Port B) |
| 0x0000_0134 | 77 | 61 | 1 | 15 | Port control module | Pin detect (Port C) |
| 0x0000_0138 | 78 | 62 | 1 | 15 | Port control module | Pin detect (Port D) |
| 0x0000_013C | 79 | 63 | 1 | 15 | Port control module | Pin detect (Port E) |
| 0x0000_0140 | 80 | 64 | 2 | 16 | Software | Software interrupt[4] |
| 0x0000_0144 | 81 | 65 | 2 | 16 | SPI2 | Single interrupt vector for all sources |
| 0x0000_0148 | 82 | 66 | 2 | 16 | UART4 | Single interrupt vector for UART status sources |
| 0x0000_014C | 83 | 67 | 2 | 16 | UART4 | Single interrupt vector for UART error sources |
| 0x0000_0150 | 84 | 68 | 2 | 17 | UART5 | Single interrupt vector for UART status sources |
| 0x0000_0154 | 85 | 69 | 2 | 17 | UART5 | Single interrupt vector for UART error sources |

*Table continues on the next page...*

**Table 3-4. Interrupt vector assignments (continued)**

| Address | Vector | IRQ[1] | NVIC non-IPR register number [2] | NVIC IPR register number [3] | Source module | Source description |
|---|---|---|---|---|---|---|
| 0x0000_0158 | 86 | 70 | 2 | 17 | CMP2 | — |
| 0x0000_015C | 87 | 71 | 2 | 17 | FTM3 | Single interrupt vector for all sources |
| 0x0000_0160 | 88 | 72 | 2 | 18 | DAC1 | — |
| 0x0000_0164 | 89 | 73 | 2 | 18 | ADC1 | — |
| 0x0000_0168 | 90 | 74 | 2 | 18 | $I^2C2$ | — |
| 0x0000_016C | 91 | 75 | 2 | 18 | CAN0 | OR'ed Message buffer (0-15) |
| 0x0000_0170 | 92 | 76 | 2 | 19 | CAN0 | Bus Off |
| 0x0000_0174 | 93 | 77 | 2 | 19 | CAN0 | Error |
| 0x0000_0178 | 94 | 78 | 2 | 19 | CAN0 | Transmit Warning |
| 0x0000_017C | 95 | 79 | 2 | 19 | CAN0 | Receive Warning |
| 0x0000_0180 | 96 | 80 | 2 | 20 | CAN0 | Wake Up |
| 0x0000_0184 | 97 | 81 | 2 | 20 | SDHC | — |

1. Indicates the NVIC's interrupt source number.
2. Indicates the NVIC's ISER, ICER, ISPR, ICPR, and IABR register number used for this IRQ. The equation to calculate this value is: IRQ div 32
3. Indicates the NVIC's IPR register number used for this IRQ. The equation to calculate this value is: IRQ div 4
4. This interrupt can only be pended or cleared via the NVIC registers.

#### 3.2.2.3.1 Determining the bitfield and register location for configuring a particular interrupt

Suppose you need to configure the low-power timer (LPTMR) interrupt. The following table is an excerpt of the LPTMR row from Interrupt channel assignments.

**Table 3-5. LPTMR interrupt vector assignment**

| Address | Vector | IRQ[1] | NVIC non-IPR register number [2] | NVIC IPR register number [3] | Source module | Source description |
|---|---|---|---|---|---|---|
| 0x0000_0138 | 74 | 58 | 1 | 14 | Low Power Timer | — |

1. Indicates the NVIC's interrupt source number.
2. Indicates the NVIC's ISER, ICER, ISPR, ICPR, and IABR register number used for this IRQ. The equation to calculate this value is: IRQ div 32
3. Indicates the NVIC's IPR register number used for this IRQ. The equation to calculate this value is: IRQ div 4

- The NVIC registers you would use to configure the interrupt are:
  - NVICISER1
  - NVICICER1

  - NVICISPR1
  - NVICICPR1
  - NVICIABR1
  - NVICIPR14
- To determine the particular IRQ's bitfield location within these particular registers:
  - NVICISER1, NVICICER1, NVICISPR1, NVICICPR1, NVICIABR1 bit location = IRQ mod 32 = 26
  - NVICIPR14 bitfield starting location = 8 * (IRQ mod 4) + 4 = 20

    Since the NVICIPR bitfields are 4-bit wide (16 priority levels), the NVICIPR14 bitfield range is 20-23

Therefore, the following bitfield locations are used to configure the LPTMR interrupts:

- NVICISER1[26]
- NVICICER1[26]
- NVICISPR1[26]
- NVICICPR1[26]
- NVICIABR1[26]
- NVICIPR14[23:20]

### 3.2.3 Asynchronous Wake-up Interrupt Controller (AWIC) Configuration

This section summarizes how the module has been configured in the chip. Full documentation for this module is provided by ARM and can be found at **arm.com**.

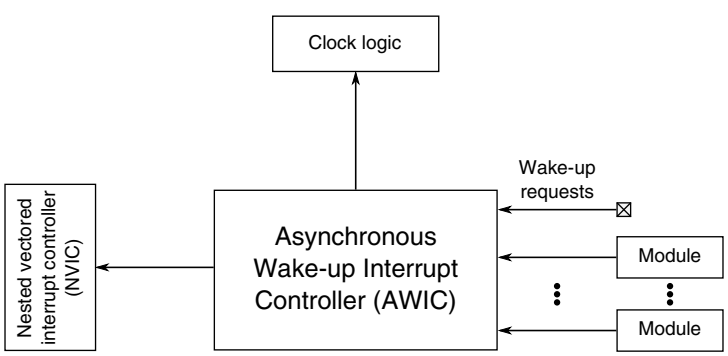



**Figure 3-3. Asynchronous Wake-up Interrupt Controller configuration**

**Table 3-6. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| System memory map | | System memory map |

*Table continues on the next page...*

**Table 3-6. Reference links to related information (continued)**

| Topic | Related module | Reference |
|---|---|---|
| Clocking | | Clock distribution |
| Power management | | Power management |
| | Nested Vectored Interrupt Controller (NVIC) | NVIC |
| Wake-up requests | | AWIC wake-up sources |

### 3.2.3.1 Wake-up sources

The device uses the following internal and external inputs to the AWIC module.

**Table 3-7. AWIC Stop and VLPS Wake-up Sources**

| Wake-up source | Description |
|---|---|
| Available system resets | RESET pin and WDOG when LPO is its clock source, and JTAG |
| Low-voltage detect | Mode Controller |
| Low-voltage warning | Mode Controller |
| Pin interrupts | Port Control Module - Any enabled pin interrupt is capable of waking the system |
| ADCx | The ADC is functional when using internal clock source |
| CMPx | Since no system clocks are available, functionality is limited |
| $I^2C$ | Address match wakeup |
| UART | Active edge on RXD |
| USB | Wakeup |
| LPTMR | Functional in Stop/VLPS modes |
| RTC | Functional in Stop/VLPS modes |
| SDHC | Wakeup |
| I2S | Functional when using an external bit clock or external master clock |
| CAN | |
| Tamper detect | Interrupt or a reset |
| NMI | Non-maskable interrupt |

## 3.2.4 JTAG Controller Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

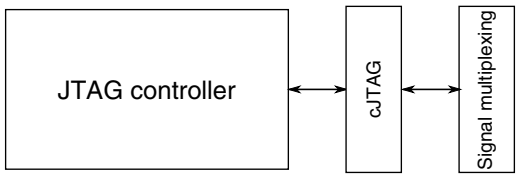



**Figure 3-4. JTAGC Controller configuration**

**Table 3-8. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | JTAGC | JTAGC |
| Signal multiplexing | Port control | Signal multiplexing |

# 3.3 System modules

## 3.3.1 SIM Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

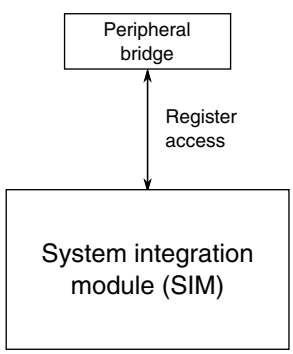



**Figure 3-5. SIM configuration**

**Table 3-9. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | SIM | SIM |
| System memory map | | System memory map |
| Clocking | | Clock distribution |
| Power management | | Power management |

## 3.3.2 System Mode Controller (SMC) Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

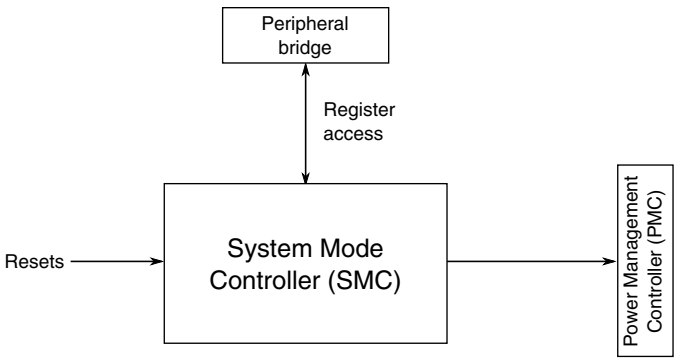



**Figure 3-6. System Mode Controller configuration**

**Table 3-10. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | System Mode Controller (SMC) | SMC |
| System memory map | | System memory map |
| Power management | | Power management |
| | Power management controller (PMC) | PMC |
| | Low-Leakage Wakeup Unit (LLWU) | LLWU |
| | Reset Control Module (RCM) | Reset |

## 3.3.3 PMC Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

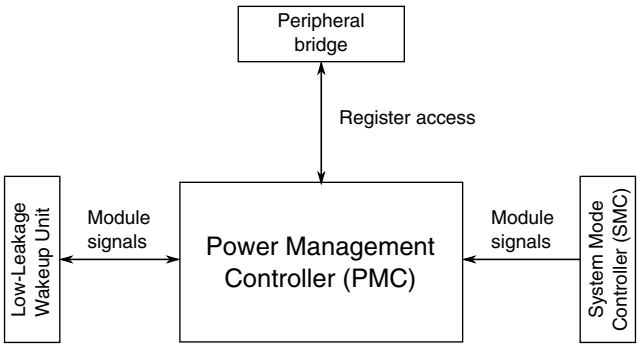



**Figure 3-7. PMC configuration**

**Table 3-11. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | PMC | PMC |
| System memory map | | System memory map |
| Power management | | Power management |
| Full description | System Mode Controller (SMC) | System Mode Controller |
| | Low-Leakage Wakeup Unit (LLWU) | LLWU |
| | Reset Control Module (RCM) | Reset |

## 3.3.4 Low-Leakage Wake-up Unit (LLWU) Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

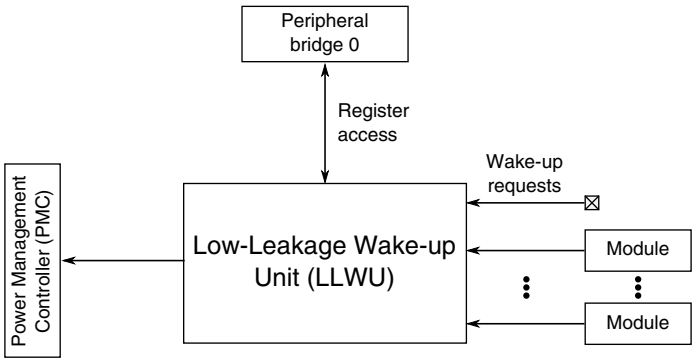



**Figure 3-8. Low-Leakage Wake-up Unit configuration**

**Table 3-12. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | LLWU | LLWU |
| System memory map | | System memory map |
| Clocking | | Clock distribution |
| Power management | | Power management chapter |
| | Power Management Controller (PMC) | Power Management Controller (PMC) |
| | Mode Controller | Mode Controller |
| Wake-up requests | | LLWU wake-up sources |

### 3.3.4.1 Wake-up Sources

This chip uses the following internal peripheral and external pin inputs as wakeup sources to the LLWU module:

- LLWU_P0-15 are external pin inputs. Any digital function multiplexed on the pin can be selected as the wakeup source. See the chip's signal multiplexing table for the digital signal options.
- LLWU_M0IF-M7IF are connections to the internal peripheral interrupt flags.

**NOTE**

$\overline{\text{RESET}}$ is also a wakeup source, depending on the bit setting in the LLWU_RST register. On devices where $\overline{\text{RESET}}$ is not a dedicated pin, it must also be enabled in the explicit port mux control.

**Table 3-13. Wakeup sources for LLWU inputs**

| Input | Wakeup source | | Input | Wakeup source |
|---|---|---|---|---|
| LLWU_P0 | PTE1/LLWU_P0 pin | | LLWU_P12 | PTD0/LLWU_P12 pin |
| LLWU_P1 | PTE2/LLWU_P1 pin | | LLWU_P13 | PTD2/LLWU_P13 pin |
| LLWU_P2 | PTE4/LLWU_P2 pin | | LLWU_P14 | PTD4/LLWU_P14 pin |
| LLWU_P3 | PTA4/LLWU_P3 pin [1] | | LLWU_P15 | PTD6/LLWU_P15 pin |
| LLWU_P4 | PTA13/LLWU_P4 pin | | LLWU_M0IF | LPTMR[2] |
| LLWU_P5 | PTB0/LLWU_P5 pin | | LLWU_M1IF | CMP0[2] |
| LLWU_P6 | PTC1/LLWU_P6 pin | | LLWU_M2IF | CMP1[2] |
| LLWU_P7 | PTC3/LLWU_P7 pin | | LLWU_M3IF | CMP2[2] |
| LLWU_P8 | PTC4/LLWU_P8 pin | | LLWU_M4IF | Reserved |
| LLWU_P9 | PTC5/LLWU_P9 pin | | LLWU_M5IF | RTC Alarm[2] |
| LLWU_P10 | PTC6/LLWU_P10 pin | | LLWU_M6IF | Reserved |
| LLWU_P11 | PTC11/LLWU_P11 pin | | LLWU_M7IF | RTC Seconds[2] |

1. The $\overline{\text{EZP\_CS}}$ signal is checked only on *Chip Reset not VLLS*, so a VLLS wakeup via a non-reset source does not cause EzPort mode entry. If NMI was enabled on entry to LLS/VLLS, asserting the NMI pin generates an NMI interrupt on exit from the low power mode.
2. Requires the peripheral and the peripheral interrupt to be enabled. The LLWU's WUME bit enables the internal module flag as a wakeup input. After wakeup, the flags are cleared based on the peripheral clearing mechanism.

## 3.3.5 MCM Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

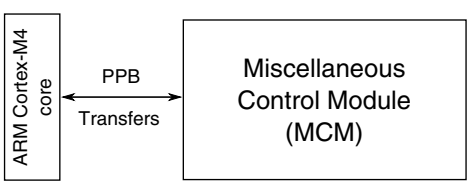



**Figure 3-9. MCM configuration**

**Table 3-14. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | Miscellaneous control module (MCM) | MCM |
| System memory map | | System memory map |
| Clocking | | Clock distribution |
| Power management | | Power management |
| Transfers<br>Private Peripheral Bus (PPB) | ARM Cortex-M4 core | ARM Cortex-M4 core |

## 3.3.6 Crossbar Switch Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

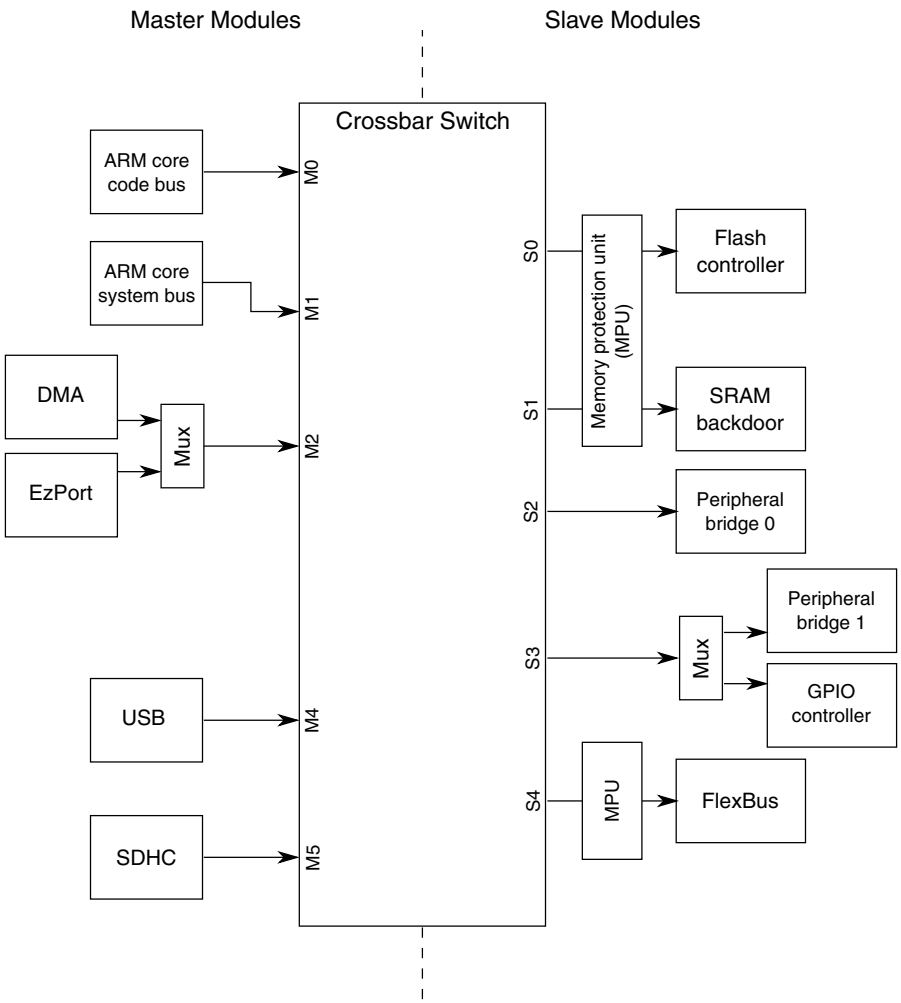



**Figure 3-10. Crossbar switch integration**

### NOTE

SDHC is not available on all packages. See the Pinout section for more information.

**Table 3-15. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | Crossbar switch | Crossbar Switch |
| System memory map | | System memory map |

*Table continues on the next page...*

**Table 3-15. Reference links to related information (continued)**

| Topic | Related module | Reference |
|---|---|---|
| Clocking | | Clock Distribution |
| Memory protection | MPU | MPU |
| Crossbar switch master | ARM Cortex-M4 core | ARM Cortex-M4 core |
| Crossbar switch master | DMA controller | DMA controller |
| Crossbar switch master | EzPort | EzPort |
| Crossbar switch master | USB FS/LS | USB FS/LS |
| Crossbar switch master | SDHC | SDHC |
| Crossbar switch slave | Flash | Flash |
| Crossbar switch slave | Peripheral bridges | Peripheral bridge |
| Crossbar switch slave | GPIO controller | GPIO controller |
| Crossbar switch slave | FlexBus | FlexBus |

### 3.3.6.1 Crossbar Switch Master Assignments

The masters connected to the crossbar switch are assigned as follows:

| Master module | Master port number |
|---|---|
| ARM core code bus | 0 |
| ARM core system bus | 1 |
| DMA/EzPort | 2 |
| USB OTG | 4 |
| SDHC | 5 |

**NOTE**

The DMA and EzPort share a master port. Since these modules never operate at the same time, no configuration or arbitration explanations are necessary.

### 3.3.6.2 Crossbar Switch Slave Assignments

The slaves connected to the crossbar switch are assigned as follows:

| Slave module | Slave port number | Protected by MPU? |
|---|---|---|
| Flash memory controller | 0 | Yes |
| SRAM backdoor | 1 | Yes |
| Peripheral bridge 0[1] | 2 | No. Protection built into bridge. |

*Table continues on the next page...*

| Slave module | Slave port number | Protected by MPU? |
|---|---|---|
| Peripheral bridge 1/GPIO[1] | 3 | No. Protection built into bridge. |
| FlexBus | 4 | Yes |

1. See System memory map for access restrictions.

### 3.3.6.3 PRS register reset values

The AXBS_PRS*n* registers reset to 0043_0210h.

## 3.3.7 Memory Protection Unit (MPU) Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

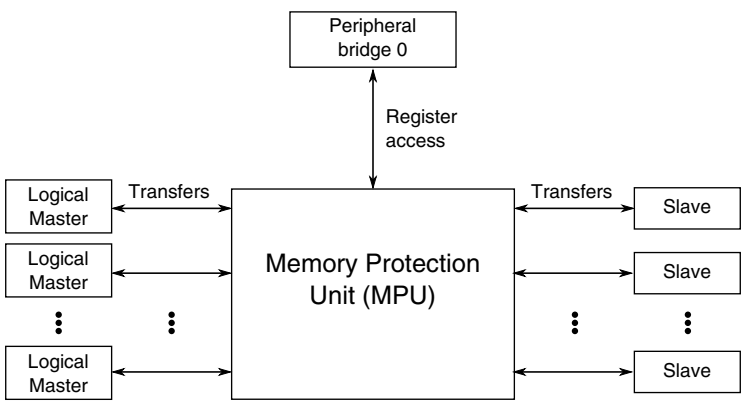



**Figure 3-11. Memory Protection Unit configuration**

**Table 3-16. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | Memory Protection Unit (MPU) | MPU |
| System memory map | | System memory map |
| Clocking | | Clock distribution |
| Power management | | Power management |
| Logical masters | | Logical master assignments |
| Slave modules | | Slave module assignments |

### 3.3.7.1 MPU Slave Port Assignments

The memory-mapped resources protected by the MPU are:

**Table 3-17. MPU Slave Port Assignments**

| Source | MPU Slave Port Assignment | Destination |
|---|---|---|
| Crossbar slave port 0 | MPU slave port 0 | Flash Controller |
| Crossbar slave port 1 | MPU slave port 1 | SRAM backdoor |
| Code Bus | MPU slave port 2 | SRAM_L frontdoor |
| System Bus | MPU slave port 3 | SRAM_U frontdoor |
| Crossbar slave port 4 | MPU slave port 4 | FlexBus |

### 3.3.7.2 MPU Logical Bus Master Assignments

The logical bus master assignments for the MPU are:

**Table 3-18. MPU Logical Bus Master Assignments**

| MPU Logical Bus Master Number | Bus Master |
|---|---|
| 0 | Core |
| 1 | Debugger |
| 2 | DMA |
| 3 | none |
| 4 | USB |
| 5 | SDHC |
| 6 | none |
| 7 | none |

### 3.3.7.3 MPU Access Violation Indications

Access violations detected by the MPU are signaled to the appropriate bus master as shown below:

**Table 3-19. Access Violation Indications**

| Bus Master | Core Indication |
|---|---|
| Core | Bus fault (interrupt vector #5) Note: To enable bus faults set the core's System Handler Control and State Register's BUSFAULTENA bit. If this bit is not set, MPU violations result in a hard fault (interrupt vector #3). |
| Debugger | The STICKYERROR flag is set in the Debug Port Control/Status Register. |
| DMA | Interrupt vector #32 |
| USB_OTG | Interrupt vector #69 |

*Table continues on the next page...*

**Table 3-19. Access Violation Indications (continued)**

| Bus Master | Core Indication |
|---|---|
| SDHC | Interrupt vector #97 |

### 3.3.7.4 Reset Values for RGD0 Registers

At reset, the MPU is enabled with a single region descriptor (RGD0) that maps the entire 4 GB address space with read, write and execute permissions given to the core, debugger and the DMA bus masters.

The following table shows the chip-specific reset values for RGD0 and RGDAAC0.

**Table 3-20. Reset Values for RGD0 Registers**

| Register | Reset value |
|---|---|
| RGD0_WORD0 | 0000_0000h |
| RGD0_WORD1 | FFFF_FFFFh |
| RGD0_WORD2 | 0061_F7DFh |
| RGD0_WORD3 | 0000_0001h |
| RGDAAC0 | 0061_F7DFh |

### 3.3.7.5 Write Access Restrictions for RGD0 Registers

In addition to configuring the initial state of RGD0, the MPU implements further access control on writes to the RGD0 registers. Specifically, the MPU assigns a priority scheme where the debugger is treated as the highest priority master followed by the core and then all the remaining masters.

The MPU does not allow writes from the core to affect the RGD0 start or end addresses nor the permissions associated with the debugger; it can only write the permission fields associated with the other masters.

These protections (summarized below) guarantee that the debugger always has access to the entire address space and those rights cannot be changed by the core or any other bus master.

**Table 3-21. Write Access to RGD0 Registers**

| Bus Master | Write Access? |
|---|---|
| Core | Partial. The Core cannot write to the following registers or register fields:<br>• RGD0_WORD0, RGD0_WORD1, RGD0_WORD3<br>• RGD0_WORD2[M1SM, M1UM]<br>• RGDAAC0[M1SM, M1UM]<br><br>**NOTE:** Changes to the RGD0_WORD2 alterable fields should be done via a write to RGDAAC0. |
| Debugger | Yes |
| All other masters | No |

## 3.3.8 Peripheral Bridge Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

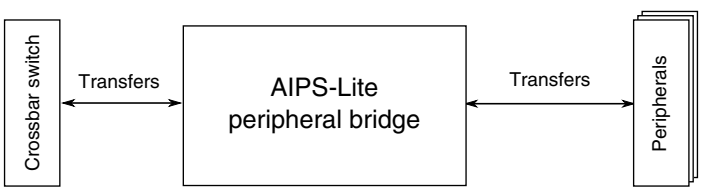



**Figure 3-12. Peripheral bridge configuration**

**Table 3-22. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | Peripheral bridge (AIPS-Lite) | Peripheral bridge (AIPS-Lite) |
| System memory map | | System memory map |
| Clocking | | Clock Distribution |
| Crossbar switch | Crossbar switch | Crossbar switch |

### 3.3.8.1 Number of peripheral bridges

This device contains.

### 3.3.8.2 Memory maps

The peripheral bridges are used to access the registers of most of the modules on this device. See AIPS0 Memory Map and AIPS1 Memory Map for the memory slot assignment for each module.

## 3.3.9 DMA request multiplexer configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

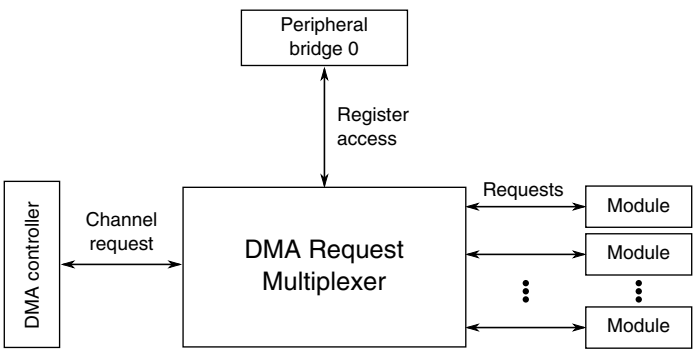



**Figure 3-13. DMA request multiplexer configuration**

**Table 3-23. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | DMA request multiplexer | DMA Mux |
| System memory map | | System memory map |
| Clocking | | Clock distribution |
| Power management | | Power management |
| Channel request | DMA controller | DMA Controller |
| Requests | | DMA request sources |

### 3.3.9.1 DMA MUX request sources

This device includes a DMA request mux that allows up to 63 DMA request signals to be mapped to any of the 16 DMA channels. Because of the mux there is not a hard correlation between any of the DMA request sources and a specific DMA channel.

## Table 3-24. DMA request sources - MUX 0

| Source number | Source module | Source description |
|---|---|---|
| 0 | — | Channel disabled[1] |
| 1 | Reserved | Not used |
| 2 | UART0 | Receive |
| 3 | UART0 | Transmit |
| 4 | UART1 | Receive |
| 5 | UART1 | Transmit |
| 6 | UART2 | Receive |
| 7 | UART2 | Transmit |
| 8 | UART3 | Receive |
| 9 | UART3 | Transmit |
| 10 | UART4 | Transmit or Receive |
| 11 | UART5 | Transmit or Receive |
| 12 | $I^2S0$ | Receive |
| 13 | $I^2S0$ | Transmit |
| 14 | SPI0 | Receive |
| 15 | SPI0 | Transmit |
| 16 | SPI1 | Transmit or Receive Transmit or Receive |
| 17 | SPI2 | Transmit or Receive |
| 18 | $I^2C0$ | — |
| 19 | $I^2C1$ or $I^2C2$ | — |
| 20 | FTM0 | Channel 0 |
| 21 | FTM0 | Channel 1 |
| 22 | FTM0 | Channel 2 |
| 23 | FTM0 | Channel 3 |
| 24 | FTM0 | Channel 4 |
| 25 | FTM0 | Channel 5 |
| 26 | FTM0 | Channel 6 |
| 27 | FTM0 | Channel 7 |
| 28 | FTM1 | Channel 0 |
| 29 | FTM1 | Channel 1 |
| 30 | FTM2 | Channel 0 |
| 31 | FTM2 | Channel 1 |
| 32 | FTM3 | Channel 0 |
| 33 | FTM3 | Channel 1 |
| 34 | FTM3 | Channel 2 |
| 35 | FTM3 | Channel 3 |
| 36 | FTM3 | Channel 4 |
| 37 | FTM3 | Channel 5 |
| 38 | FTM3 | Channel 6 |

*Table continues on the next page...*

**Table 3-24. DMA request sources - MUX 0 (continued)**

| Source number | Source module | Source description |
|---|---|---|
| 39 | FTM3 | Channel 7 |
| 40 | ADC0 | — |
| 41 | ADC1 | — |
| 42 | CMP0 | — |
| 43 | CMP1 | — |
| 44 | CMP2 | — |
| 45 | DAC0 | — |
| 46 | DAC1 | — |
| 47 | CMT | — |
| 48 | PDB | — |
| 49 | Port control module | Port A |
| 50 | Port control module | Port B |
| 51 | Port control module | Port C |
| 52 | Port control module | Port D |
| 53 | Port control module | Port E |
| 54 | DMA MUX | Always enabled |
| 55 | DMA MUX | Always enabled |
| 56 | DMA MUX | Always enabled |
| 57 | DMA MUX | Always enabled |
| 58 | DMA MUX | Always enabled |
| 59 | DMA MUX | Always enabled |
| 60 | DMA MUX | Always enabled |
| 61 | DMA MUX | Always enabled |
| 62 | DMA MUX | Always enabled |
| 63 | DMA MUX | Always enabled |

1. Configuring a DMA channel to select source 0 or any of the reserved sources disables that DMA channel.

### 3.3.9.2 DMA transfers via PIT trigger

The PIT module can trigger a DMA transfer on the first four DMA channels. The assignments are detailed at PIT/DMA Periodic Trigger Assignments .

## 3.3.10 DMA Controller Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

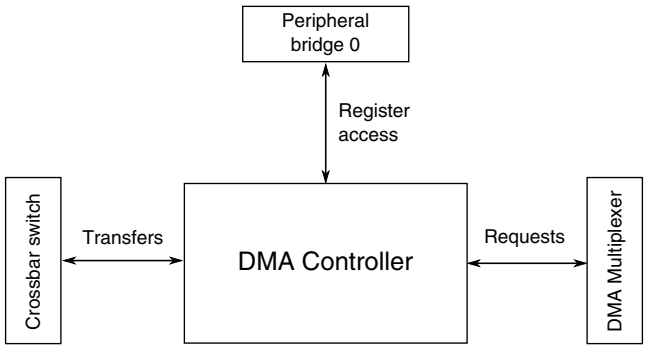



**Figure 3-14. DMA Controller configuration**

**Table 3-25. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | DMA Controller | DMA Controller |
| System memory map | | System memory map |
| Register access | Peripheral bridge (AIPS-Lite 0) | AIPS-Lite 0 |
| Clocking | | Clock distribution |
| Power management | | Power management |
| Transfers | Crossbar switch | Crossbar switch |

## 3.3.11 External Watchdog Monitor (EWM) Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

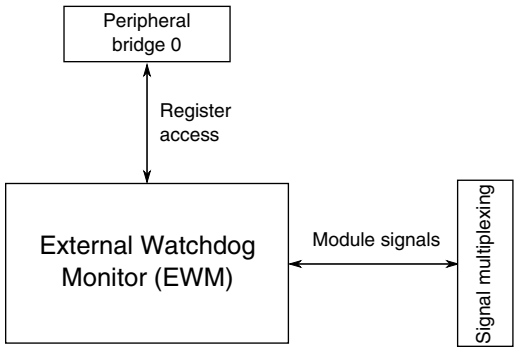



**Figure 3-15. External Watchdog Monitor configuration**

**Table 3-26. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | External Watchdog Monitor (EWM) | EWM |
| System memory map | | System memory map |
| Clocking | | Clock distribution |
| Power management | | Power management |
| Signal multiplexing | Port Control Module | Signal multiplexing |

### 3.3.11.1 EWM clocks

This table shows the EWM clocks and the corresponding chip clocks.

**Table 3-27. EWM clock connections**

| Module clock | Chip clock |
|---|---|
| Low Power Clock | 1 kHz LPO Clock |

### 3.3.11.2 EWM low-power modes

This table shows the EWM low-power modes and the corresponding chip low-power modes.

**Table 3-28. EWM low-power modes**

| Module mode | Chip mode |
|---|---|
| Wait | Wait, VLPW |
| Stop | Stop, VLPS, LLS |

### 3.3.11.3 $\overline{\text{EWM\_OUT}}$ pin state in low power modes

During Wait, Stop, and Power Down modes the $\overline{\text{EWM\_OUT}}$ pin preserve its state before entering Wait or Stop mode. When the CPU enters a Run mode from Wait or Stop recovery, the pin resumes its previous state before entering Wait or Stop mode. When the CPU enters Run mode from Power Down, the pin returns to its reset state.

## 3.3.12 Watchdog Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

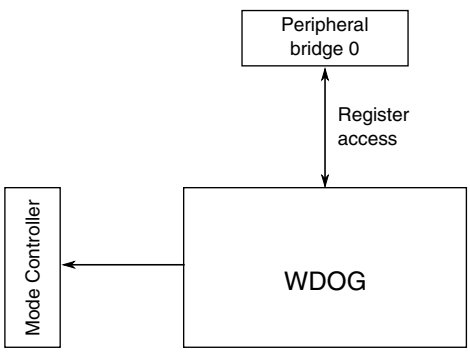



**Figure 3-16. Watchdog configuration**

**Table 3-29. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | Watchdog | Watchdog |
| System memory map | | System memory map |
| Clocking | | Clock distribution |
| Power management | | Power management |
| | Mode Controller (MC) | |

### 3.3.12.1 WDOG clocks

This table shows the WDOG module clocks and the corresponding chip clocks.

**Table 3-30. WDOG clock connections**

| Module clock | Chip clock |
|---|---|
| LPO Oscillator | 1 kHz LPO Clock |
| Alt Clock | Bus Clock |
| Fast Test Clock | Bus Clock |
| System Bus Clock | Bus Clock |

### 3.3.12.2 WDOG low-power modes

This table shows the WDOG low-power modes and the corresponding chip low-power modes.

**Table 3-31. WDOG low-power modes**

| Module mode | Chip mode |
|---|---|
| Wait | Wait, VLPW |
| Stop | Stop, VLPS |
| Power Down | LLS, VLLSx |

## 3.4 Clock modules

### 3.4.1 MCG Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

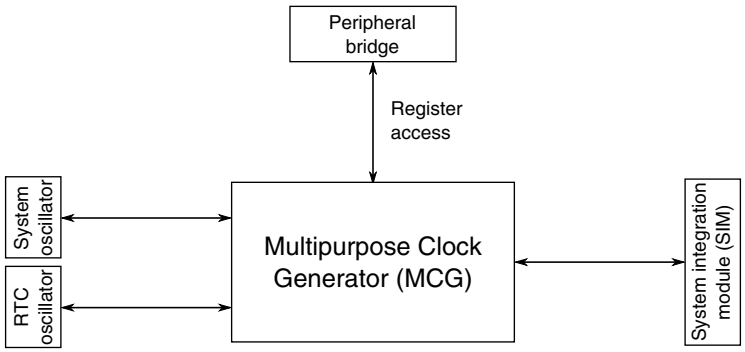



**Figure 3-17. MCG configuration**

**Table 3-32. Reference links to related information**

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

## 3.4.2 OSC Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

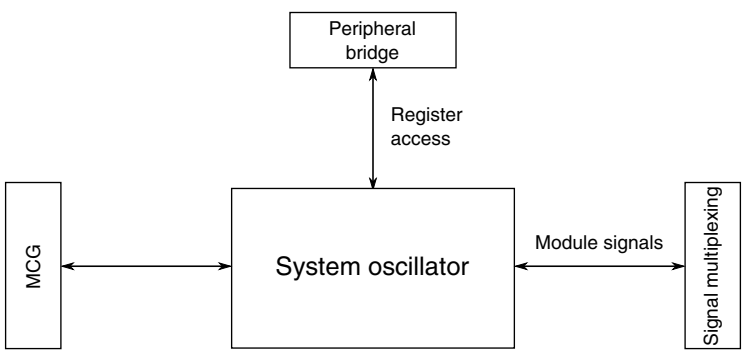



**Figure 3-18. OSC configuration**

**Table 3-33. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | OSC | OSC |
| System memory map | | System memory map |
| Clocking | | Clock distribution |

*Table continues on the next page...*

**Table 3-33. Reference links to related information (continued)**

| Topic | Related module | Reference |
|---|---|---|
| Power management | | Power management |
| Signal multiplexing | Port control | Signal multiplexing |
| Full description | MCG | MCG |

### 3.4.2.1 OSC modes of operation with MCG

The MCG's C2 register bits configure the oscillator frequency range. See the OSC and MCG chapters for more details.

## 3.4.3 RTC OSC configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

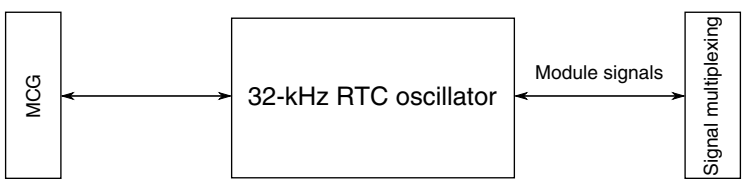



**Figure 3-19. RTC OSC configuration**

**Table 3-34. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | RTC OSC | RTC OSC |
| Signal multiplexing | Port control | Signal multiplexing |
| Full description | MCG | MCG |

# 3.5 Memories and memory interfaces

## 3.5.1 Flash Memory Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

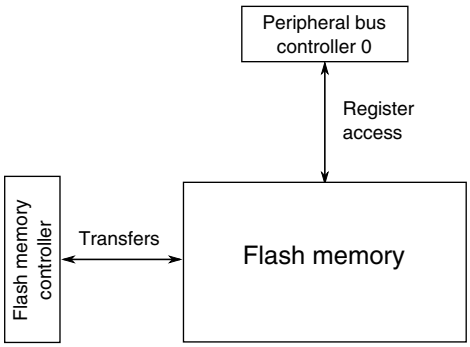



**Figure 3-20. Flash memory configuration**

**Table 3-35. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | Flash memory | Flash memory |
| System memory map | | System memory map |
| Clocking | | Clock Distribution |
| Transfers | Flash memory controller | Flash memory controller |
| Register access | Peripheral bridge | Peripheral bridge |

### 3.5.1.1 Flash memory types

This device contains the following types of flash memory:

- Program flash memory — non-volatile flash memory that can execute program code
- FlexMemory — encompasses the following memory types:
  - For devices with FlexNVM: FlexNVM — Non-volatile flash memory that can execute program code, store data, or backup EEPROM data
  - For devices with FlexNVM: FlexRAM — RAM memory that can be used as traditional RAM or as high-endurance EEPROM storage, and also accelerates flash programming
  - For devices with only program flash memory: Programming acceleration RAM — RAM memory that accelerates flash programming

### 3.5.1.2 Flash Memory Sizes

The devices covered in this document contain:

- For devices with program flash only:

- 2 blocks (512 KB each) blocks of program flash consisting of 4 KB sectors
- 1 block of programming Acceleration RAM

- For devices that contain FlexNVM:
  - 1 block (512 KB) of program flash consisting of 4 KB sectors
  - 1 block (128 KB) of FlexNVM consisting of 4 KB sectors
  - 1 block of FlexRAM

The amounts of flash memory for the devices covered in this document are:

| Device | Program flash (KB) | Block 0 (P-Flash) address range[1] | FlexNVM (KB) | Block 1 (FlexNVM/ P-Flash) address range[1] | FlexRAM/ Programming Acceleration RAM (KB) | FlexRAM/ Programming Acceleration RAM address range |
|---|---|---|---|---|---|---|
| MK21FX512VLQ12 | 512 | 0x0000_0000–0x0007_FFFF | 128 | 0x1000_0000–0x1001_FFFF | 4 | 0x1400_0000–0x1400_0FFF |
| MK21FN1M0VLQ12 | 1024 | 0x0000_0000–0x0007_FFFF | — | 0x0008_0000–0x000F_FFFF | 4 | 0x1400_0000–0x1400_0FFF |
| MK21FX512VMD12 | 512 | 0x0000_0000–0x0007_FFFF | 128 | 0x1000_0000–0x1001_FFFF | 4 | 0x1400_0000–0x1400_0FFF |
| MK21FN1M0VMD12 | 1024 | 0x0000_0000–0x0007_FFFF | — | 0x0008_0000–0x000F_FFFF | 4 | 0x1400_0000–0x1400_0FFF |

1. For program flash only devices: The addresses shown assume program flash swap is disabled (default configuration).

### 3.5.1.3 Flash Memory Size Considerations

Since this document covers devices that contain program flash only and devices that contain program flash and FlexNVM, there are some items to consider when reading the flash memory chapter.

- The flash memory chapter shows a mixture of information depending on the device you are using.
- For the program flash only devices:
  - Two program flash blocks are supported: block 0 and block 1. The two blocks are contiguous in the memory map.
  - The program flash blocks support a swap feature in which the starting address of the program flash blocks can be swapped.
  - The FlexRAM is not available as EEPROM or traditional RAM. Its space is used only for programming acceleration through the Program Section command.
  - The programming acceleration RAM is used for the Program Section command.
- For the devices containing program flash and FlexNVM:
  - Since there is only one program flash block, the program flash swap feature is not available.

### 3.5.1.4 Flash Memory Map

The various flash memories and the flash registers are located at different base addresses as shown in the following figure. The base address for each is specified in System memory map.

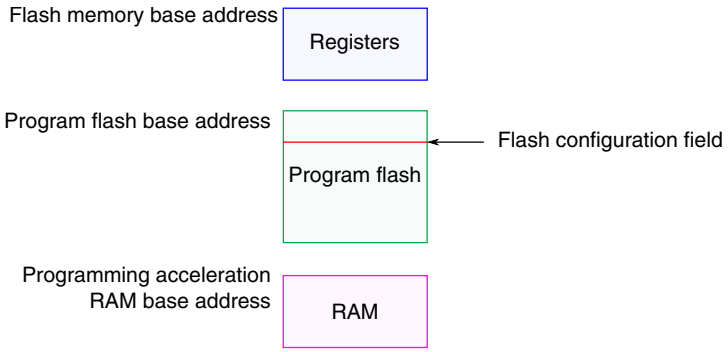



**Figure 3-21. Flash memory map for devices containing only program flash**

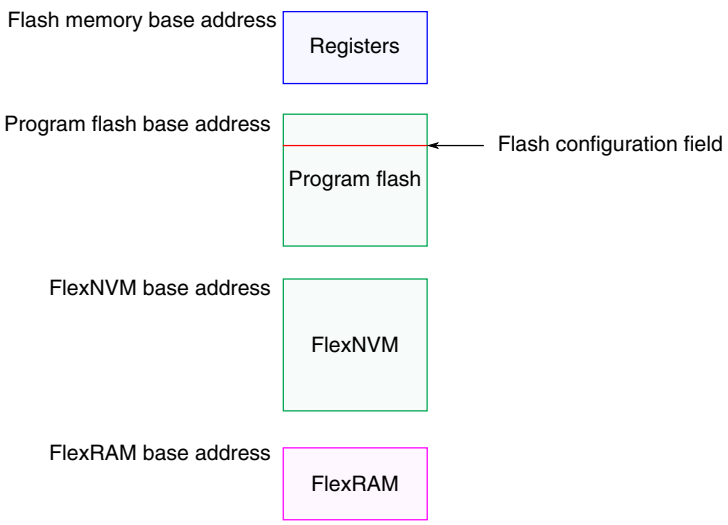



**Figure 3-22. Flash memory map for devices containing FlexNVM**

### 3.5.1.5 Flash Security

How flash security is implemented on this device is described in Chip Security.

#### 3.5.1.6 Flash Modes

The flash memory operates in NVM normal and NVM special modes. The flash memory enters NVM special mode when the EzPort is enabled ($\overline{\text{EZP\_CS}}$ asserted during reset). Otherwise, flash memory operates in NVM normal mode.

#### 3.5.1.7 Erase All Flash Contents

An Erase All Flash Blocks operation can be launched by software through a series of peripheral bus writes to flash registers. In addition the entire flash memory may be erased external to the flash memory from the SWJ-DP debug port by setting DAP_CONTROL[0]. DAP_STATUS[0] is set to indicate the mass erase command has been accepted. DAP_STATUS[0] is cleared when the mass erase completes.

The EzPort can also initiate an erase of flash contents by issuing a bulk erase (BE) command. See the EzPort chapter for more details.

#### 3.5.1.8 FTF_FOPT Register

The flash memory's FTF_FOPT register allows the user to customize the operation of the MCU at boot time. See FOPT boot options for details of its definition.

### 3.5.2 Flash Memory Controller Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

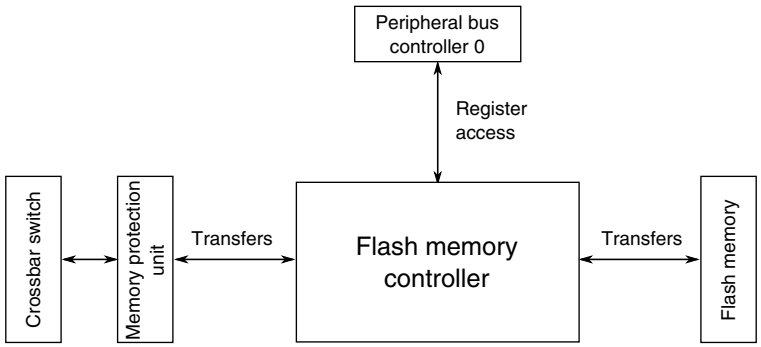



**Figure 3-23. Flash memory controller configuration**

**Table 3-36. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | Flash memory controller | Flash memory controller |
| System memory map | | System memory map |
| Clocking | | Clock Distribution |
| Transfers | Flash memory | Flash memory |
| Transfers | MPU | MPU |
| Transfers | Crossbar switch | Crossbar Switch |
| Register access | Peripheral bridge | Peripheral bridge |

### 3.5.2.1 Number of masters

The Flash Memory Controller supports up to eight crossbar switch masters. However, this device has a different number of crossbar switch masters. See Crossbar Switch Configuration for details on the master port assignments.

### 3.5.2.2 Program Flash Swap

On devices that contain program flash memory only, the program flash memory blocks may swap their base addresses.

While not using swap:

- FMC_PFB0CR controls the lower code addresses (block 0)
- FMC_PFB1CR controls the upper code addresses (block 1)

If swap is used, the opposite is true:

- FMC_PFB0CR controls the upper code addresses (now in block 0)
- FMC_PFB1CR controls the lower code addresses (now in block 1)

### 3.5.3 SRAM Configuration

This section summarizes how the module has been configured in the chip.

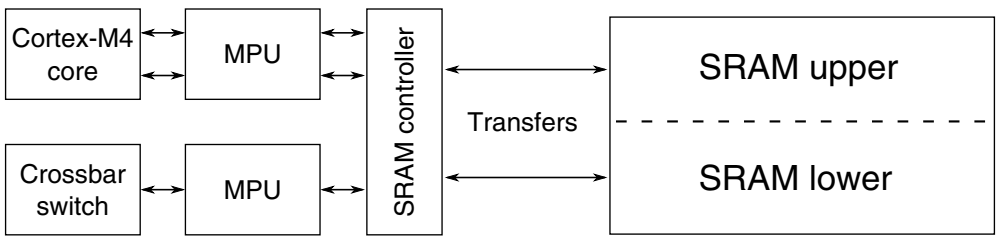



**Figure 3-24. SRAM configuration**

**Table 3-37. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | SRAM | SRAM |
| System memory map | | System memory map |
| Clocking | | Clock Distribution |
| Transfers | SRAM controller | SRAM controller |
| | ARM Cortex-M4 core | ARM Cortex-M4 core |
| | Memory protection unit | Memory protection unit |

#### 3.5.3.1 SRAM sizes

This device contains SRAM The amount of SRAM for the devices covered in this document is shown in the following table.

| Device | SRAM (KB) |
|---|---|
| MK21FX512VLQ12 | 128 |
| MK21FN1M0VLQ12 | 128 |
| MK21FX512VMD12 | 128 |
| MK21FN1M0VMD12 | 128 |

#### 3.5.3.2 SRAM Arrays

The on-chip SRAM is split into two equally-sized logical arrays, SRAM_L and SRAM_U.

The on-chip RAM is implemented such that the SRAM_L and SRAM_U ranges form a contiguous block in the memory map. As such:

- SRAM_L is anchored to 0x1FFF_FFFF and occupies the space before this ending address.
- SRAM_U is anchored to 0x2000_0000 and occupies the space after this beginning address.

Valid address ranges for SRAM_L and SRAM_U are then defined as:

- SRAM_L = [0x2000_0000–(SRAM_size/2)] to 0x1FFF_FFFF
- SRAM_U = 0x2000_0000 to [0x2000_0000+(SRAM_size/2)-1]

This is illustrated in the following figure.

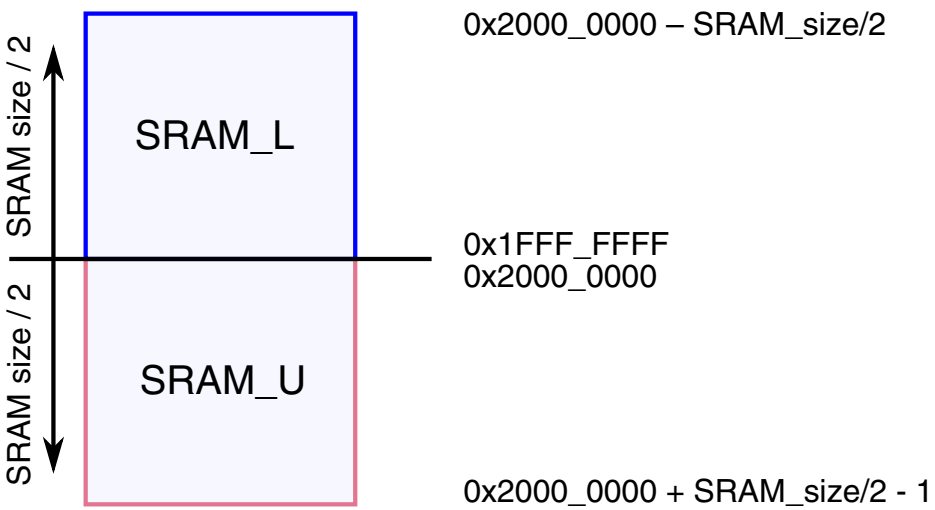



**Figure 3-25. SRAM blocks memory map**

For example, for a device containing 64 KB of SRAM the ranges are:

- SRAM_L: 0x1FFF_8000 – 0x1FFF_FFFF
- SRAM_U: 0x2000_0000 – 0x2000_7FFF

### 3.5.3.3 SRAM retention in low power modes

The SRAM is retained down to VLLS3 mode.

In VLLS2 the 4 or 16 KB (user option) region of SRAM_U from 0x2000_0000 is powered. These different regions (or partitions) of SRAM are labeled as follows:

- RAM1: the 4 KB region always powered in VLLS2
- RAM2: the additional 12 KB region optionally powered in VLLS2
- RAM3: the rest of system RAM

In VLLS1 and VLLS0 no SRAM is retained; however, the 32-byte register file is available.

### 3.5.3.4 SRAM accesses

The SRAM is split into two logical arrays that are 32-bits wide.

- SRAM_L — Accessible by the code bus of the Cortex-M4 core and by the backdoor port.
- SRAM_U — Accessible by the system bus of the Cortex-M4 core and by the backdoor port.

The backdoor port makes the SRAM accessible to the non-core bus masters (such as DMA).

The following figure illustrates the SRAM accesses within the device.

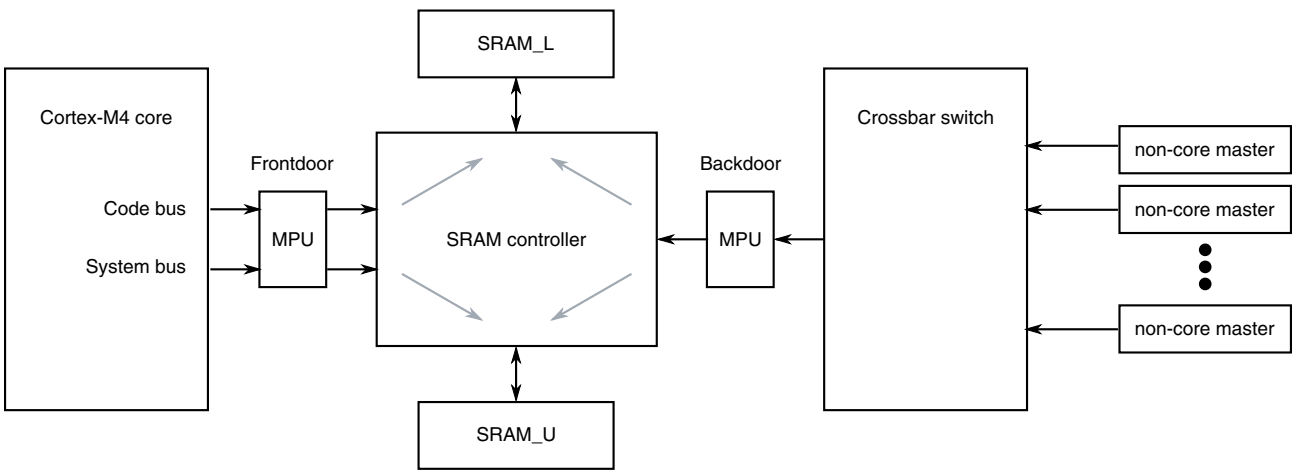



**Figure 3-26. SRAM access diagram**

The following simultaneous accesses can be made to different logical regions of the SRAM:

- Core code and core system
- Core code and non-core master
- Core system and non-core master

**NOTE**

Two non-core masters cannot access SRAM simultaneously. The required arbitration and serialization is provided by the crossbar switch. The SRAM_{L,U} arbitration is controlled by the SRAM controller based on the configuration bits in the MCM module.

**NOTE**

Burst-access cannot occur across the 0x2000_0000 boundary that separates the two SRAM arrays. The two arrays should be treated as separate memory ranges for burst accesses.

## 3.5.4 System Register File Configuration

This section summarizes how the module has been configured in the chip.

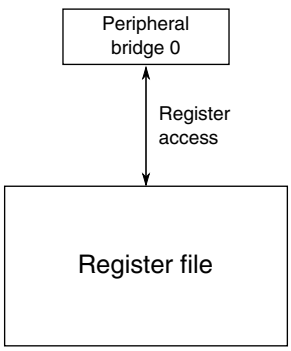



**Figure 3-27. System Register file configuration**

**Table 3-38. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | Register file | Register file |
| System memory map | | System memory map |
| Clocking | | Clock distribution |
| Power management | | Power management |

### 3.5.4.1 System Register file

This device includes a 32-byte register file that is powered in all power modes.

Also, it retains contents during low-voltage detect (LVD) events and is only reset during a power-on reset.

## 3.5.5 VBAT Register File Configuration

This section summarizes how the module has been configured in the chip.

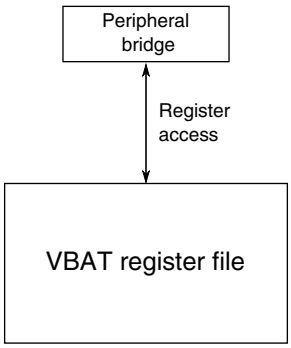



**Figure 3-28. VBAT Register file configuration**

**Table 3-39. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | VBAT register file | VBAT register file |
| System memory map | | System memory map |
| Clocking | | Clock distribution |
| Power management | | Power management |

### 3.5.5.1 VBAT register file

This device includes a 32-byte register file that is powered in all power modes and is powered by VBAT.

It is only reset during VBAT power-on reset.

## 3.5.6 EzPort Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.


**Figure 3-29. EzPort configuration**

**Table 3-40. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | EzPort | EzPort |

*Table continues on the next page...*

**Table 3-40. Reference links to related information (continued)**

| Topic | Related module | Reference |
|---|---|---|
| System memory map | | System memory map |
| Clocking | | Clock Distribution |
| Transfers | Crossbar switch | Crossbar switch |
| Signal Multiplexing | Port control | Signal Multiplexing |

#### 3.5.6.1 JTAG instruction

The system JTAG controller implements an EZPORT instruction. When executing this instruction, the JTAG controller resets the core logic and asserts the EzPort chip select signal to force the processor into EzPort mode.

#### 3.5.6.2 Flash Option Register (FOPT)

The FOPT[EZPORT_DIS] bit can be used to prevent entry into EzPort mode during reset. If the FOPT[EZPORT_DIS] bit is cleared, then the state of the chip select signal ($\overline{\text{EZP\_CS}}$) is ignored and the MCU always boots in normal mode.

This option is useful for systems that use the $\overline{\text{EZP\_CS}}$/NMI signal configured for its NMI function. Disabling EzPort mode prevents possible unwanted entry into EzPort mode if the external circuit that drives the NMI signal asserts it during reset.

The FOPT register is loaded from the flash option byte. If the flash option byte is modified the new value takes effect for any subsequent resets, until the value is changed again.

### 3.5.7 FlexBus Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

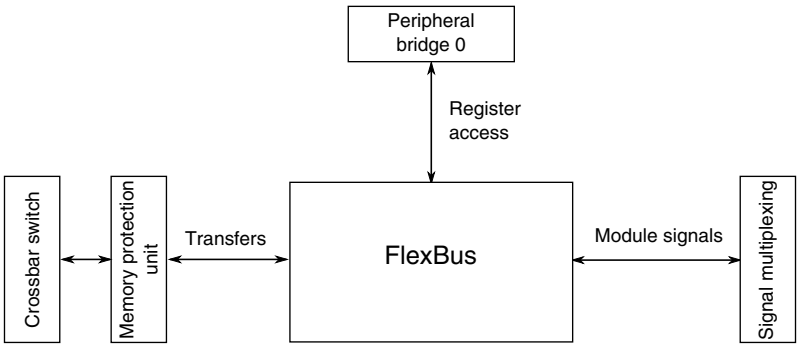



**Figure 3-30. FlexBus configuration**

**Table 3-41. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | FlexBus | FlexBus |
| System memory map | | System memory map |
| Clocking | | Clock distribution |
| Power management | | Power management |
| Transfers | Memory protection unit (MPU) | Memory protection unit (MPU) |
| Signal multiplexing | Port control | Signal multiplexing |

### 3.5.7.1 FlexBus clocking

The system provides a dedicated clock source to the FlexBus module's external CLKOUT. Its clock frequency is derived from a divider of the MCGOUTCLK. See Clock Distribution for more details.

### 3.5.7.2 FlexBus signal multiplexing

The multiplexing of the FlexBus address and data signals is controlled by the port control module. However, the multiplexing of some of the FlexBus control signals are controlled by the port control and FlexBus modules. The port control module registers control whether the FlexBus or another module signals are available on the external pin, while the FlexBus's CSPMCR register configures which FlexBus signals are available from the module. The control signals are grouped as illustrated:

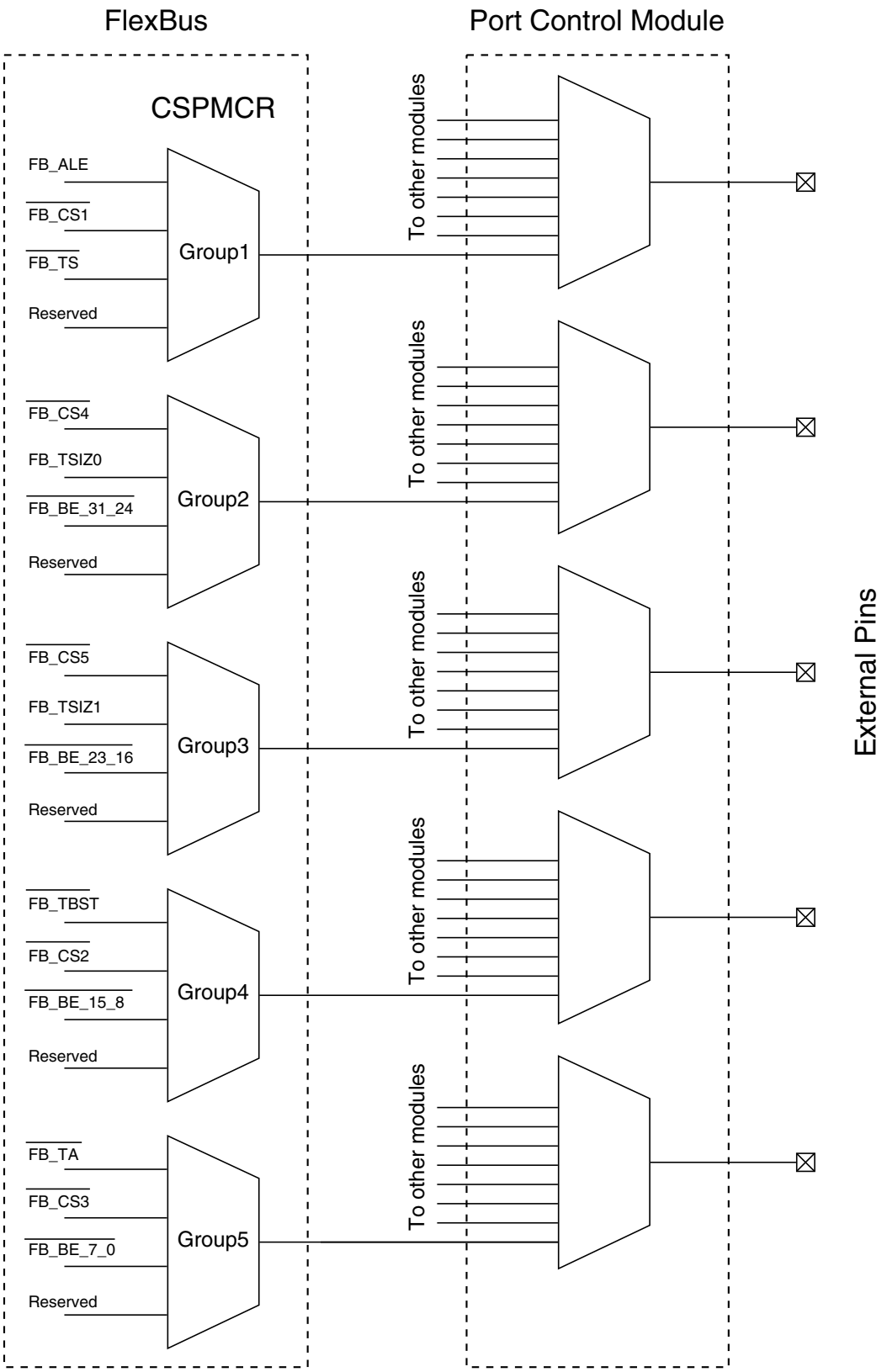



**Figure 3-31. FlexBus control signal multiplexing**

Therefore, use the CSPMCR and port control registers to configure which control signal is available on the external pin. All control signals, except for $\overline{\text{FB\_TA}}$, are assigned to the ALT5 function in the port control module. Since, unlike the other control signals, $\overline{\text{FB\_TA}}$ is an input signal, it is assigned to the ALT6 function.

#### 3.5.7.3 FlexBus CSCR0 reset value

On this device the CSCR0 resets to 0x003F_FC00. Configure this register as needed before performing any FlexBus access.

#### 3.5.7.4 FlexBus Security

When security is enabled on the device, FlexBus accesses may be restricted by configuring the field in the SIM's SOPT2 register. See System Integration Module (SIM) for details.

#### 3.5.7.5 FlexBus line transfers

Line transfers are not possible from the ARM Cortex-M4 core. Ignore any references to line transfers in the FlexBus chapter.

## 3.6 Security

### 3.6.1 CRC Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

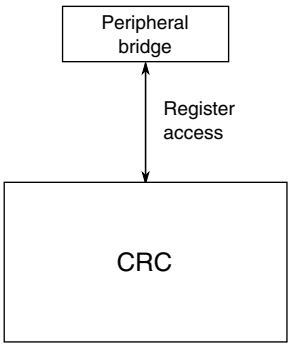



**Figure 3-32. CRC configuration**

**Table 3-42. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | CRC | CRC |
| System memory map | | System memory map |
| Power management | | Power management |

## 3.6.2 MMCAU Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

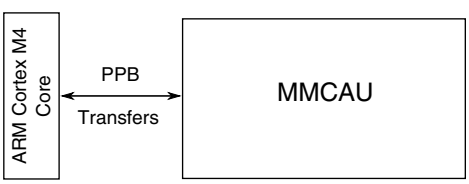



**Figure 3-33. MMCAU configuration**

**Table 3-43. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | MMCAU | MMCAU |
| System memory map | | System memory map |
| Clocking | | Clock Distribution |
| Power Management | | Power Management |
| Transfers<br>Private Peripheral Bus (PPB) | ARM Cortex M4 Core | |

## 3.6.3 RNG Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

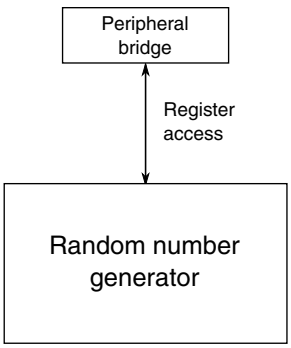



**Figure 3-34. RNG configuration**

**Table 3-44. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | RNG | RNG |
| System memory map | | System memory map |
| Clocking | | Clock distribution |
| Power management | | Power management |

### 3.6.3.1 RNGA Base Addresses

RNGA can be accessed through both AIPS0 and AIPS1. When accessed through AIPS0, the base address is 4002_9000h and when accessed through AIPS1, the base address is 400A_0000h.

## 3.6.4 DryIce (tamper detect and secure storage) configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

**NOTE**

The information on this module is available only under NDA.
Please contact your local sales office for details.

# 3.7 Analog

## 3.7.1 16-bit SAR ADC Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

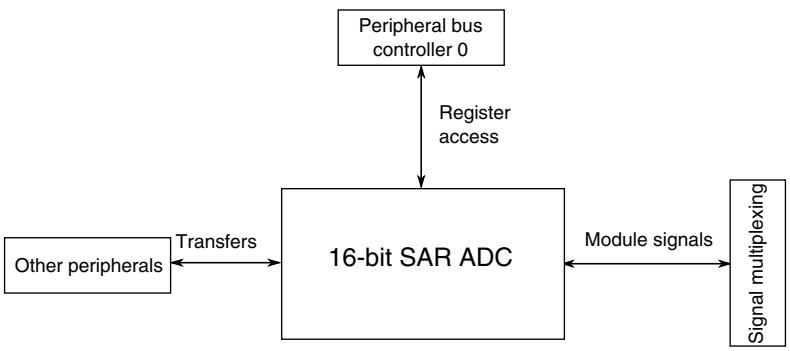



**Figure 3-35. 16-bit SAR ADC configuration**

**Table 3-45. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | 16-bit SAR ADC | 16-bit SAR ADC |
| System memory map | | System memory map |
| Clocking | | Clock distribution |
| Power management | | Power management |
| Signal multiplexing | Port control | Signal multiplexing |

### 3.7.1.1 ADC instantiation information

This device contains two ADCs.

#### 3.7.1.1.1 Number of ADC channels

The number of ADC channels present on the device is determined by the pinout of the specific device package. For details regarding the number of ADC channel available on a particular package, refer to the signal multiplexing chapter of this MCU.

### 3.7.1.2 DMA Support on ADC

Applications may require continuous sampling of the ADC (4K samples/sec) that may have considerable load on the CPU. Though using PDB to trigger ADC may reduce some CPU load, the ADC supports DMA request functionality for higher performance when the ADC is sampled at a very high rate or cases where PDB is bypassed. The ADC can trigger the DMA (via DMA req) on conversion completion.

### 3.7.1.3 Connections/channel assignment

#### 3.7.1.3.1 ADC0 Connections/Channel Assignment

**NOTE**

As indicated by the following sections, each ADCx_DPx input and certain ADCx_DMx inputs may operate as single-ended ADC channels in single-ended mode.

##### 3.7.1.3.1.1 ADC0 Channel Assignment for 144-Pin Package

| ADC Channel (SC1n[ADCH]) | Channel | Input signal (SC1n[DIFF]= 1) | Input signal (SC1n[DIFF]= 0) |
|---|---|---|---|
| 00000 | DAD0 | ADC0_DP0 and ADC0_DM0[1] | ADC0_DP0[2] |
| 00001 | DAD1 | ADC0_DP1 and ADC0_DM1[3] | ADC0_DP1 |
| 00010 | DAD2 | ADC0_DP2 and ADC0_DM2 | ADC0_DP2 |
| 00011 | DAD3 | ADC0_DP3 and ADC0_DM3[4] | ADC0_DP3[5] |
| 00100[6] | AD4a | Reserved | Reserved |
| 00101[6] | AD5a | Reserved | Reserved |
| 00110[6] | AD6a | Reserved | Reserved |
| 00111[6] | AD7a | Reserved | Reserved |
| 00100[6] | AD4b | Reserved | ADC0_SE4b |
| 00101[6] | AD5b | Reserved | ADC0_SE5b |
| 00110[6] | AD6b | Reserved | ADC0_SE6b |
| 00111[6] | AD7b | Reserved | ADC0_SE7b |
| 01000 | AD8 | Reserved | ADC0_SE8[7] |
| 01001 | AD9 | Reserved | ADC0_SE9[8] |
| 01010 | AD10 | Reserved | ADC0_SE10 |
| 01011 | AD11 | Reserved | ADC0_SE11 |
| 01100 | AD12 | Reserved | ADC0_SE12 |
| 01101 | AD13 | Reserved | ADC0_SE13 |
| 01110 | AD14 | Reserved | ADC0_SE14 |
| 01111 | AD15 | Reserved | ADC0_SE15 |

*Table continues on the next page...*

| ADC Channel (SC1n[ADCH]) | Channel | Input signal (SC1n[DIFF]= 1) | Input signal (SC1n[DIFF]= 0) |
|---|---|---|---|
| 10000 | AD16 | Reserved | ADC0_SE16 |
| 10001 | AD17 | Reserved | ADC0_SE17 |
| 10010 | AD18 | Reserved | ADC0_SE18 |
| 10011 | AD19 | Reserved | ADC0_DM0[9] |
| 10100 | AD20 | Reserved | ADC0_DM1 |
| 10101 | AD21 | Reserved | ADC0_SE21 |
| 10110 | AD22 | Reserved | ADC0_SE22 |
| 10111 | AD23 | Reserved | 12-bit DAC0 Output/ ADC0_SE23 |
| 11000 | AD24 | Reserved | Reserved |
| 11001 | AD25 | Reserved | Reserved |
| 11010 | AD26 | Temperature Sensor (Diff) | Temperature Sensor (S.E) |
| 11011 | AD27 | Bandgap (Diff)[10] | Bandgap (S.E)[10] |
| 11100 | AD28 | Reserved | Reserved |
| 11101 | AD29 | -VREFH (Diff) | VREFH (S.E) |
| 11110 | AD30 | Reserved | VREFL |
| 11111 | AD31 | Module Disabled | Module Disabled |

1. Interleaved with ADC1_DP3 and ADC1_DM3
2. Interleaved with ADC1_DP3
3. Interleaved with ADC0_DP2 and ADC1_SE6a
4. Interleaved with ADC1_DP0 and ADC1_DM0
5. Interleaved with ADC1_DP0
6. ADCx_CFG2[MUXSEL] bit selects between ADCx_SEn channels a and b. Refer to MUXSEL description in ADC chapter for details.
7. Interleaved with ADC1_SE8
8. Interleaved with ADC1_SE9
9. Interleaved with ADC1_DM3
10. This is the PMC bandgap 1V reference voltage not the VREF module 1.2 V reference voltage. Prior to reading from this ADC channel, ensure that you enable the bandgap buffer by setting the PMC_REGSC[BGBE] bit. Refer to the device data sheet for the bandgap voltage ($V_{BG}$) specification.

### 3.7.1.4 ADC1 Connections/Channel Assignment

**NOTE**

As indicated in the following tables, each ADCx_DPx input and certain ADCx_DMx inputs may operate as single-ended ADC channels in single-ended mode.

### 3.7.1.4.1 ADC1 Channel Assignment for 144-Pin Package

| ADC Channel (SC1n[ADCH]) | Channel | Input signal (SC1n[DIFF]= 1) | Input signal (SC1n[DIFF]= 0) |
|---|---|---|---|
| 00000 | DAD0 | ADC1_DP0 and ADC1_DM0[1] | ADC1_DP0[2] |
| 00001 | DAD1 | ADC1_DP1 and ADC1_DM1 | ADC1_DP1 |
| 00010 | DAD2 | Reserved | Reserved |
| 00011 | DAD3 | ADC1_DP3 and ADC1_DM3[3] | ADC1_DP3[4] |
| 00100[5] | AD4a | Reserved | ADC1_SE4a |
| 00101[5] | AD5a | Reserved | ADC1_SE5a |
| 00110[5] | AD6a | Reserved | ADC1_SE6a |
| 00111[5] | AD7a | Reserved | ADC1_SE7a |
| 00100[5] | AD4b | Reserved | ADC1_SE4b |
| 00101[5] | AD5b | Reserved | ADC1_SE5b |
| 00110[5] | AD6b | Reserved | ADC1_SE6b |
| 00111[5] | AD7b | Reserved | ADC1_SE7b |
| 01000 | AD8 | Reserved | ADC1_SE8[6] |
| 01001 | AD9 | Reserved | ADC1_SE9[7] |
| 01010 | AD10 | Reserved | ADC1_SE10 |
| 01011 | AD11 | Reserved | ADC1_SE11 |
| 01100 | AD12 | Reserved | ADC1_SE12 |
| 01101 | AD13 | Reserved | ADC1_SE13 |
| 01110 | AD14 | Reserved | ADC1_SE14 |
| 01111 | AD15 | Reserved | ADC1_SE15 |
| 10000 | AD16 | Reserved | ADC1_SE16 |
| 10001 | AD17 | Reserved | ADC1_SE17 |
| 10010 | AD18 | Reserved | VREF Output |
| 10011 | AD19 | Reserved | ADC1_DM0[8] |
| 10100 | AD20 | Reserved | ADC1_DM1 |
| 10101 | AD21 | Reserved | Reserved |
| 10110 | AD22 | Reserved | Reserved |
| 10111 | AD23 | Reserved | 12-bit DAC1 Output/ ADC1_SE23 |
| 11000 | AD24 | Reserved | Reserved |
| 11001 | AD25 | Reserved | Reserved |
| 11010 | AD26 | Temperature Sensor (Diff) | Temperature Sensor (S.E) |
| 11011 | AD27 | Bandgap (Diff)[9] | Bandgap (S.E)[9] |
| 11100 | AD28 | Reserved | Reserved |
| 11101 | AD29 | -VREFH (Diff) | VREFH (S.E) |
| 11110 | AD30 | Reserved | VREFL |
| 11111 | AD31 | Module Disabled | Module Disabled |

1. Interleaved with ADC0_DP3 and ADC0_DM3
2. Interleaved with ADC0_DP3

3. Interleaved with ADC0_DP0 and ADC0_DM0
4. Interleaved with ADC0_DP0
5. ADCx_CFG2[MUXSEL] bit selects between ADCx_SEn channels a and b. Refer to MUXSEL description in ADC chapter for details.
6. Interleaved with ADC0_SE8
7. Interleaved with ADC0_SE9
8. Interleaved with ADC0_DM3
9. This is the PMC bandgap 1V reference voltage not the VREF module 1.2 V reference voltage. Prior to reading from this ADC channel, ensure that you enable the bandgap buffer by setting the PMC_REGSC[BGBE] bit. Refer to the device data sheet for the bandgap voltage ($V_{BG}$) specification.

### 3.7.1.5 ADC Channels MUX Selection

The following figure shows the assignment of ADCx_SEn channels a and b through a MUX selection to ADC. To select between alternate set of channels, refer to ADCx_CFG2[MUXSEL] bit settings for more details.

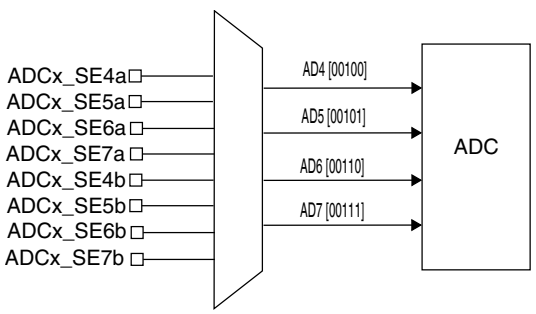



**Figure 3-36. ADCx_SEn channels a and b selection**

### 3.7.1.6 ADC Hardware Interleaved Channels

The AD8 and AD9 channels on ADCx are interleaved in hardware using the following configuration.

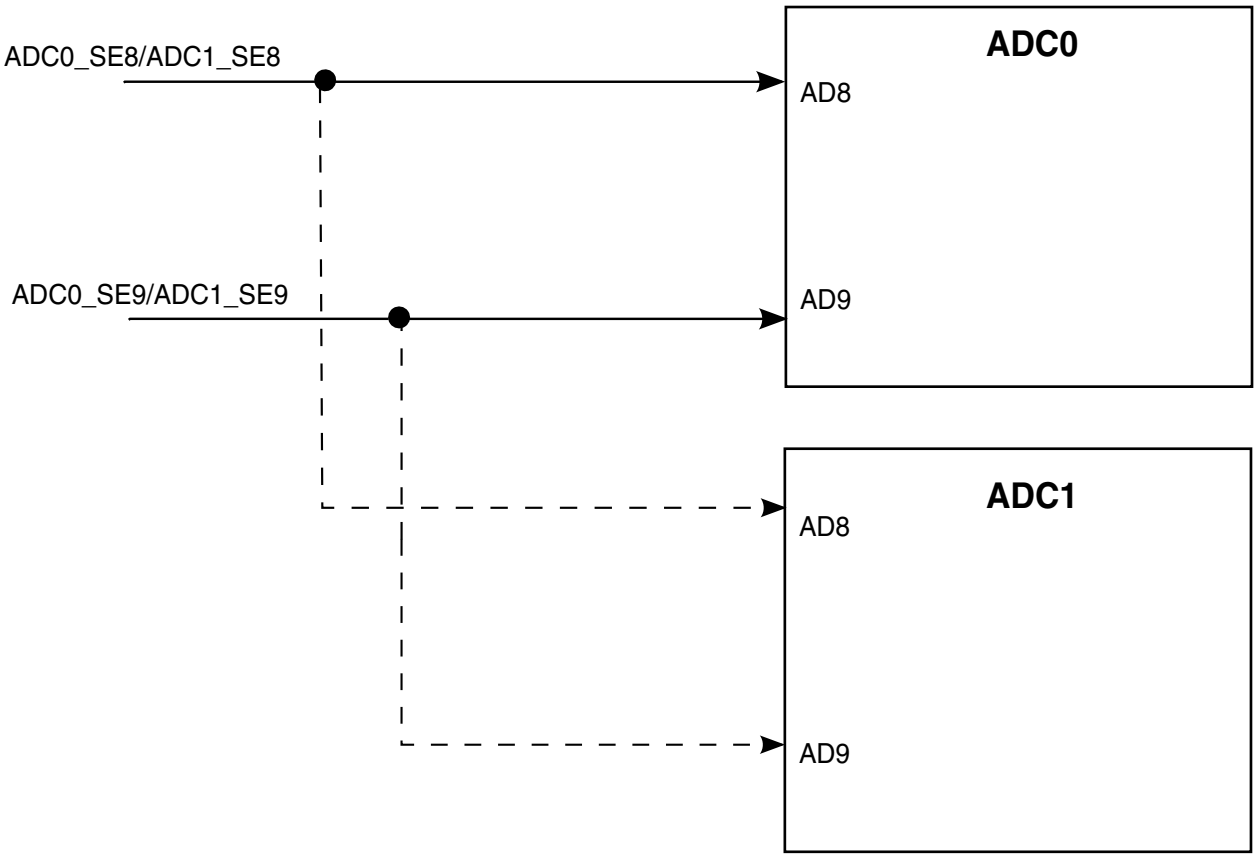



**Figure 3-37. ADC hardware interleaved channels integration**

### 3.7.1.7 ADC Reference Options

The ADC supports the following references:

- VREFH/VREFL - connected as the primary reference option
- 1.2 V VREF_OUT - connected as the $V_{ALT}$ reference option

ADCx_SC2[REFSEL] bit selects the voltage reference sources for ADC. Refer to REFSEL description in ADC chapter for more details.

### 3.7.1.8 ADC triggers

The ADC supports both software and hardware triggers. The primary hardware mechanism for triggering the ADC is the PDB. The PDB itself can be triggered by other peripherals. For example: RTC (Alarm, Seconds) signal is connected to the PDB. The PDB trigger can receive the RTC (alarm/seconds) trigger input forcing ADC conversions in run mode (where PDB is enabled). On the other hand, the ADC can conduct conversions in low power modes, not triggered by PDB. This allows the ADC to do

conversions in low power mode and store the output in the result register. The ADC generates interrupt when the data is ready in the result register that wakes the system from low power mode. The PDB can also be bypassed by using the ADCxTRGSEL bits in the SOPT7 register.

For operation of triggers in different modes, refer to Power Management chapter.

#### 3.7.1.9 Alternate clock

For this device, the alternate clock is connected to OSCERCLK.

**NOTE**

This clock option is only usable when OSCERCLK is in the MHz range. A system with OSCERCLK in the kHz range has the optional clock source below minimum ADC clock operating frequency.

#### 3.7.1.10 ADC low-power modes

This table shows the ADC low-power modes and the corresponding chip low-power modes.

**Table 3-46. ADC low-power modes**

| Module mode | Chip mode |
|---|---|
| Wait | Wait, VLPW |
| Normal Stop | Stop, VLPS |
| Low Power Stop | LLS, VLLS3, VLLS2, VLLS1, VLLS0 |

### 3.7.2 CMP Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

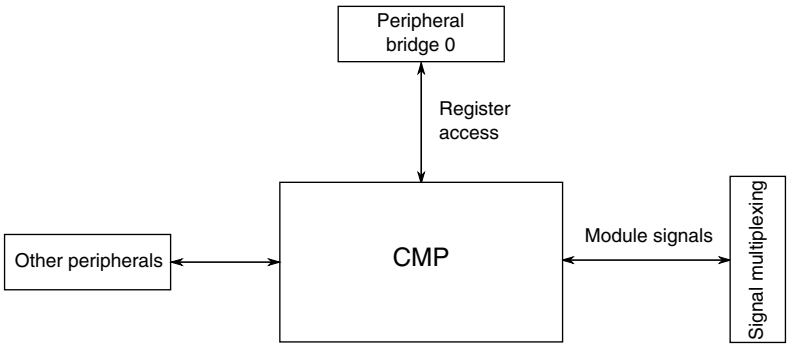



**Figure 3-38. CMP configuration**

**Table 3-47. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | Comparator (CMP) | Comparator |
| System memory map | | System memory map |
| Clocking | | Clock distribution |
| Power management | | Power management |
| Signal multiplexing | Port control | Signal multiplexing |

### 3.7.2.1 CMP input connections

The following table shows the fixed internal connections to the CMP.

| CMP Inputs | CMP0 | CMP1 | CMP2 |
|---|---|---|---|
| IN0 | CMP0_IN0 | CMP1_IN0 | CMP2_IN0 |
| IN1 | CMP0_IN1 | CMP1_IN1 | CMP2_IN1 |
| IN2 | CMP0_IN2 | CMP1_IN2 | CMP2_IN2 |
| IN3 | CMP0_IN3 | 12-bit DAC0_OUT/ CMP1_IN3 | 12-bit DAC1_OUT/ CMP2_IN3 |
| IN4 | 12-bit DAC1_OUT/ CMP0_IN4 | — | — |
| IN5 | VREF output/CMP0_IN5 | VREF output/CMP1_IN5 | — |
| IN6 | Bandgap | Bandgap | Bandgap |
| IN7 | 6b DAC0 reference | 6b DAC1 reference | 6b DAC2 reference |

### 3.7.2.2 CMP external references

The 6-bit DAC sub-block supports selection of two references. For this device, the references are connected as follows:

- VREF_OUT - $V_{in1}$ input
- VDD - $V_{in2}$ input

### 3.7.2.3 External window/sample input

Individual PDB pulse-out signals control each CMP Sample/Window timing.

## 3.7.3 12-bit DAC Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

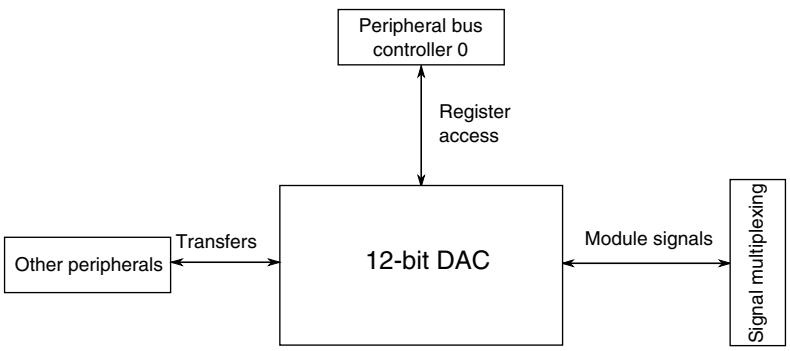



**Figure 3-39. 12-bit DAC configuration**

**Table 3-48. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | 12-bit DAC | 12-bit DAC |
| System memory map | | System memory map |
| Clocking | | Clock distribution |
| Power management | | Power management |
| Signal multiplexing | Port control | Signal multiplexing |

### 3.7.3.1 12-bit DAC Overview

This device contains two 12-bit digital-to-analog converters (DAC) with programmable reference generator output. The DAC includes a FIFO for DMA support.

### 3.7.3.2 12-bit DAC Output

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.

### 3.7.3.3 12-bit DAC Reference

For this device VREF_OUT and VDDA are selectable as the DAC reference. VREF_OUT is connected to the DACREF_1 input and VDDA is connected to the DACREF_2 input. Use DACx_C0[DACRFS] control bit to select between these two options.

Be aware that if the DAC and ADC use the VREF_OUT reference simultaneously, some degradation of ADC accuracy is to be expected due to DAC switching.

### 3.7.3.4 DAC0 Base Addresses

DAC0 can be accessed through both AIPS0 and AIPS1. When accessed through AIPS0, the base address is 4003_F000h and when accessed through AIPS1, the base address is 400C_C000h.

## 3.7.4 VREF Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

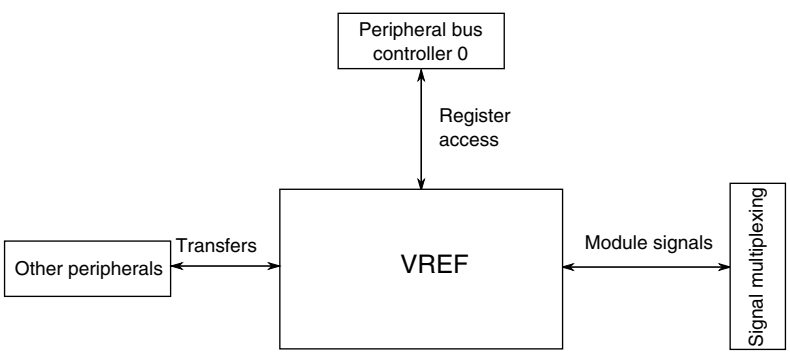



**Figure 3-40. VREF configuration**

**Table 3-49. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | VREF | VREF |

*Table continues on the next page...*

**Table 3-49. Reference links to related information (continued)**

| Topic | Related module | Reference |
|---|---|---|
| System memory map | | System memory map |
| Clocking | | Clock distribution |
| Power management | | Power management |
| Signal multiplexing | Port control | Signal multiplexing |

#### 3.7.4.1 VREF Overview

This device includes a voltage reference (VREF) to supply an accurate 1.2 V voltage output.

The voltage reference can provide a reference voltage to external peripherals or a reference to analog peripherals, such as the ADC, DAC, or CMP.

**NOTE**

PMC_REGSC[BGEN] bit must be set if the VREF regulator is required to remain operating in VLPx modes.

**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.

## 3.8 Timers

### 3.8.1 PDB Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

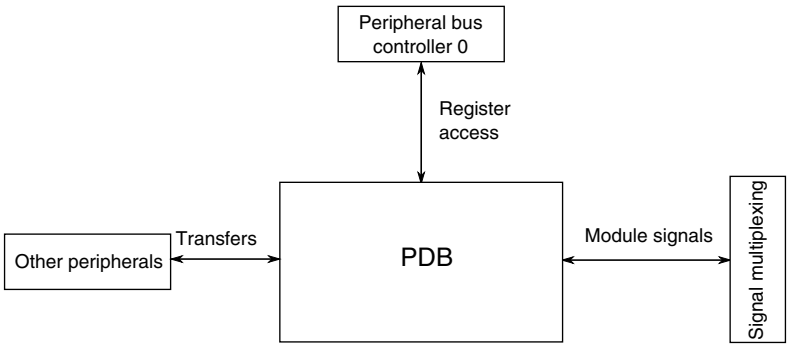



**Figure 3-41. PDB configuration**

**Table 3-50. Reference links to related information**

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

## 3.8.1.1 PDB Instantiation

### 3.8.1.1.1 PDB Output Triggers

**Table 3-51. PDB output triggers**

| | |
|---|---|
| Number of PDB channels for ADC trigger | 2 |
| Number of PDB channels to inter-connect with FTM0 | 1 |
| Number of pre-triggers per PDB channel | 2 |
| Number of DAC triggers | 2 |
| Number of PulseOut | 3 |

### 3.8.1.1.2 PDB Input Trigger Connections

**Table 3-52. PDB Input Trigger Options**

| PDB Trigger | PDB Input |
|---|---|
| 0000 | External Trigger |
| 0001 | CMP 0 |
| 0010 | CMP 1 |
| 0011 | CMP 2 |
| 0100 | PIT Ch 0 Output |

*Table continues on the next page...*

**Table 3-52. PDB Input Trigger Options (continued)**

| PDB Trigger | PDB Input |
|---|---|
| 0101 | PIT Ch 1 Output |
| 0110 | PIT Ch 2 Output |
| 0111 | PIT Ch 3 Output |
| 1000 | FTM0 initialization trigger and channel triggers, as programmed in the FTM external trigger register (EXTTRIG) |
| 1001 | FTM1 initialization trigger and channel triggers, as programmed in the FTM external trigger register (EXTTRIG) |
| 1010 | FTM2 initialization trigger and channel triggers, as programmed in the FTM external trigger register (EXTTRIG) |
| 1011 | FTM3 initialization trigger and channel triggers, as programmed in the FTM external trigger register (EXTTRIG) |
| 1100 | RTC Alarm |
| 1101 | RTC Seconds |
| 1110 | LPTMR Output |
| 1111 | Software Trigger |

### 3.8.1.2 PDB Module Interconnections

| PDB trigger outputs | Connection |
|---|---|
| Channel 0 triggers | ADC0 trigger |
| Channel 1 triggers | ADC1 trigger and synchronous input 1 of FTM0 |
| DAC triggers | DAC0 and DAC1 trigger |
| Pulse-out | Pulse-out connected to each CMP module's sample/window input to control sample operation |

### 3.8.1.3 Back-to-back acknowledgement connections

In this MCU, PDB back-to-back operation acknowledgment connections are implemented as follows:

- PDB channel 0 pre-trigger 0 acknowledgement input: ADC1SC1B_COCO
- PDB channel 0 pre-trigger 1 acknowledgement input: ADC0SC1A_COCO
- PDB channel 1 pre-trigger 0 acknowledgement input: ADC0SC1B_COCO
- PDB channel 1 pre-trigger 1 acknowledgement input: ADC1SC1A_COCO

So, the back-to-back chain is connected as a ring:

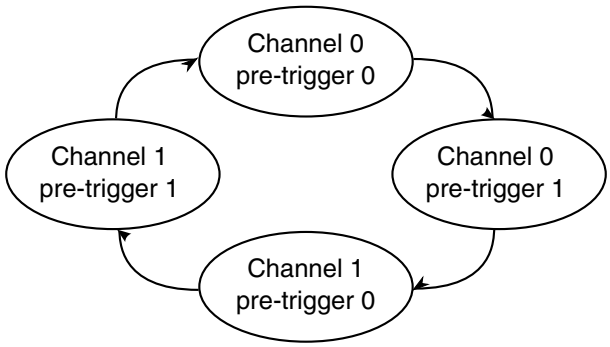



**Figure 3-42. PDB back-to-back chain**

The application code can set the PDB*x*_CH*n*C1[BB] bits to configure the PDB pre-triggers as a single chain or several chains.

### 3.8.1.4 PDB Interval Trigger Connections to DAC

In this MCU, PDB interval trigger connections to DAC are implemented as follows.

- PDB interval trigger 0 connects to DAC0 hardware trigger input.
- PDB interval trigger 1 connects to DAC1 hardware trigger input.

### 3.8.1.5 DAC External Trigger Input Connections

In this MCU, the following DAC external trigger inputs are implemented.

- DAC external trigger input 0: ADC0SC1A_COCO
- DAC external trigger input 1: ADC1SC1A_COCO

**NOTE**

Application code can set the PDBx_DACINTCn[EXT] bit to allow DAC external trigger input when the corresponding ADC Conversion complete flag, ADCx_SC1n[COCO], is set.

### 3.8.1.6 Pulse-Out Connection

Individual PDB Pulse-Out signals are connected to each CMP block and used for sample window.

### 3.8.1.7 Pulse-Out Enable Register Implementation

The following table shows the comparison of pulse-out enable register at the module and chip level.

**Table 3-53. PDB pulse-out enable register**

| Register | Module implementation | Chip implementation |
|---|---|---|
| POnEN | 7:0 - POEN<br>31:8 - Reserved | 0 - POEN[0] for CMP0<br>1 - POEN[1] for CMP1<br>2 - POEN[2] for CMP2<br>31:3 - Reserved |

## 3.8.2 FlexTimer Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

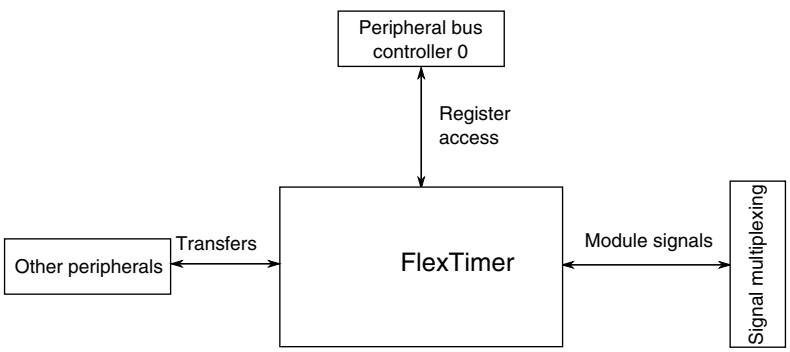



**Figure 3-43. FlexTimer configuration**

**Table 3-54. Reference links to related information**

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

### 3.8.2.1 Instantiation Information

This device contains four FlexTimer modules.

The following table shows how these modules are configured.

**Table 3-55. FTM Instantiations**

| FTM instance | Number of channels | Features/usage |
|---|---|---|
| FTM0 | 8 | 3-phase motor + 2 general purpose or stepper motor |
| FTM1 | 2[1] | Quadrature decoder or general purpose |
| FTM2 | 2[1] | Quadrature decoder or general purpose |
| FTM3 | 8 | 3-phase motor + 2 general purpose or stepper motor |

1. Only channels 0 and 1 are available.

Compared with the FTM0 and FTM3 configuration, the FTM1 and FTM2 configuration adds the Quadrature decoder feature and reduces the number of channels.

### 3.8.2.2 External Clock Options

By default each FTM is clocked by the internal bus clock (the FTM refers to it as system clock). Each module contains a register setting that allows the module to be clocked from an external clock instead. There are two external FTM_CLKINx pins that can be selected by any FTM module via the SOPT4 register in the SIM module.

### 3.8.2.3 Fixed frequency clock

The fixed frequency clock for each FTM is MCGFFCLK.

### 3.8.2.4 FTM Interrupts

The FlexTimer has multiple sources of interrupt. However, these sources are OR'd together to generate a single interrupt request per FTM module to the interrupt controller. When an FTM interrupt occurs, read the FTM status registers (FMS, SC, and STATUS) to determine the exact interrupt source.

### 3.8.2.5 FTM Fault Detection Inputs

The following fault detection input options for the FTM modules are selected via the SOPT4 register in the SIM module. The external pin option is selected by default.

- FTM0 FAULT0 = FTM0_FLT0 pin or CMP0 output

- FTM0 FAULT1 = FTM0_FLT1 pin or CMP1 output
- FTM0 FAULT2 = FTM0_FLT2 pin or CMP2 output
- FTM0 FAULT3 = FTM0_FLT3 pin

- FTM1 FAULT0 = FTM1_FLT0 pin or CMP0 output
- FTM1 FAULT1 = CMP1 output
- FTM1 FAULT2 = CMP2 output

- FTM2 FAULT0 = FTM2_FLT0 pin or CMP0 output
- FTM2 FAULT1 = CMP1 output
- FTM2 FAULT2 = CMP2 output

- FTM3 FAULT0 = FTM3_FLT0 pin or CMP0 output
- FTM3 FAULT1 = CMP2 output

#### 3.8.2.6 FTM Hardware Triggers

The FTM synchronization hardware triggers are connected in the chip as follows:

- FTM0 hardware trigger 0 = CMP0 Output or FTM1 Match (when enabled in the FTM1 External Trigger (EXTTRIG) register)
- FTM0 hardware trigger 1 = PDB channel 1 Trigger Output or FTM2 Match (when enabled in the FTM2 External Trigger (EXTTRIG) register)
- FTM0 hardware trigger 2 = FTM0_FLT0 pin

- FTM1 hardware trigger 0 = CMP0 Output
- FTM1 hardware trigger 1 = CMP1 Output
- FTM1 hardware trigger 2 = FTM1_FLT0 pin

- FTM2 hardware trigger 0 = CMP0 Output
- FTM2 hardware trigger 1 = CMP2 Output
- FTM2 hardware trigger 2 = FTM2_FLT0 pin

- FTM3 hardware trigger 0 = FTM1 Match (when enabled in the FTM1 External Trigger (EXTTRIG) register)
- FTM3 hardware trigger 1 = FTM2 Match (when enabled in the FTM2 External Trigger (EXTTRIG) register)
- FTM3 hardware trigger 2 = FTM3_FLT0 pin

For the triggers with more than one option, the SOPT4 register in the SIM module controls the selection.

### 3.8.2.7 Input capture options for FTM module instances

The following channel 0 input capture source options are selected via the SOPT4 register in the SIM module. The external pin option is selected by default.

- FTM1 channel 0 input capture = FTM1_CH0 pin or CMP0 output or CMP1 output or USB start of frame pulse
- FTM2 channel 0 input capture = FTM2_CH0 pin or CMP0 output or CMP1 output

**NOTE**

When the USB start of frame pulse option is selected as an FTM channel input capture, disable the USB SOF token interrupt in the USB Interrupt Enable register (INTEN[SOFTOKEN]) to avoid USB enumeration conflicts.

### 3.8.2.8 FTM output triggers for other modules

FTM output triggers can be selected as input triggers for the PDB and ADC modules. See PDB Instantiation and ADC triggers.

### 3.8.2.9 FTM Global Time Base

This chip provides the optional FTM global time base feature (see Global time base (GTB)).

FTM0 provides the only source for the FTM global time base. The other FTM modules can share the time base as shown in the following figure:

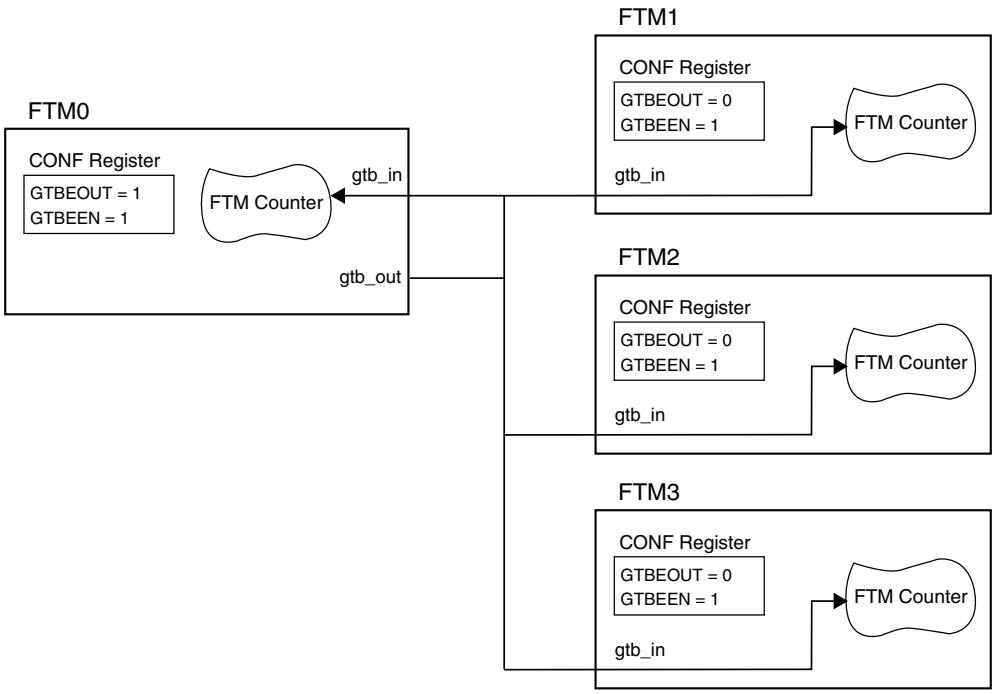



**Figure 3-44. FTM Global Time Base Configuration**

### 3.8.2.10 FTM BDM and debug halt mode

In the FTM chapter, references to the chip being in "BDM" are the same as the chip being in “debug halt mode".

### 3.8.2.11 FTM2 Base Addresses

FTM2 can be accessed through both AIPS0 and AIPS1. When accessed through AIPS0, the base address is 4003_A000h and when accessed through AIPS1, the base address is 400B_8000h.

## 3.8.3 PIT Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module’s dedicated chapter.

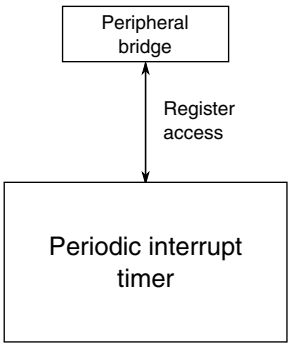



**Figure 3-45. PIT configuration**

**Table 3-56. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | PIT | PIT |
| System memory map | | System memory map |
| Clocking | | Clock Distribution |
| Power management | | Power management |

### 3.8.3.1 PIT/DMA Periodic Trigger Assignments

The PIT generates periodic trigger events to the DMA Mux as shown in the table below.

**Table 3-57. PIT channel assignments for periodic DMA triggering**

| DMA Channel Number | PIT Channel |
|---|---|
| DMA Channel 0 | PIT Channel 0 |
| DMA Channel 1 | PIT Channel 1 |
| DMA Channel 2 | PIT Channel 2 |
| DMA Channel 3 | PIT Channel 3 |

### 3.8.3.2 PIT/ADC Triggers

PIT triggers are selected as ADCx trigger sources using the SOPT7[ADCxTRGSEL] bits in the SIM module. For more details, refer to SIM chapter.

## 3.8.4 Low-power timer configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

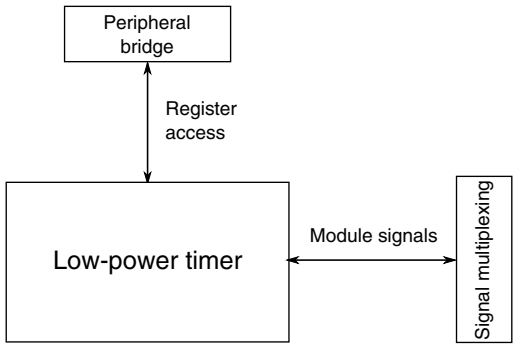



**Figure 3-46. LPTMR configuration**

**Table 3-58. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | Low-power timer | Low-power timer |
| System memory map | | System memory map |
| Clocking | | Clock Distribution |
| Power management | | Power management |
| Signal Multiplexing | Port control | Signal Multiplexing |

### 3.8.4.1 LPTMR prescaler/glitch filter clocking options

The prescaler and glitch filter of the LPTMR module can be clocked from one of four sources determined by the LPTMR0_PSR[PCS] bitfield. The following table shows the chip-specific clock assignments for this bitfield.

**NOTE**

The chosen clock must remain enabled if the LPTMR is to continue operating in all required low-power modes.

| LPTMR0_PSR[PCS] | Prescaler/glitch filter clock number | Chip clock |
|---|---|---|
| 00 | 0 | MCGIRCLK — internal reference clock (not available in VLPS/LLS/VLLS modes) |
| 01 | 1 | LPO — 1 kHz clock (not available in VLLS0 mode) |
| 10 | 2 | ERCLK32K — secondary external reference clock |
| 11 | 3 | OSCERCLK — external reference clock (not available in VLLS0 mode) |

See Clock Distribution for more details on these clocks.

### 3.8.4.2 LPTMR pulse counter input options

The LPTMR_CSR[TPS] bitfield configures the input source used in pulse counter mode. The following table shows the chip-specific input assignments for this bitfield.

| LPTMR_CSR[TPS] | Pulse counter input number | Chip input |
|---|---|---|
| 00 | 0 | CMP0 output |
| 01 | 1 | LPTMR_ALT1 pin |
| 10 | 2 | LPTMR_ALT2 pin |
| 11 | 3 | Reserved |

## 3.8.5 CMT Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

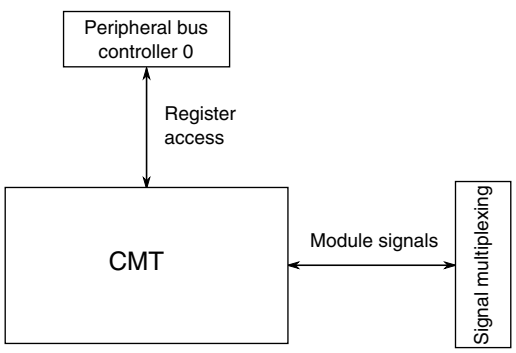



**Figure 3-47. CMT configuration**

**Table 3-59. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | Carrier modulator transmitter (CMT) | CMT |
| System memory map | | System memory map |
| Clocking | | Clock distribution |
| Power management | | Power management |
| Signal multiplexing | Port control | Signal multiplexing |

### 3.8.5.1 Instantiation Information

This device contains one CMT module.

### 3.8.5.2 IRO Drive Strength

The IRO pad requires higher current drive than can be obtained from a single pad. For this device, the pin associated with the CMT_IRO signal is doubled bonded to two pads.

The SOPT2[PTD7PAD] field in SIM module can be used to configure the pin associated with the CMT_IRO signal as a higher current output port pin.

## 3.8.6 RTC configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

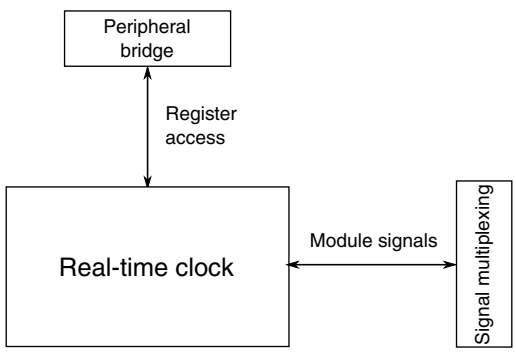



**Figure 3-48. RTC configuration**

**Table 3-60. 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 |

### 3.8.6.1 RTC_CLKOUT signal

When the RTC is enabled and the port control module selects the RTC_CLKOUT function, the RTC_CLKOUT signal outputs a 1 Hz or 32 kHz output derived from RTC oscillator as shown below.

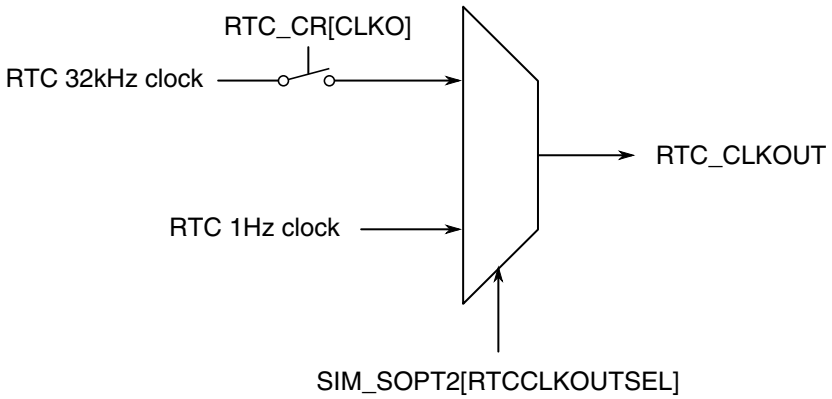



**Figure 3-49. RTC_CLKOUT generation**

# 3.9 Communication interfaces

## 3.9.1 Universal Serial Bus (USB) FS Subsystem

The USB FS subsystem includes these components:

- Dual-role USB OTG-capable (On-The-Go) controller that supports a full-speed (FS) device or FS/LS host. The module complies with the USB 2.0 specification.
- USB transceiver that includes internal 15 kΩ pulldowns on the D+ and D- lines for host mode functionality.
- A 3.3 V regulator.
- USB device charger detection module.
- VBUS detect signal: To detect a valid VBUS in device mode, use a GPIO signal that can wake the chip in all power modes.

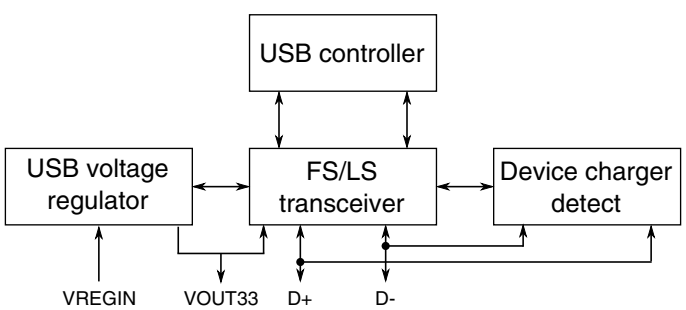



**Figure 3-50. USB Subsystem Overview**

**NOTE**

Use the following code sequence to select USB clock source, USB clock divide ratio, and enable its clock gate to avoid

potential clock glitches which may result in USB enumeration stage failure.

1. Select the USB clock source by configuring SIM_SOPT2.
2. Select the desired clock divide ratio by configuring SIM_CLKDIV2.
3. Enable USB clock gate by setting SIM_SCGC4.

#### 3.9.1.1 USB Wakeup

When the USB detects that there is no activity on the USB bus for more than 3 ms, the INT_STAT[SLEEP] bit is set. This bit can cause an interrupt and software decides the appropriate action.

Waking from a low power mode (except in LLS/VLLS mode where USB is not powered) occurs through an asynchronous interrupt triggered by activity on the USB bus. Setting the USBTRC0[USBRESMEN] bit enables this function.

#### 3.9.1.2 USB Power Distribution

This chip includes an internal 5 V to 3.3 V USB regulator that powers the USB transceiver or the MCU (depending on the application).

##### 3.9.1.2.1 AA/AAA cells power supply

The chip can be powered by two AA/AAA cells. In this case, the MCU is powered through VDD which is within the 1.8 to 3.0 V range. After USB cable insertion is detected, the USB regulator is enabled to power the USB transceiver.

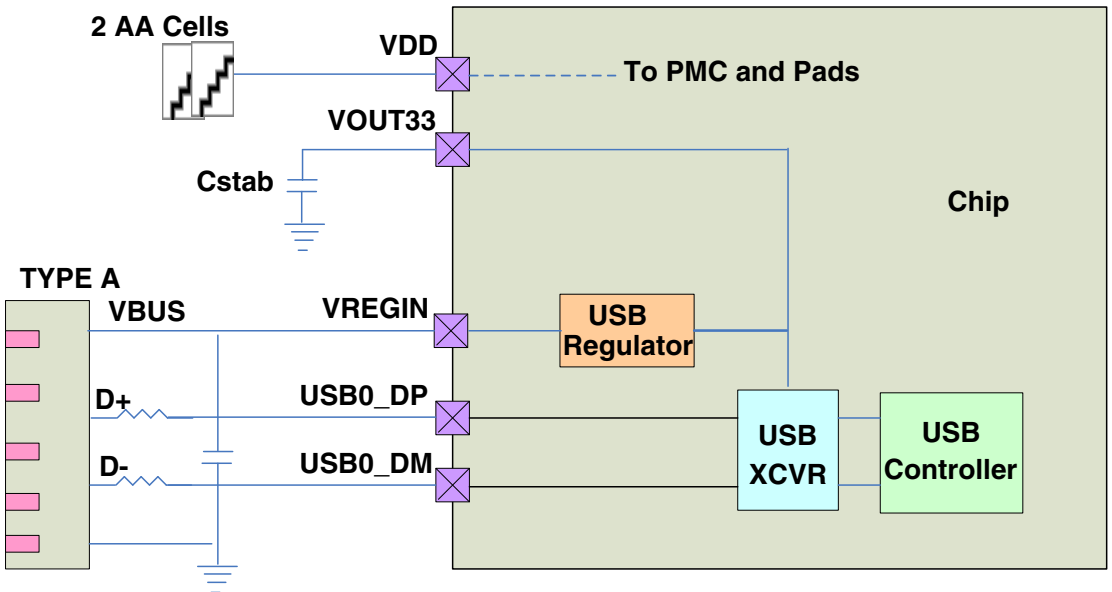



**Figure 3-51. USB regulator AA cell usecase**

### 3.9.1.2.2 Li-Ion battery power supply

The chip can also be powered by a single Li-ion battery. In this case, VOUT33 is connected to VDD. The USB regulator must be enabled by default to power the MCU. When connected to a USB host, the input source of this regulator is switched to the USB bus supply from the Li-ion battery. To charge the battery, the MCU can configure the battery charger according to the charger detection information.

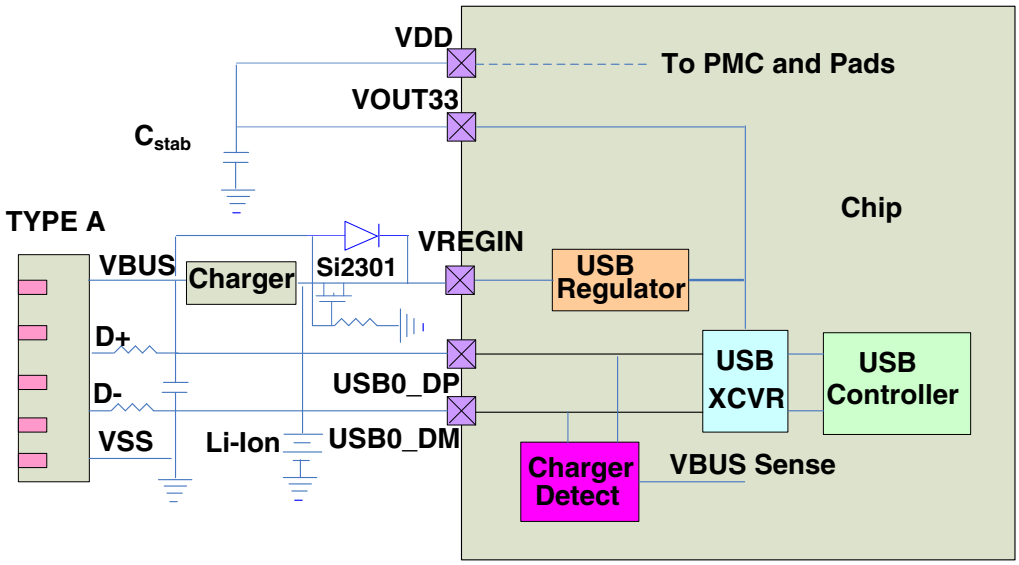



**Figure 3-52. USB regulator Li-ion usecase**

### 3.9.1.2.3 USB bus power supply

The chip can also be powered by the USB bus directly. In this case, VOUT33 is connected to VDD. The USB regulator must be enabled by default to power the MCU, then to power USB transceiver or external sensor.

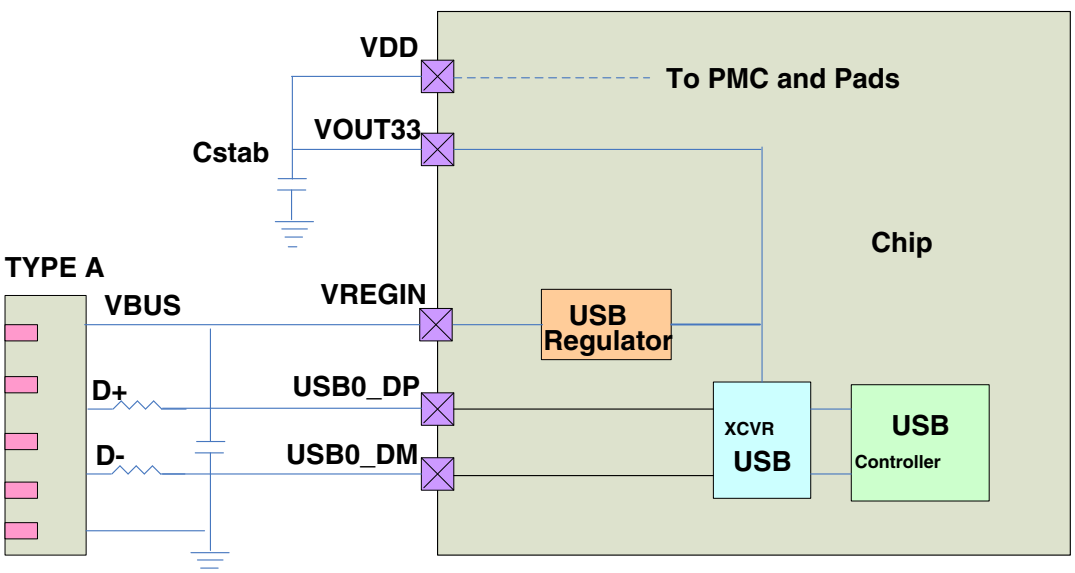



**Figure 3-53. USB regulator bus supply**

## 3.9.1.3 USB power management

The regulator should be put into STANDBY mode whenever the chip is in Stop mode.

### 3.9.1.4 USB controller configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

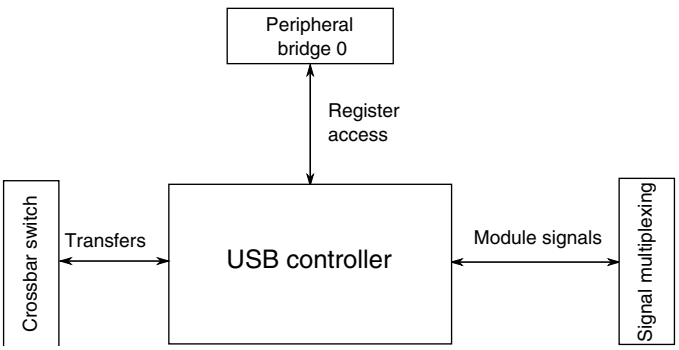



**Figure 3-54. USB controller configuration**

**Table 3-61. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | USB controller | USB controller |
| System memory map | | System memory map |
| Clocking | | Clock Distribution |
| Transfers | Crossbar switch | Crossbar switch |
| Signal Multiplexing | Port control | Signal Multiplexing |

**NOTE**

When USB is not used in the application, it is recommended that the USB regulator VREGIN and VOUT33 pins remain floating.

### 3.9.1.5 USB DCD Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

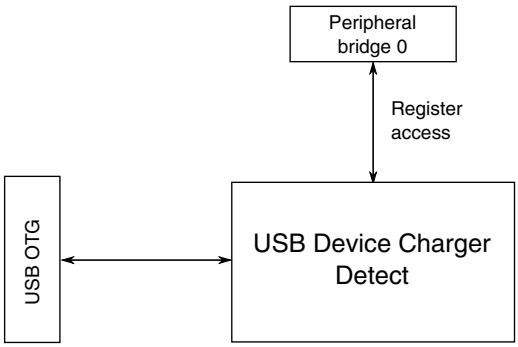



**Figure 3-55. USB DCD configuration**

**Table 3-62. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | USB DCD | USB DCD |
| System memory map | | System memory map |
| Clocking | | Clock Distribution |
| | USB FS/LS controller | USB FS/LS controller |

### 3.9.1.6 USB Voltage Regulator Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

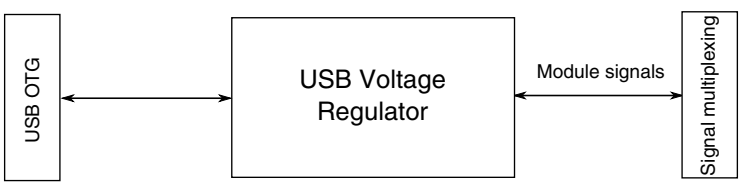



**Figure 3-56. USB Voltage Regulator configuration**

**Table 3-63. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | USB Voltage Regulator | USB Voltage Regulator |
| System memory map | | System memory map |
| Clocking | | Clock Distribution |
| | USB controller | USB controller |
| Signal Multiplexing | Port control | Signal Multiplexing |

**NOTE**

When USB is not used in the application, it is recommended that the USB regulator VREGIN and VOUT33 pins remain floating.

## 3.9.2 CAN Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

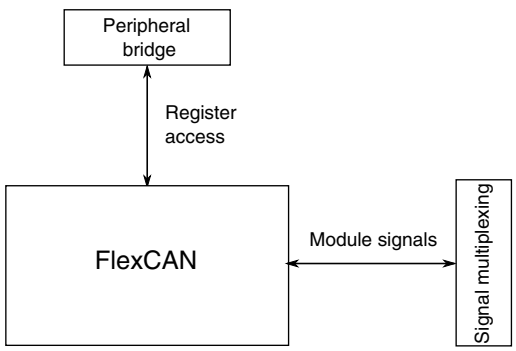



**Figure 3-57. CAN configuration**

**Table 3-64. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | CAN | CAN |
| System memory map | | System memory map |
| Clocking | | Clock Distribution |
| Power management | | Power management |
| Signal Multiplexing | Port control | Signal Multiplexing |

### 3.9.2.1 Reset value of MDIS bit

The CAN_MCR[MDIS] bit is set after reset. Therefore, FlexCAN module is disabled following a reset.

### 3.9.2.2 Number of message buffers

Each FlexCAN module contains 16 message buffers. Each message buffer is 16 bytes.

### 3.9.2.3 FlexCAN Clocking

#### 3.9.2.3.1 Clocking Options

CAN_CTRL1[CLKSRC] register bit selects between clocking the FlexCAN from the internal bus clock or the input clock (OSCERCLK).

#### 3.9.2.3.2 Clock Gating

The clock to each CAN module can be gated on and off using the SCGC*n*[CAN*x*] bits. These bits are cleared after any reset, which disables the clock to the corresponding module. The appropriate clock enable bit should be set by software at the beginning of the FlexCAN initialization routine to enable the module clock before attempting to initialize any of the FlexCAN registers.

### 3.9.2.4 FlexCAN Interrupts

The FlexCAN has multiple sources of interrupt requests. However, some of these sources are OR'd together to generate a single interrupt request. See below for the mapping of the individual interrupt sources to the interrupt request:

| Request | Sources |
|---|---|
| Message buffer | Message buffers 0-15 |
| Bus off | Bus off |
| Error | • Bit1 error<br>• Bit0 error<br>• Acknowledge error<br>• Cyclic redundancy check (CRC) error<br>• Form error<br>• Stuffing error<br>• Transmit error warning<br>• Receive error warning |
| Transmit Warning | Transmit Warning |
| Receive Warning | Receive Warning |
| Wake-up | Wake-up |

### 3.9.2.5 FlexCAN Operation in Low Power Modes

The FlexCAN module is operational in VLPR and VLPW modes. With the 2 MHz bus clock, the fastest supported FlexCAN transfer rate is 256 kbps. With the 4 MHz bus clock, the fastest supported FlexCAN transfer rate is 512 kbps.The bit timing parameters in the module must be adjusted for the new frequency, but full functionality is possible.

The FlexCAN module can be configured to generate a wakeup interrupt in STOP and VLPS modes. When the FlexCAN is configured to generate a wakeup, a recessive to dominant transition on the CAN bus generates an interrupt.

### 3.9.2.6 FlexCAN Doze Mode

The Doze mode for the FlexCAN module is the same as the Wait and VLPW modes for the chip.

## 3.9.3 SPI configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

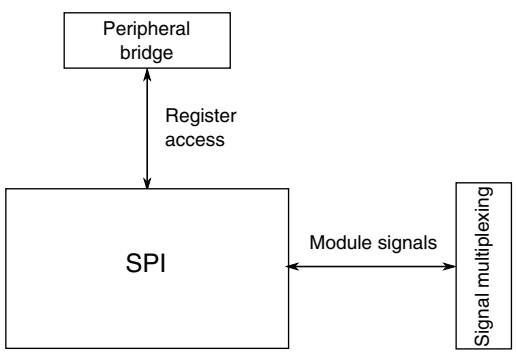



**Figure 3-58. SPI configuration**

**Table 3-65. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | SPI | SPI |
| System memory map | | System memory map |
| Clocking | | Clock Distribution |
| Signal Multiplexing | Port control | Signal Multiplexing |

### 3.9.3.1 SPI Modules Configuration

This device contains three SPI modules.

### 3.9.3.2 SPI clocking

The SPI module is clocked by the internal bus clock (the DSPI refers to it as system clock). The module has an internal divider, with a minimum divide is two. So, the SPI can run at a maximum frequency of bus clock/2.

### 3.9.3.3 Number of CTARs

SPI CTAR registers define different transfer attribute configurations. The SPI module supports up to eight CTAR registers. This device supports two CTARs on all instances of the SPI.

In master mode, the CTAR registers define combinations of transfer attributes, such as frame size, clock phase, clock polarity, data bit ordering, baud rate, and various delays. In slave mode only CTAR0 is used, and a subset of its bitfields sets the slave transfer attributes.

### 3.9.3.4 TX FIFO size

**Table 3-66. SPI transmit FIFO size**

| SPI Module | Transmit FIFO size |
|---|---|
| SPI0 | 4 |
| SPI1 | 1 |
| SPI2 | 1 |

### 3.9.3.5 RX FIFO Size

SPI supports up to 16-bit frame size during reception.

**Table 3-67. SPI receive FIFO size**

| SPI Module | Receive FIFO size |
|---|---|
| SPI0 | 4 |
| SPI1 | 1 |
| SPI2 | 1 |

### 3.9.3.6 Number of PCS signals

The following table shows the number of peripheral chip select signals available per SPI module.

**Table 3-68. SPI PCS signals**

| SPI Module | PCS Signals |
|---|---|
| SPI0 | SPI_PCS[5:0] |
| SPI1 | SPI_PCS[3:0] |
| SPI2 | SPI_PCS[1:0] |

### 3.9.3.7 SPI Operation in Low Power Modes

In VLPR and VLPW modes the SPI is functional; however, the reduced system frequency also reduces the max frequency of operation for the SPI. In VLPR and VLPW modes the max SPI_CLK frequency is 2MHz.

In stop and VLPS modes, the clocks to the SPI module are disabled. The module is not functional, but it is powered so that it retains state.

There is one way to wake from stop mode via the SPI, which is explained in the following section.

#### 3.9.3.7.1 Using GPIO Interrupt to Wake from stop mode

Here are the steps to use a GPIO to create a wakeup upon reception of SPI data in slave mode:

1. Point the GPIO interrupt vector to the desired interrupt handler.
2. Enable the GPIO input to generate an interrupt on either the rising or falling edge (depending on the polarity of the chip select signal).
3. Enter Stop or VLPS mode and Wait for the GPIO interrupt.

**NOTE**

It is likely that in using this approach the first word of data from the SPI host might not be received correctly. This is dependent on the transfer rate used for the SPI, the delay between chip select assertion and presentation of data, and the system interrupt latency.

#### 3.9.3.8 SPI Doze Mode

The Doze mode for the SPI module is the same as the Wait and VLPW modes for the chip.

#### 3.9.3.9 SPI Interrupts

The SPI has multiple sources of interrupt requests. However, these sources are OR'd together to generate a single interrupt request per SPI module to the interrupt controller. When an SPI interrupt occurs, read the SPI_SR to determine the exact interrupt source.

#### 3.9.3.10 SPI clocks

This table shows the SPI module clocks and the corresponding chip clocks.

**Table 3-69. SPI clock connections**

| Module clock | Chip clock |
|---|---|
| System Clock | Bus Clock |

#### 3.9.3.11 Writing SPI Transmit FIFO

In Master mode, only 32-bit write accesses are supported for the PUSHR register, the top TX FIFO entry. In Slave mode, 8-bit, 16-bit and 32-bit write accesses are supported for the PUSHR register, although each write increments the write pointer.

### 3.9.4 I2C Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

**NOTE**

The Open Drain mode on SDA, SCL pins has to be enabled by setting PORTx_PCRn[ODE] bits.

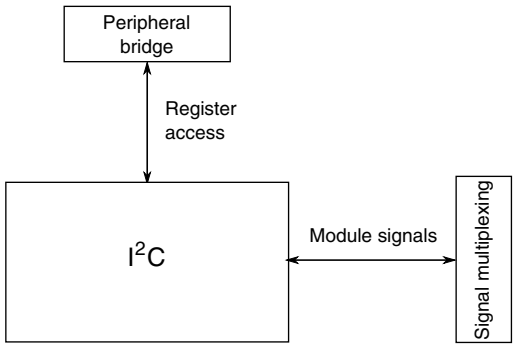



**Figure 3-59. I2C configuration**

**Table 3-70. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | I2C | I2C |
| System memory map | | System memory map |
| Clocking | | Clock Distribution |
| Power management | | Power management |
| Signal Multiplexing | Port control | Signal Multiplexing |

### 3.9.4.1 Number of I2C modules

This device has three $I^2C$ modules.

## 3.9.5 UART Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

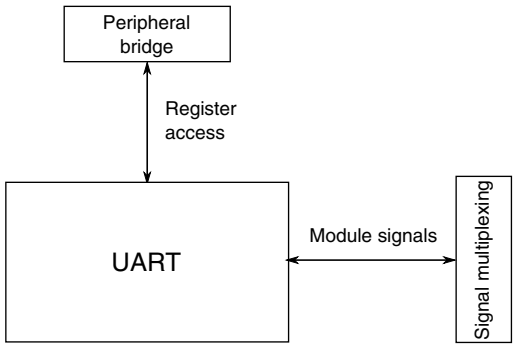



**Figure 3-60. UART configuration**

**Table 3-71. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | UART | UART |
| System memory map | | System memory map |
| Clocking | | Clock Distribution |
| Power management | | Power management |
| Signal Multiplexing | Port control | Signal Multiplexing |

### 3.9.5.1 UART configuration information

This device contains six UART modules. This section describes how each module is configured on this device.

1. Standard features of all UARTs:
   - RS-485 support
   - Hardware flow control (RTS/CTS)
   - 9-bit UART to support address mark with parity
   - MSB/LSB configuration on data
2. UART0 and UART1 are clocked from the core clock, the remaining UARTs are clocked on the bus clock. The maximum baud rate is 1/16 of related source clock frequency.
3. IrDA is available on all UARTs
4. UART0 contains the standard features plus ISO7816
5. AMR support on all UARTs. The pin control and interrupts (PORT) module supports open-drain for all I/O.
6. UART0 and UART1 contain 8-entry transmit and 8-entry receive FIFOs
7. All other UARTs contain a 1-entry transmit and receive FIFOs

### 3.9.5.2 UART wakeup

The UART can be configured to generate an interrupt/wakeup on the first active edge that it receives.

### 3.9.5.3 UART interrupts

The UART has multiple sources of interrupt requests. However, some of these sources are OR'd together to generate a single interrupt request. See below for the mapping of the individual interrupt sources to the interrupt request:

The status interrupt combines the following interrupt sources:

| Source | UART 0 | UART 1 | UART 2 | UART 3 | UART 4 | UART 5 |
|---|---|---|---|---|---|---|
| Transmit data empty | x | x | x | x | x | x |
| Transmit complete | x | x | x | x | x | x |
| Idle line | x | x | x | x | x | x |
| Receive data full | x | x | x | x | x | x |
| LIN break detect | x | x | x | x | x | x |
| RxD pin active edge | x | x | x | x | x | x |
| Initial character detect | x | — | — | — | — | — |

The error interrupt combines the following interrupt sources:

| Source | UART 0 | UART 1 | UART 2 | UART 3 | UART 4 | UART 5 |
|---|---|---|---|---|---|---|
| Receiver overrun | x | x | x | x | x | x |
| Noise flag | x | x | x | x | x | x |
| Framing error | x | x | x | x | x | x |
| Parity error | x | x | x | x | x | x |
| Transmitter buffer overflow | x | x | x | x | x | x |
| Receiver buffer overflow | x | x | x | x | x | x |
| Receiver buffer underflow | x | x | x | x | x | x |
| Transmit threshold (ISO7816) | x | — | — | — | — | — |

*Table continues on the next page...*

| Source | UART 0 | UART 1 | UART 2 | UART 3 | UART 4 | UART 5 |
|---|---|---|---|---|---|---|
| Receiver threshold (ISO7816) | x | — | — | — | — | — |
| Wait timer (ISO7816) | x | — | — | — | — | — |
| Character wait timer (ISO7816) | x | — | — | — | — | — |
| Block wait timer (ISO7816) | x | — | — | — | — | — |
| Guard time violation (ISO7816) | x | — | — | — | — | — |

## 3.9.6 SDHC Configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

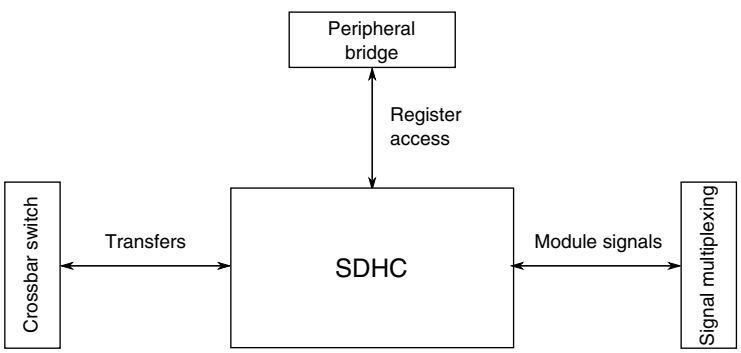



**Figure 3-61. SDHC configuration**

**Table 3-72. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | SDHC | SDHC |
| System memory map | | System memory map |
| Clocking | | Clock Distribution |
| Power management | | Power management |
| Transfers | Crossbar switch | Crossbar switch |
| Signal Multiplexing | Port control | Signal Multiplexing |

### 3.9.6.1 SDHC clocking

In addition to the system clock, the SDHC needs a clock for the base for the external card clock. There are four possible clock sources for this clock, selected by the SIM's SOPT2 register:

- Core/system clock
- MCGPLLCLK or MCGFLLCLK
- OSCERCLK
- Bypass clock from off-chip (SDHC0_CLKIN)

**NOTE**

SDHC0_CLKIN is not available on all packages. See the Pinout section for more information.

### 3.9.6.2 SD bus pullup/pulldown constraints

The SD standard requires the SD bus signals (except the SD clock) to be pulled up during data transfers. The SDHC also provides a feature of detecting card insertion/removal, by detecting voltage level changes on DAT[3] of the SD bus. To support this DAT[3] must be pulled down. To avoid a situation where the SDHC detects voltage changes due to normal data transfers on the SD bus as card insertion/removal, the interrupt relating to this event must be disabled after the card has been inserted and detected. It can be re-enabled after the card is removed.

## 3.9.7 $I^2S$ configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

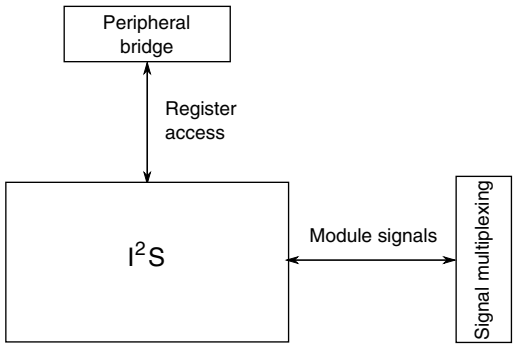



**Figure 3-62. I²S configuration**

**Table 3-73. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | I²S | I2S |
| System memory map | | System memory map |
| Clocking | | Clock Distribution |
| Power management | | Power management |
| Signal multiplexing | Port control | Signal Multiplexing |

### 3.9.7.1 Instantiation information

This device contains one I²S module.

As configured on the device, module features include:

- TX data lines: 2
- RX data lines: 2
- FIFO size (words): 8
- Maximum words per frame: 32
- Maximum bit clock divider: 512

### 3.9.7.2 I²S/SAI clocking

#### 3.9.7.2.1 Audio Master Clock

The audio master clock (MCLK) is used to generate the bit clock when the receiver or transmitter is configured for an internally generated bit clock. The audio master clock can also be output to or input from a pin. The transmitter and receiver have the same audio master clock inputs.

#### 3.9.7.2.2 Bit Clock

The I²S/SAI transmitter and receiver support asynchronous bit clocks (BCLKs) that can be generated internally from the audio master clock or supplied externally. The module also supports the option for synchronous operation between the receiver and transmitter product.

#### 3.9.7.2.3 Bus Clock

The bus clock is used by the control registers and to generate synchronous interrupts and DMA requests.

#### 3.9.7.2.4 I²S/SAI clock generation

Each SAI peripheral can control the input clock selection, pin direction and divide ratio of one audio master clock.

The MCLK Input Clock Select bit of the MCLK Control Register (MCR[MICS]) selects the clock input to the I²S/SAI module's MCLK divider.

The module's MCLK Divide Register (MDR) configures the MCLK divide ratio.

The module's MCLK Output Enable bit of the MCLK Control Register (MCR[MOE]) controls the direction of the MCLK pin. The pin is the input from the pin when MOE is 0, and the pin is the output from the clock divider when MOE is 1.

The transmitter and receiver can independently select between the bus clock and the audio master clock to generate the bit clock. Each module's Clocking Mode field of the Transmit Configuration 2 Register and Receive Configuration 2 Register (TCR2[MSEL] and RCR2[MSEL]) selects the master clock.

#### 3.9.7.2.5 Clock gating and I²S/SAI initialization

The clock to the I²S/SAI module can be gated using a bit in the SIM. To minimize power consumption, these bits are cleared after any reset, which disables the clock to the corresponding module. The clock enable bit should be set by software at the beginning of the module initialization routine to enable the module clock before initialization of any of the I²S/SAI registers.

### 3.9.7.3 I²S/SAI operation in low power modes

#### 3.9.7.3.1 Stop and very low power modes

In Stop mode, the SAI transmitter and/or receiver can continue operating provided the appropriate Stop Enable bit is set (TCSR[STOPE] and/or RCSR[STOPE], respectively), and provided the transmitter and/or receiver is/are using an externally generated bit clock or an Audio Master Clock that remains operating in Stop mode. The SAI transmitter and/or receiver can generate an asynchronous interrupt to wake the CPU from Stop mode.

In VLPS mode, the module behaves as it does in stop mode if VLPS mode is entered from run mode. However, if VLPS mode is entered from VLPR mode, the FIFO might underflow or overflow before wakeup from stop mode due to the limits in bus bandwidth. In VLPW and VLPR modes, the module is limited by the maximum bus clock frequencies.

When operating from an internally generated bit clock or Audio Master Clock that is disabled in stop modes:

In Stop mode, if the Transmitter Stop Enable (TCSR[STOPE]) bit is clear, the transmitter is disabled after completing the current transmit frame, and, if the Receiver Stop Enable (RCSR[STOPE]) bit is clear, the receiver is disabled after completing the current receive frame. Entry into Stop mode is prevented–not acknowledged–while waiting for the transmitter and receiver to be disabled at the end of the current frame.

#### 3.9.7.3.2 Low-leakage modes

When entering low-leakage modes, the Stop Enable (TCSR[STOPE] and RCSR[STOPE]) bits are ignored and the SAI is disabled after completing the current transmit and receive Frames. Entry into stop mode is prevented (not acknowledged) while waiting for the transmitter and receiver to be disabled at the end of the current frame.

## 3.10 Human-machine interfaces

### 3.10.1 GPIO configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

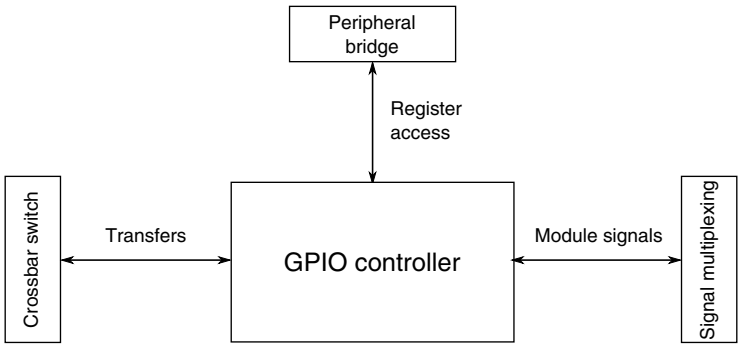



**Figure 3-63. GPIO configuration**

**Table 3-74. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | GPIO | GPIO |
| System memory map | | System memory map |
| Clocking | | Clock Distribution |
| Power management | | Power management |
| Transfers | Crossbar switch | Clock Distribution |
| Signal Multiplexing | Port control | Signal Multiplexing |

### 3.10.1.1 GPIO access protection

The GPIO module does not have access protection because it is not connected to a peripheral bridge slot and is not protected by the MPU.

### 3.10.1.2 Number of GPIO signals

The number of GPIO signals available on the devices covered by this document are detailed in Orderable part numbers .

# Chapter 4
# Memory Map

## 4.1 Introduction

This device contains various memories and memory-mapped peripherals which are located in one 32-bit contiguous memory space. This chapter describes the memory and peripheral locations within that memory space.

## 4.2 System memory map

The following table shows the high-level device memory map. This map provides the complete architectural address space definition for the various sections. Based on the physical sizes of the memories and peripherals, the actual address regions used may be smaller.

**Table 4-1. System memory map**

| System 32-bit Address Range | Destination Slave | Access |
|---|---|---|
| 0x0000_0000–0x07FF_FFFF | Program flash and read-only data<br>(Includes exception vectors in first 1024 bytes) | All masters |
| 0x0800_0000–0x0FFF_FFFF | FlexBus (Aliased area) | Cortex-M4 core (M0) only |
| 0x1000_0000–0x13FF_FFFF | • For MK21FX512VLQ12: FlexNVM<br>• For MK21FN1M0VLQ12: Reserved<br>• For MK21FX512VMD12: FlexNVM<br>• For MK21FN1M0VMD12: Reserved | All masters |
| 0x1400_0000–0x17FF_FFFF | For devices with FlexNVM: FlexRAM | All masters |
| 0x1400_0000–0x17FF_FFFF | For devices with program flash only: Programming acceleration RAM | – |
| 0x1800_0000–0x1BFF_FFFF | FlexBus (Aliased Area) | Cortex-M4 core (M0) only |
| 0x1C00_0000–0x1FFF_FFFF | SRAM_L: Lower SRAM (ICODE/DCODE) | All masters |
| 0x2000_0000–0x200F_FFFF | SRAM_U: Upper SRAM bitband region | All masters |

*Table continues on the next page...*

**Table 4-1. System memory map (continued)**

| System 32-bit Address Range | Destination Slave | Access |
|---|---|---|
| 0x2010_0000–0x21FF_FFFF | Reserved | – |
| 0x2200_0000–0x23FF_FFFF | Aliased to TCMU SRAM bitband | Cortex-M4 core only |
| 0x2400_0000–0x3FFF_FFFF | Reserved | – |
| 0x4000_0000–0x4007_FFFF | Bitband region for AIPS0 | Cortex-M4 core & DMA/EzPort |
| 0x4008_0000–0x400F_EFFF | Bitband region for AIPS1 | Cortex-M4 core & DMA/EzPort |
| 0x400F_F000–0x400F_FFFF | Bitband region for GPIO | Cortex-M4 core & DMA/EzPort |
| 0x4010_0000–0x41FF_FFFF | Reserved | – |
| 0x4200_0000–0x43FF_FFFF | Aliased to AIPS and GPIO bitband | Cortex-M4 core only |
| 0x4400_0000–0x5FFF_FFFF | Reserved | – |
| 0x6000_0000–0x7FFF_FFFF | FlexBus (External Memory - Write-back) | All masters |
| 0x8000_0000–0x9FFF_FFFF | FlexBus (External Memory - Write-through) | All masters |
| 0xA000_0000–0xDFFF_FFFF | FlexBus (External Peripheral - Not executable) | All masters |
| 0xE000_0000–0xE00F_FFFF | Private Peripherals | Cortex-M4 core only |
| 0xE010_0000–0xFFFF_FFFF | Reserved | – |

**NOTE**

1. EzPort master port is statically muxed with DMA master port. Access rights to AIPS-Lite peripheral bridge and general purpose input/output (GPIO) module address space is limited to the core, DMA and EzPort.
2. ARM Cortex-M4 core access privileges also includes accesses via the debug interface.

## 4.2.1 Aliased bit-band regions

The SRAM_U, AIPS-Lite, and general purpose input/output (GPIO) module resources reside in the Cortex-M4 processor bit-band regions.

The processor also includes two 32 MB aliased bit-band regions associated with the two 1 MB bit-band spaces. Each 32-bit location in the 32 MB space maps to an individual bit in the bit-band region. A 32-bit write in the alias region has the same effect as a read-modify-write operation on the targeted bit in the bit-band region.

Bit 0 of the value written to the alias region determines what value is written to the target bit:

- Writing a value with bit 0 set writes a 1 to the target bit.
- Writing a value with bit 0 clear writes a 0 to the target bit.

A 32-bit read in the alias region returns either:

- a value of 0x0000_0000 to indicate the target bit is clear
- a value of 0x0000_0001 to indicate the target bit is set

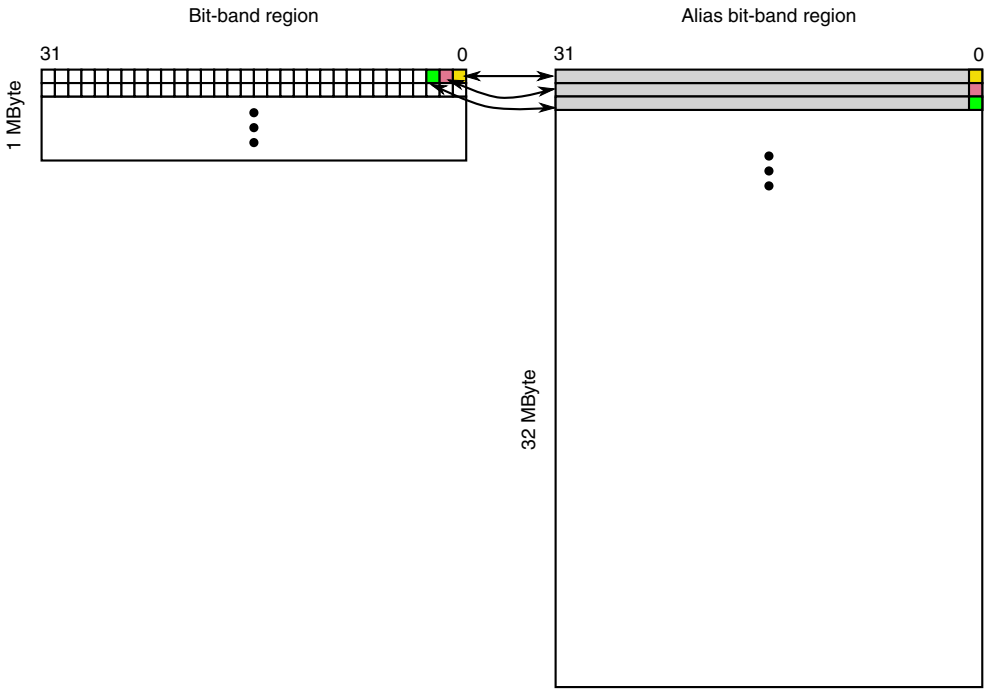



**Figure 4-1. Alias bit-band mapping**

**NOTE**

Each bit in bit-band region has an equivalent bit that can be manipulated through bit 0 in a corresponding long word in the alias bit-band region.

## 4.3 Flash Memory Map

The various flash memories and the flash registers are located at different base addresses as shown in the following figure. The base address for each is specified in System memory map.

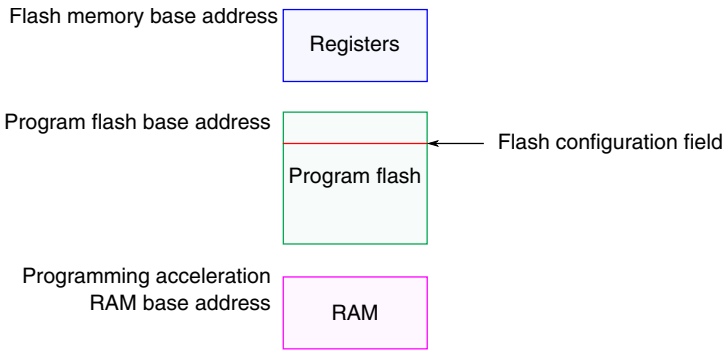



**Figure 4-2. Flash memory map for devices containing only program flash**

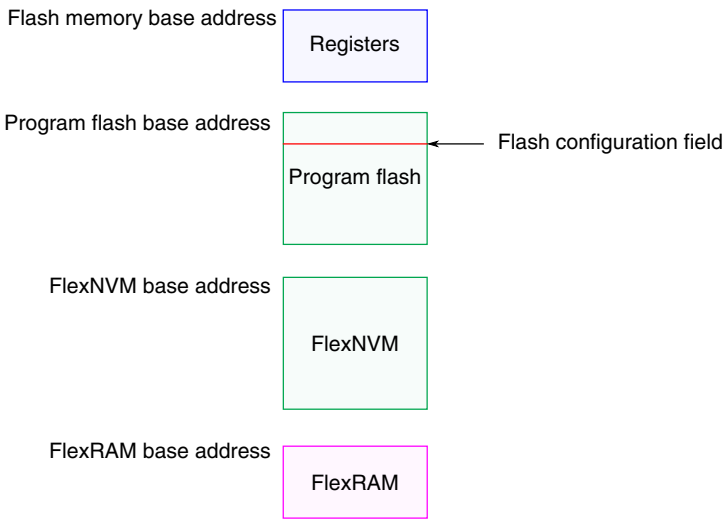



**Figure 4-3. Flash memory map for devices containing FlexNVM**

## 4.3.1 Alternate Non-Volatile IRC User Trim Description

The following non-volatile locations (4 bytes) are reserved for custom IRC user trim supported by some development tools. An alternate IRC trim to the factory loaded trim can be stored at this location. To override the factory trim, user software must load new values into the MCG trim registers.

| Non-Volatile Byte Address | Alternate IRC Trim Value |
|---|---|
| 0x0000_03FC | Reserved |
| 0x0000_03FD | Reserved |
| 0x0000_03FE (bit 0) | SCFTRIM |
| 0x0000_03FE (bit 4:1) | FCTRIM |
| 0x0000_03FF | SCTRIM |

## 4.4 SRAM memory map

The on-chip RAM is split in two regions: SRAM_L and SRAM_U. The RAM is implemented such that the SRAM_L and SRAM_U ranges form a contiguous block in the memory map. See SRAM Arrays for details.

Accesses to the SRAM_L and SRAM_U memory ranges outside the amount of RAM on the device causes the bus cycle to be terminated with an error followed by the appropriate response in the requesting bus master.

## 4.5 Peripheral bridge (AIPS-Lite0 and AIPS-Lite1) memory map

The peripheral memory map is accessible via two slave ports on the crossbar switch in the 0x4000_0000–0x400F_FFFF region. The device implements two peripheral bridges (AIPS-Lite 0 and 1):

- AIPS-Lite0 covers 512 KB
- AIPS-Lite1 covers 508 KB with 4 KB assigned to the general purpose input/output module (GPIO)

AIPS-Lite0 is connected to crossbar switch slave port 2, and is accessible at locations 0x4000_0000–0x4007_FFFF.

AIPS-Lite1 and the general purpose input/output module share the connection to crossbar switch slave port 3. The AIPS-Lite1 is accessible at locations 0x4008_0000–0x400F_EFFF. The general purpose input/output module is accessible in a 4-kbyte region at 0x400F_F000–0x400F_FFFF. Its direct connection to the crossbar switch provides master access without incurring wait states associated with accesses via the AIPS-Lite controllers.

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

For programming model accesses via the peripheral bridges, there is generally only a small range within the 4 KB slots that is implemented. Accessing an address that is not implemented in the peripheral results in a transfer error termination.

### 4.5.1 Read-after-write sequence and required serialization of memory operations

In some situations, a write to a peripheral must be completed fully before a subsequent action can occur. Examples of such situations include:

- Exiting an interrupt service routine (ISR)
- Changing a mode
- Configuring a function

In these situations, application software must perform a read-after-write sequence to guarantee the required serialization of the memory operations:

1. Write the peripheral register.
2. Read the written peripheral register to verify the write.
3. Continue with subsequent operations.

### 4.5.2 Peripheral Bridge 0 (AIPS-Lite 0) Memory Map

**Table 4-2. Peripheral bridge 0 slot assignments**

| System 32-bit base address | Slot number | Module |
|---|---|---|
| 0x4000_0000 | 0 | Peripheral bridge 0 (AIPS-Lite 0) |
| 0x4000_1000 | 1 | — |
| 0x4000_2000 | 2 | — |
| 0x4000_3000 | 3 | — |
| 0x4000_4000 | 4 | Crossbar switch |
| 0x4000_5000 | 5 | — |
| 0x4000_6000 | 6 | — |
| 0x4000_7000 | 7 | — |
| 0x4000_8000 | 8 | DMA controller |
| 0x4000_9000 | 9 | DMA controller transfer control descriptors |
| 0x4000_A000 | 10 | — |
| 0x4000_B000 | 11 | — |
| 0x4000_C000 | 12 | FlexBus |
| 0x4000_D000 | 13 | MPU |
| 0x4000_E000 | 14 | — |
| 0x4000_F000 | 15 | — |
| 0x4001_0000 | 16 | — |
| 0x4001_1000 | 17 | — |
| 0x4001_2000 | 18 | — |
| 0x4001_3000 | 19 | — |
| 0x4001_4000 | 20 | — |

*Table continues on the next page...*

## Table 4-2. Peripheral bridge 0 slot assignments (continued)

| System 32-bit base address | Slot number | Module |
|---|---|---|
| 0x4001_5000 | 21 | — |
| 0x4001_6000 | 22 | — |
| 0x4001_7000 | 23 | — |
| 0x4001_8000 | 24 | — |
| 0x4001_9000 | 25 | — |
| 0x4001_A000 | 26 | — |
| 0x4001_B000 | 27 | — |
| 0x4001_C000 | 28 | — |
| 0x4001_D000 | 29 | — |
| 0x4001_E000 | 30 | — |
| 0x4001_F000 | 31 | Flash memory controller |
| 0x4002_0000 | 32 | Flash memory |
| 0x4002_1000 | 33 | DMA channel mutiplexer 0 |
| 0x4002_2000 | 34 | — |
| 0x4002_3000 | 35 | — |
| 0x4002_4000 | 36 | FlexCAN 0 |
| 0x4002_5000 | 37 | — |
| 0x4002_6000 | 38 | — |
| 0x4002_7000 | 39 | — |
| 0x4002_8000 | 40 | — |
| 0x4002_9000 | 41 | [1] Random Number Generator (RNGA) |
| 0x4002_A000 | 42 | — |
| 0x4002_B000 | 43 | — |
| 0x4002_C000 | 44 | SPI 0 |
| 0x4002_D000 | 45 | SPI 1 |
| 0x4002_E000 | 46 | — |
| 0x4002_F000 | 47 | I2S 0 |
| 0x4003_0000 | 48 | — |
| 0x4003_1000 | 49 | — |
| 0x4003_2000 | 50 | CRC |
| 0x4003_3000 | 51 | — |
| 0x4003_4000 | 52 | — |
| 0x4003_5000 | 53 | USB DCD |
| 0x4003_6000 | 54 | Programmable delay block (PDB) |
| 0x4003_7000 | 55 | Periodic interrupt timers (PIT) |
| 0x4003_8000 | 56 | FlexTimer (FTM) 0 |
| 0x4003_9000 | 57 | FlexTimer (FTM) 1 |
| 0x4003_A000 | 58 | [2] FlexTimer (FTM) 2 |
| 0x4003_B000 | 59 | Analog-to-digital converter (ADC) 0 |

*Table continues on the next page...*

## Table 4-2. Peripheral bridge 0 slot assignments (continued)

| System 32-bit base address | Slot number | Module |
|---|---|---|
| 0x4003_C000 | 60 | — |
| 0x4003_D000 | 61 | Real-time clock (RTC) |
| 0x4003_E000 | 62 | VBAT register file |
| 0x4003_F000 | 63 | [3]DAC0 |
| 0x4004_0000 | 64 | Low-power timer (LPTMR) |
| 0x4004_1000 | 65 | System register file |
| 0x4004_2000 | 66 | — |
| 0x4004_3000 | 67 | — |
| 0x4004_4000 | 68 | — |
| 0x4004_5000 | 69 | — |
| 0x4004_6000 | 70 | — |
| 0x4004_7000 | 71 | SIM low-power logic |
| 0x4004_8000 | 72 | System integration module (SIM) |
| 0x4004_9000 | 73 | Port A multiplexing control |
| 0x4004_A000 | 74 | Port B multiplexing control |
| 0x4004_B000 | 75 | Port C multiplexing control |
| 0x4004_C000 | 76 | Port D multiplexing control |
| 0x4004_D000 | 77 | Port E multiplexing control |
| 0x4004_E000 | 78 | — |
| 0x4004_F000 | 79 | — |
| 0x4005_0000 | 80 | — |
| 0x4005_1000 | 81 | — |
| 0x4005_2000 | 82 | Software watchdog |
| 0x4005_3000 | 83 | — |
| 0x4005_4000 | 84 | — |
| 0x4005_5000 | 85 | — |
| 0x4005_6000 | 86 | — |
| 0x4005_7000 | 87 | — |
| 0x4005_8000 | 88 | — |
| 0x4005_9000 | 89 | — |
| 0x4005_A000 | 90 | — |
| 0x4005_B000 | 91 | — |
| 0x4005_C000 | 92 | — |
| 0x4005_D000 | 93 | — |
| 0x4005_E000 | 94 | — |
| 0x4005_F000 | 95 | — |
| 0x4006_0000 | 96 | — |
| 0x4006_1000 | 97 | External watchdog |
| 0x4006_2000 | 98 | Carrier modulator timer (CMT) |

*Table continues on the next page...*

## Table 4-2. Peripheral bridge 0 slot assignments (continued)

| System 32-bit base address | Slot number | Module |
|---|---|---|
| 0x4006_3000 | 99 | — |
| 0x4006_4000 | 100 | Multi-purpose Clock Generator (MCG) |
| 0x4006_5000 | 101 | System oscillator (OSC) |
| 0x4006_6000 | 102 | $I^2C$ 0 |
| 0x4006_7000 | 103 | $I^2C$ 1 |
| 0x4006_8000 | 104 | — |
| 0x4006_9000 | 105 | — |
| 0x4006_A000 | 106 | UART 0 |
| 0x4006_B000 | 107 | UART 1 |
| 0x4006_C000 | 108 | UART 2 |
| 0x4006_D000 | 109 | UART 3 |
| 0x4006_E000 | 110 | — |
| 0x4006_F000 | 111 | — |
| 0x4007_0000 | 112 | — |
| 0x4007_1000 | 113 | — |
| 0x4007_2000 | 114 | USB OTG FS/LS |
| 0x4007_3000 | 115 | Analog comparator (CMP) / 6-bit digital-to-analog converter (DAC) |
| 0x4007_4000 | 116 | Voltage reference (VREF) |
| 0x4007_5000 | 117 | — |
| 0x4007_6000 | 118 | — |
| 0x4007_7000 | 119 | — |
| 0x4007_8000 | 120 | — |
| 0x4007_9000 | 121 | — |
| 0x4007_A000 | 122 | — |
| 0x4007_B000 | 123 | — |
| 0x4007_C000 | 124 | Low-leakage wakeup unit (LLWU) |
| 0x4007_D000 | 125 | Power management controller (PMC) |
| 0x4007_E000 | 126 | System Mode controller (SMC) |
| 0x4007_F000 | 127 | Reset Control Module (RCM) |
| 0x400F_F000 | | GPIO controller |

1. Random Number Generator is also available in AIPS1
2. FTM2 is also available in AIPS1
3. DAC0 is also available in AIPS1

## 4.5.3 Peripheral Bridge 1 (AIPS-Lite 1) Memory Map

**Table 4-3. Peripheral bridge 1 slot assignments**

| System 32-bit base address | Slot number | Module |
|---|---|---|
| 0x4008_0000 | 0 | Peripheral bridge 1 (AIPS-Lite 1) |
| 0x4008_1000 | 1 | — |
| 0x4008_2000 | 2 | — |
| 0x4008_3000 | 3 | — |
| 0x4008_4000 | 4 | — |
| 0x4008_5000 | 5 | — |
| 0x4008_6000 | 6 | — |
| 0x4008_7000 | 7 | — |
| 0x4008_8000 | 8 | — |
| 0x4008_9000 | 9 | — |
| 0x4008_A000 | 10 | — |
| 0x4008_B000 | 11 | — |
| 0x4008_C000 | 12 | — |
| 0x4008_D000 | 13 | — |
| 0x4008_E000 | 14 | — |
| 0x4008_F000 | 15 | — |
| 0x4009_0000 | 16 | — |
| 0x4009_1000 | 17 | — |
| 0x4009_2000 | 18 | — |
| 0x4009_3000 | 19 | — |
| 0x4009_4000 | 20 | — |
| 0x4009_5000 | 21 | — |
| 0x4009_6000 | 22 | — |
| 0x4009_7000 | 23 | — |
| 0x4009_8000 | 24 | — |
| 0x4009_9000 | 25 | — |
| 0x4009_A000 | 26 | — |
| 0x4009_B000 | 27 | — |
| 0x4009_C000 | 28 | — |
| 0x4009_D000 | 29 | — |
| 0x4009_E000 | 30 | — |
| 0x4009_F000 | 31 | — |
| 0x400A_0000 | 32 | [1]Random number generator (RNGA) |
| 0x400A_1000 | 33 | — |
| 0x400A_2000 | 34 | — |
| 0x400A_3000 | 35 | — |
| 0x400A_4000 | 36 | — |

*Table continues on the next page...*

## Table 4-3. Peripheral bridge 1 slot assignments (continued)

| System 32-bit base address | Slot number | Module |
|---|---|---|
| 0x400A_5000 | 37 | — |
| 0x400A_6000 | 38 | — |
| 0x400A_7000 | 39 | — |
| 0x400A_8000 | 40 | — |
| 0x400A_9000 | 41 | — |
| 0x400A_A000 | 42 | — |
| 0x400A_B000 | 43 | — |
| 0x400A_C000 | 44 | SPI 2 |
| 0x400A_D000 | 45 | — |
| 0x400A_E000 | 46 | — |
| 0x400A_F000 | 47 | |
| 0x400B_0000 | 48 | — |
| 0x400B_1000 | 49 | eSDHC |
| 0x400B_2000 | 50 | — |
| 0x400B_3000 | 51 | — |
| 0x400B_4000 | 52 | — |
| 0x400B_5000 | 53 | — |
| 0x400B_6000 | 54 | — |
| 0x400B_7000 | 55 | — |
| 0x400B_8000 | 56 | [2]FlexTimer (FTM) 2 |
| 0x400B_9000 | 57 | FlexTimer (FTM) 3 |
| 0x400B_A000 | 58 | — |
| 0x400B_B000 | 59 | Analog-to-digital converter (ADC) 1 |
| 0x400B_C000 | 60 | — |
| 0x400B_D000 | 61 | — |
| 0x400B_E000 | 62 | — |
| 0x400B_F000 | 63 | — |
| 0x400C_0000 | 64 | — |
| 0x400C_1000 | 65 | — |
| 0x400C_2000 | 66 | — |
| 0x400C_3000 | 67 | — |
| 0x400C_4000 | 68 | — |
| 0x400C_5000 | 69 | — |
| 0x400C_6000 | 70 | — |
| 0x400C_7000 | 71 | — |
| 0x400C_8000 | 72 | — |
| 0x400C_9000 | 73 | — |
| 0x400C_A000 | 74 | — |
| 0x400C_B000 | 75 | — |

*Table continues on the next page...*

## Table 4-3. Peripheral bridge 1 slot assignments (continued)

| System 32-bit base address | Slot number | Module |
|---|---|---|
| 0x400C_C000 | 76 | [3]12-bit digital-to-analog converter (DAC) 0 |
| 0x400C_D000 | 77 | 12-bit digital-to-analog converter (DAC) 1 |
| 0x400C_E000 | 78 | — |
| 0x400C_F000 | 79 | — |
| 0x400D_0000 | 80 | — |
| 0x400D_1000 | 81 | — |
| 0x400D_2000 | 82 | — |
| 0x400D_3000 | 83 | — |
| 0x400D_4000 | 84 | — |
| 0x400D_5000 | 85 | — |
| 0x400D_6000 | 86 | — |
| 0x400D_7000 | 87 | — |
| 0x400D_8000 | 88 | — |
| 0x400D_9000 | 89 | — |
| 0x400D_A000 | 90 | — |
| 0x400D_B000 | 91 | — |
| 0x400D_C000 | 92 | — |
| 0x400D_D000 | 93 | — |
| 0x400D_E000 | 94 | — |
| 0x400D_F000 | 95 | — |
| 0x400E_0000 | 96 | — |
| 0x400E_1000 | 97 | — |
| 0x400E_2000 | 98 | — |
| 0x400E_3000 | 99 | — |
| 0x400E_4000 | 100 | — |
| 0x400E_5000 | 101 | — |
| 0x400E_6000 | 102 | I2C2 |
| 0x400E_7000 | 103 | — |
| 0x400E_8000 | 104 | — |
| 0x400E_9000 | 105 | — |
| 0x400E_A000 | 106 | UART 4 |
| 0x400E_B000 | 107 | UART 5 |
| 0x400E_C000 | 108 | — |
| 0x400E_D000 | 109 | — |
| 0x400E_E000 | 110 | — |
| 0x400E_F000 | 111 | — |
| 0x400F_0000 | 112 | — |
| 0x400F_1000 | 113 | — |
| 0x400F_2000 | 114 | — |

*Table continues on the next page...*

**Table 4-3. Peripheral bridge 1 slot assignments (continued)**

| System 32-bit base address | Slot number | Module |
|---|---|---|
| 0x400F_3000 | 115 | — |
| 0x400F_4000 | 116 | — |
| 0x400F_5000 | 117 | — |
| 0x400F_6000 | 118 | — |
| 0x400F_7000 | 119 | — |
| 0x400F_8000 | 120 | — |
| 0x400F_9000 | 121 | — |
| 0x400F_A000 | 122 | — |
| 0x400F_B000 | 123 | — |
| 0x400F_C000 | 124 | — |
| 0x400F_D000 | 125 | — |
| 0x400F_E000 | 126 | — |
| 0x400F_F000 | Not an AIPS-Lite slot. The 32-bit general purpose input/output module that shares the crossbar switch slave port with the AIPS-Lite is accessed at this address. | |

1. RNGA is also available in AIPS0
2. FTM2 is also available in AIPS0
3. DAC0 is also available in AIPS0

## 4.6 Private Peripheral Bus (PPB) memory map

The PPB is part of the defined ARM bus architecture and provides access to select processor-local modules. These resources are only accessible from the core; other system masters do not have access to them.

**Table 4-4. PPB memory map**

| System 32-bit Address Range | Resource |
|---|---|
| 0xE000_0000–0xE000_0FFF | Instrumentation Trace Macrocell (ITM) |
| 0xE000_1000–0xE000_1FFF | Data Watchpoint and Trace (DWT) |
| 0xE000_2000–0xE000_2FFF | Flash Patch and Breakpoint (FPB) |
| 0xE000_3000–0xE000_DFFF | Reserved |
| 0xE000_E000–0xE000_EFFF | System Control Space (SCS) (for NVIC and FPU) |
| 0xE000_F000–0xE003_FFFF | Reserved |
| 0xE004_0000–0xE004_0FFF | Trace Port Interface Unit (TPIU) |
| 0xE004_1000–0xE004_1FFF | Embedded Trace Macrocell (ETM) |
| 0xE004_2000–0xE004_2FFF | Embedded Trace Buffer (ETB) |
| 0xE004_3000–0xE004_3FFF | Embedded Trace Funnel |
| 0xE004_4000–0xE007_FFFF | Reserved |

*Table continues on the next page...*

**Table 4-4. PPB memory map (continued)**

| System 32-bit Address Range | Resource |
|---|---|
| 0xE008_0000–0xE008_0FFF | Miscellaneous Control Module (MCM)(including ETB Almost Full) |
| 0xE008_1000–0xE008_1FFF | Memory Mapped Cryptographic Acceleration Unit (MMCAU) |
| 0xE00F_F000–0xE00F_FFFF | ROM Table - allows auto-detection of debug components |

# Chapter 5
# Clock Distribution

## 5.1 Introduction

The MCG module controls which clock source is used to derive the system clocks. The clock generation logic divides the selected clock source into a variety of clock domains, including the clocks for the system bus masters, system bus slaves, and flash memory . The clock generation logic also implements module-specific clock gating to allow granular shutoff of modules.

The primary clocks for the system are generated from the MCGOUTCLK clock. The clock generation circuitry provides several clock dividers that allow different portions of the device to be clocked at different frequencies. This allows for trade-offs between performance and power dissipation.

Various modules, such as the USB OTG Controller, have module-specific clocks that can be generated from the MCGPLLCLK or MCGFLLCLK clock. In addition, there are various other module-specific clocks that have other alternate sources. Clock selection for most modules is controlled by the SOPT registers in the SIM module.

## 5.2 Programming model

The selection and multiplexing of system clock sources is controlled and programmed via the MCG module. The setting of clock dividers and module clock gating for the system are programmed via the SIM module. Reference those sections for detailed register and bit descriptions.

## 5.3 High-Level device clocking diagram

The following system oscillator, MCG, and SIM module registers control the multiplexers, dividers, and clock gates shown in the below figure:

| | OSC | MCG | SIM |
|---|---|---|---|
| Multiplexers | MCG_C*x* | MCG_C*x* | SIM_SOPT1, SIM_SOPT2 |
| Dividers | — | MCG_C*x* | SIM_CLKDIV*x* |
| Clock gates | OSC_CR | MCG_C1 | SIM_SCGC*x* |

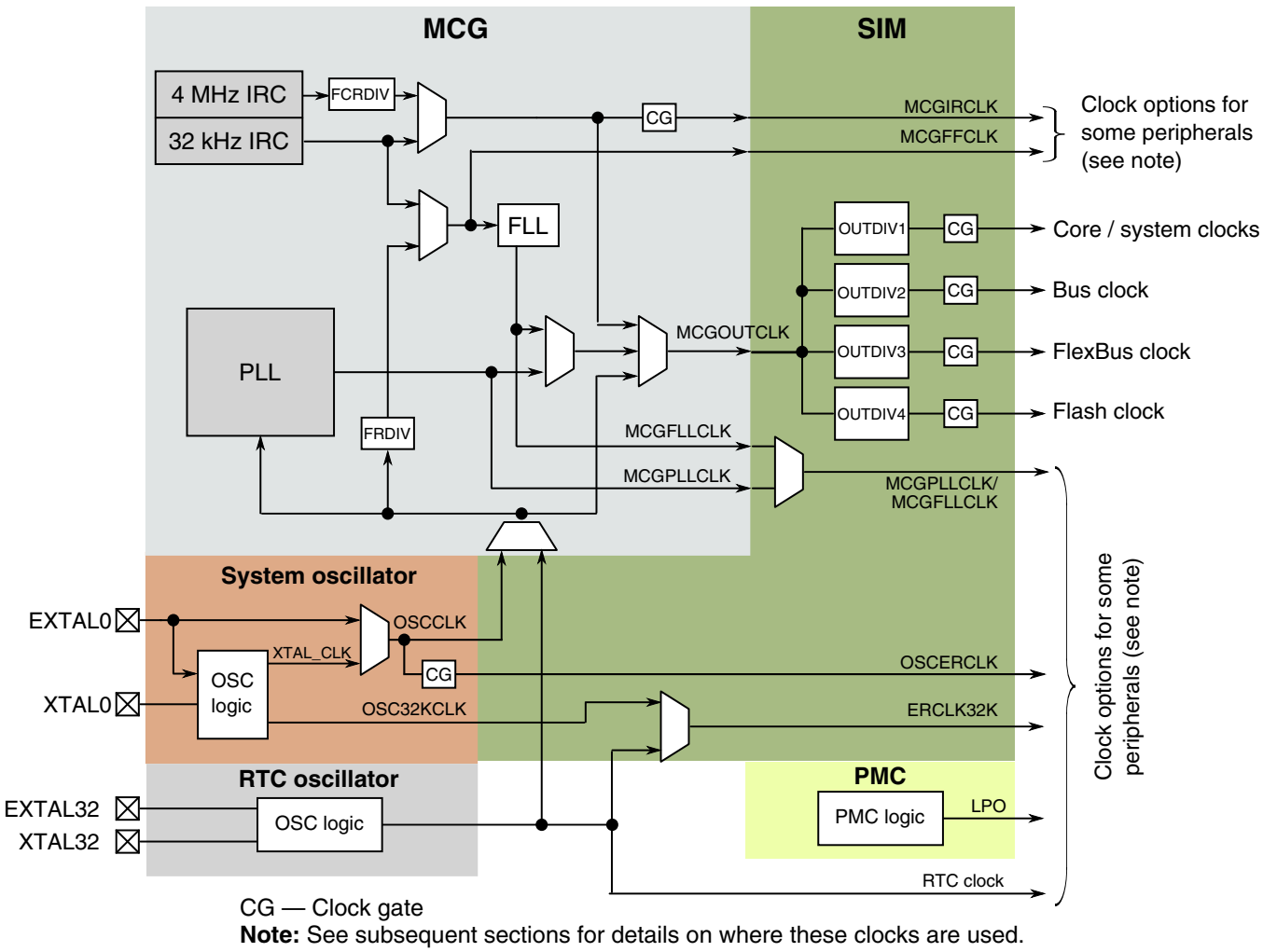



**Figure 5-1. Clocking diagram**

## 5.4 Clock definitions

The following table describes the clocks in the previous block diagram.

| Clock name | Description |
|---|---|
| Core clock | MCGOUTCLK divided by OUTDIV1 clocks the ARM Cortex-M4 core |

*Table continues on the next page...*

| Clock name | Description |
|---|---|
| System clock | MCGOUTCLK divided by OUTDIV1 clocks the crossbar switch and bus masters directly connected to the crossbar. In addition, this clock is used for UART0 and UART1. |
| Bus clock | MCGOUTCLK divided by OUTDIV2 clocks the bus slaves and peripheral (excluding memories) |
| FlexBus clock | MCGOUTCLK divided by OUTDIV3 clocks the external FlexBus interface |
| Flash clock | MCGOUTCLK divided by OUTDIV4 clocks the flash memory |
| MCGIRCLK | MCG output of the slow or fast internal reference clock |
| MCGFFCLK | MCG output of the slow internal reference clock or a divided MCG external reference clock. |
| MCGOUTCLK | MCG output of either IRC, MCGFLLCLK , MCGPLLCLK or MCG's external reference clock that sources the core, system, bus, FlexBus, and flash clock. It is also an option for the debug trace clock. |
| MCGFLLCLK | MCG output of the FLL. MCGFLLCLK may clock some modules. |
| MCGPLLCLK | MCG output of the PLL. MCGFLLCLK or MCGPLLCLK may clock some modules. |
| OSCCLK | System oscillator output of the internal oscillator or sourced directly from EXTAL |
| OSCERCLK | System oscillator output sourced from OSCCLKthat may clock some on-chip modules |
| OSC32KCLK | System oscillator 32kHz output |
| ERCLK32K | Clock source for some modules that is chosen as OSC32KCLK or the RTC clock. |
| RTC clock | RTC oscillator output for the RTC module and the DryIce module |
| LPO | PMC 1kHz output |

## 5.4.1 Device clock summary

The following table provides more information regarding the on-chip clocks.

**Table 5-1. Clock Summary**

| Clock name | Run mode clock frequency | VLPR mode clock frequency | Clock source | Clock is disabled when… |
|---|---|---|---|---|
| MCGOUTCLK | Up to 120 MHz | Up to 4 MHz | MCG | In all stop modes |
| Core clock | Up to 120 MHz | Up to 4 MHz | MCGOUTCLK clock divider | In all wait and stop modes |
| System clock | Up to 120 MHz | Up to 4 MHz | MCGOUTCLK clock divider | In all stop modes |
| Bus clock | Up to 60 MHz | Up to 4 MHz | MCGOUTCLK clock divider | In all stop modes |

*Table continues on the next page...*

## Table 5-1. Clock Summary (continued)

| Clock name | Run mode clock frequency | VLPR mode clock frequency | Clock source | Clock is disabled when… |
|---|---|---|---|---|
| FlexBus clock (FB_CLK) | Up to 50 MHz | Up to 4 MHz | MCGOUTCLK clock divider | In all stop modes or FlexBus disabled |
| Flash clock | Up to 25 MHz | Up to 800 kHz | MCGOUTCLK clock divider | In all stop modes |
| Internal reference (MCGIRCLK) | 30-40 kHz or MHz | 4 MHz only | MCG | MCG_C1[IRCLKEN] cleared, Stop mode and MCG_C1[IREFSTEN] cleared, or VLPS/LLS/VLLS mode |
| External reference (OSCERCLK) | Up to 50 MHz (bypass), 30-40 kHz, or 3-32 MHz (crystal) | Up to 4 MHz (bypass), 30-40 kHz (low-range crystal) or Up to 4 MHz (high-range crystal) | System OSC | System OSC's OSC_CR[ERCLKEN] cleared, or Stop mode and OSC_CR[EREFSTEN] cleared |
| External reference 32kHz (ERCLK32K) | 30-40 kHz | 30-40 kHz | System OSC or RTC OSC depending on SIM_SOPT1[OSC32KSEL] | System OSC's OSC_CR[ERCLKEN] cleared or RTC's RTC_CR[OSCE] cleared |
| RTC_CLKOUT | 1 Hz or 32 kHz | 1 Hz or 32 kHz | RTC clock | Clock is disabled in LLS and VLLSx modes |
| LPO | 1 kHz | 1 kHz | PMC | in VLLS0 |
| USB FS clock | 48 MHz | N/A | or MCGFLLCLK with fractional clock divider, or USB_CLKIN | USB FS OTG is disabled |
| I2S master clock | Up to 25 MHz | Up to 12.5 MHz | System clock , MCGPLLCLK , OSCERCLK with fractional clock divider, or I2S_CLKIN | $I^2S$ is disabled |
| SDHC clock | Up to 50 MHz | N/A | System clock, MCGPLLCLK/ MCGFLLCLK, or OSCERCLK | SDHC is disabled |
| TRACE clock | Up to 100 MHz | Up to 4 MHz | System clock or MCGOUTCLK | Trace is disabled |

## 5.5 Internal clocking requirements

The clock dividers are programmed via the SIM module's CLKDIV registers. Each divider is programmable from a divide-by-1 through divide-by-16 setting. The following requirements must be met when configuring the clocks for this device:

1. The core and system clock frequencies must be 120 MHz or slower.
2. The bus clock frequency must be programmed to 60 MHz or less and an integer divide of the core clock.
3. The flash clock frequency must be programmed to 25 MHz or less and an integer divide of the core clock.
4. The FlexBus clock frequency must be programmed to 50 Mhz or less and an integer divide of the core clock.

The following are a few of the more common clock configurations for this device:

Option 1:

| Clock | Frequency |
|---|---|
| Core clock | 120 MHz |
| System clock | 120 MHz |
| Bus clock | 60 MHz |
| FlexBus clock | 40 MHz |
| Flash clock | 24 MHz |

### 5.5.1 Clock divider values after reset

Each clock divider is programmed via the SIM module's CLKDIV*n* registers. The flash memory's FTF_FOPT[LPBOOT] bit controls the reset value of the core clock, system clock, bus clock, and flash clock dividers as shown below:

| FTF_FOPT [LPBOOT] | Core/system clock | Bus clock | FlexBus clock | Flash clock | Description |
|---|---|---|---|---|---|
| 0 | 0x7 (divide by 8) | 0x7 (divide by 8) | 0xF (divide by 16) | 0xF (divide by 16) | Low power boot |
| 1 | 0x0 (divide by 1) | 0x0 (divide by 1) | 0x1 (divide by 2) | 0x1 (divide by 2) | Fast clock boot |

This gives the user flexibility for a lower frequency, low-power boot option. The flash erased state defaults to fast clocking mode, since where the low power boot (FTF_FOPT[LPBOOT]) bit resides in flash is logic 1 in the flash erased state.

To enable the low power boot option program FTF_FOPT[LPBOOT] to zero. During the reset sequence, if LPBOOT is cleared, the system is in a slow clock configuration. Upon any system reset, the clock dividers return to this configurable reset state.

### 5.5.2 VLPR mode clocking

The clock dividers cannot be changed while in VLPR mode. They must be programmed prior to entering VLPR mode to guarantee:

- the core/system, FlexBus, and bus clocks are less than or equal to 4 MHz, and
- the flash memory clock is less than or equal to 1 MHz

**NOTE**

When the MCG is in BLPI and clocking is derived from the Fast IRC, the clock divider controls, MCG_SC[FCRDIV] and SIM_CLKDIV1[OUTDIV4], must be programmed such that the resulting flash clock nominal frequency is 800 kHz or less. In this case, one example of correct configuration is MCG_SC[FCRDIV]=000b and SIM_CLKDIV1[OUTDIV4]=0100b, resulting in a divide by 5 setting.

## 5.6 Clock Gating

The clock to each module can be individually gated on and off using the SIM module's SCGC*x* registers. These bits are cleared after any reset, which disables the clock to the corresponding module to conserve power. Prior to initializing a module, set the corresponding bit in SCGC*x* register to enable the clock. Before turning off the clock, make sure to disable the module.

Any bus access to a peripheral that has its clock disabled generates an error termination.

## 5.7 Module clocks

The following table summarizes the clocks associated with each module.

**Table 5-2. Module clocks**

| Module | Bus interface clock | Internal clocks | I/O interface clocks |
|---|---|---|---|
| **Core modules** | | | |

*Table continues on the next page...*

## Table 5-2. Module clocks (continued)

| Module | Bus interface clock | Internal clocks | I/O interface clocks |
|---|---|---|---|
| ARM Cortex-M4 core | System clock | Core clock | — |
| NVIC | System clock | — | — |
| DAP | System clock | — | — |
| ITM | System clock | — | — |
| ETM | System clock | TRACE clock | TRACE_CLKOUT |
| ETB | System clock | — | — |
| cJTAG, JTAGC | — | — | JTAG_CLK |
| **System modules** | | | |
| DMA | System clock | — | — |
| DMA Mux | Bus clock | — | — |
| Port control | Bus clock | LPO | — |
| Crossbar Switch | System clock | — | — |
| Peripheral bridges | System clock | Bus clock, Flash clock | — |
| MPU | System clock | — | — |
| LLWU, PMC, SIM, RCM | Flash clock | LPO | — |
| Mode controller | Flash clock | — | — |
| MCM | System clock | — | — |
| EWM | Bus clock | LPO | — |
| Watchdog timer | Bus clock | LPO | — |
| **Clocks** | | | |
| MCG | Bus clock | MCGOUTCLK, MCGPLLCLK, MCGFLLCLK, MCGIRCLK, OSCERCLK | — |
| OSC | Bus clock | OSCERCLK | — |
| **Memory and memory interfaces** | | | |
| Flash Controller | System clock | Flash clock | — |
| Flash memory | Flash clock | — | — |
| FlexBus | System clock | — | CLKOUT |
| EzPort | System clock | — | EZP_CLK |
| **Security** | | | |
| CRC | Bus clock | — | — |
| MMCAU | System clock | — | — |
| | Bus clock | — | — |
| **Analog** | | | |
| ADC | Bus clock | OSCERCLK | — |
| CMP | Bus clock | — | — |
| VREF | Bus clock | — | — |
| **Timers** | | | |
| PDB | Bus clock | — | — |
| FlexTimers | Bus clock | MCGFFCLK | FTM_CLKIN*x* |

*Table continues on the next page...*

**Table 5-2. Module clocks (continued)**

| Module | Bus interface clock | Internal clocks | I/O interface clocks |
|---|---|---|---|
| PIT | Bus clock | — | — |
| LPTMR | Flash clock | LPO, OSCERCLK, MCGIRCLK, ERCLK32K | — |
| CMT | Bus clock | — | — |
| RTC | Flash clock | EXTAL32 | — |
| **Communication interfaces** | | | |
| USB FS OTG | System clock | USB FS clock | — |
| USB DCD | Bus clock | — | — |
| FlexCAN | Bus clock | OSCERCLK | — |
| DSPI | Bus clock | — | DSPI_SCK |
| $I^2C$ | Bus clock | — | I2C_SCL |
| UART0, UART1 | System clock | — | — |
| UART2-5 | Bus clock | — | — |
| SDHC | System clock | SDHC clock | SDHC_DCLK |
| $I^2S$ | Bus clock | $I^2S$ master clock | I2S_TX_BCLK, I2S_RX_BCLK |
| **Human-machine interfaces** | | | |
| GPIO | System clock | — | — |

## 5.7.1 PMC 1-kHz LPO clock

The Power Management Controller (PMC) generates a 1-kHz clock that is enabled in all modes of operation, including all low power modes except VLLS0. This 1-kHz source is commonly referred to as LPO clock or 1-kHz LPO clock.

## 5.7.2 WDOG clocking

The WDOG may be clocked from two clock sources as shown in the following figure.

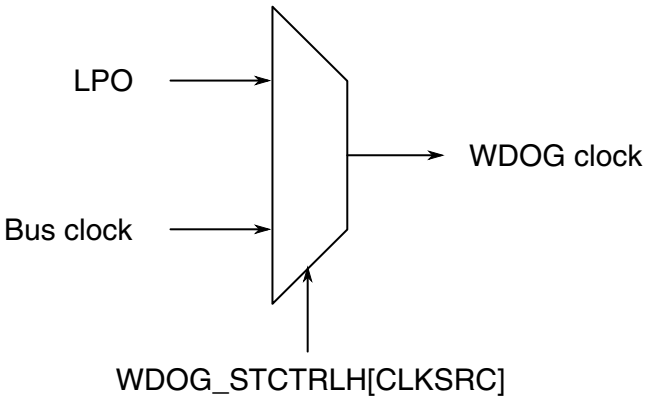



**Figure 5-2. WDOG clock generation**

### 5.7.3 Debug trace clock

The debug trace clock source can be clocked as shown in the following figure.

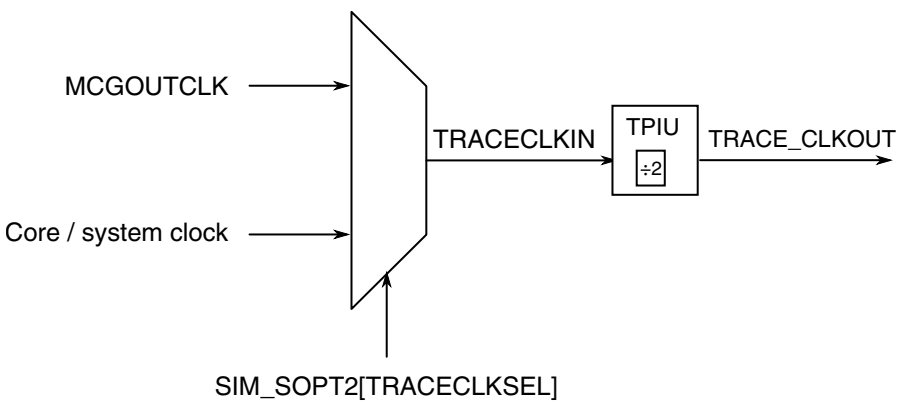



**Figure 5-3. Trace clock generation**

### 5.7.4 PORT digital filter clocking

The digital filters in the PORTD module can be clocked as shown in the following figure.

**NOTE**

In stop mode, the digital input filters are bypassed unless they are configured to run from the 1 kHz LPO clock source.

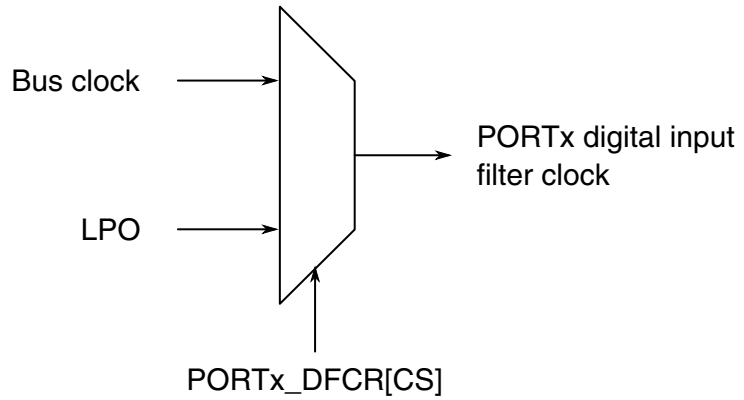



**Figure 5-4. PORTx digital input filter clock generation**

## 5.7.5 LPTMR clocking

The prescaler and glitch filters in each of the LPTMR*x* modules can be clocked as shown in the following figure.

**NOTE**

The chosen clock must remain enabled if the LPTMR*x* is to continue operating in all required low-power modes.

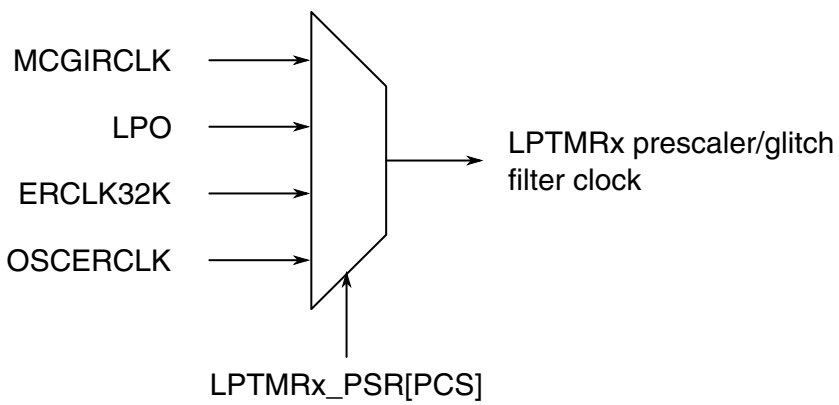



**Figure 5-5. LPTMRx prescaler/glitch filter clock generation**

## 5.7.6 USB FS OTG Controller clocking

The USB FS OTG controller is a bus master attached to the crossbar switch. As such, it uses the system clock.

**NOTE**

For the USB FS OTG controller to operate, the minimum system clock frequency is 20 MHz.

The USB OTG controller also requires a 48 MHz clock. The clock source options are shown below.

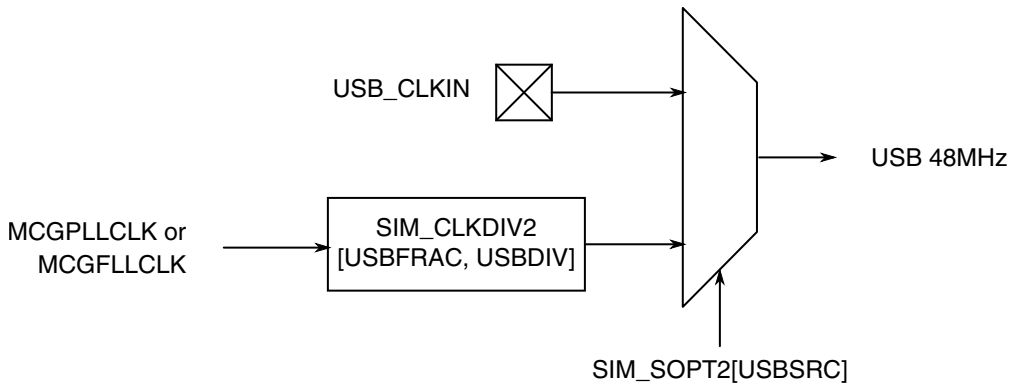



**Figure 5-6. USB 48 MHz clock source**

**NOTE**

The MCGFLLCLK does not meet the USB jitter specifications for certification.

## 5.7.7 FlexCAN clocking

The clock for the FlexCAN's protocol engine can be selected as shown in the following figure.

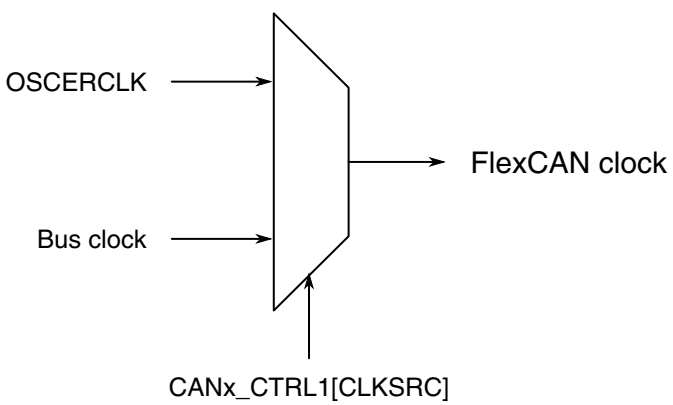



**Figure 5-7. FlexCAN clock generation**

## 5.7.8 UART clocking

UART0 and UART1 modules operate from the core/system clock, which provides higher performance level for these modules. All other UART modules operate from the bus clock.

## 5.7.9 SDHC clocking

The SDHC module has four possible clock sources for the external clock source, as shown in the following figure.

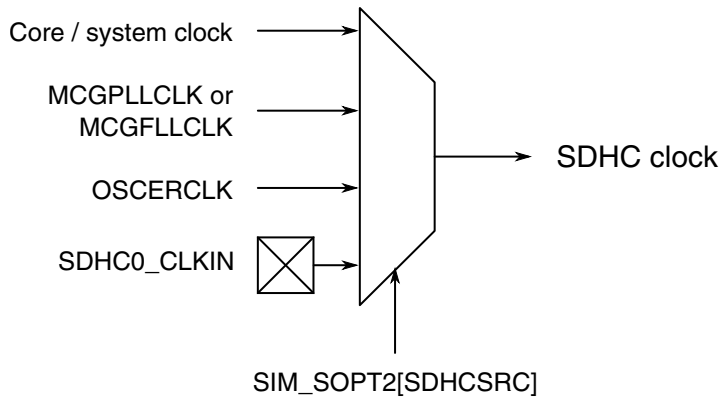



**Figure 5-8. SDHC clock generation**

**NOTE**

SDHC0_CLKIN is not available on all packages. See the Pinout section for more information.

## 5.7.10 I²S/SAI clocking

The audio master clock (MCLK) is used to generate the bit clock when the receiver or transmitter is configured for an internally generated bit clock. The audio master clock can also be output to or input from a pin. The transmitter and receiver have the same audio master clock inputs.

Each SAI peripheral can control the input clock selection, pin direction and divide ratio of one audio master clock.

The I²S/SAI transmitter and receiver support asynchronous bit clocks (BCLKs) that can be generated internally from the audio master clock or supplied externally. The module also supports the option for synchronous operation between the receiver and transmitter product.

The transmitter and receiver can independently select between the bus clock and the audio master clock to generate the bit clock.

The MCLK and BCLK source options appear in the following figure.

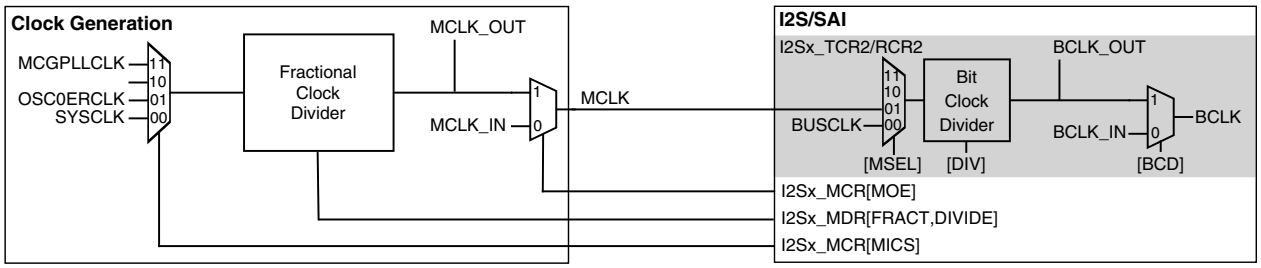



**Figure 5-9. I²S/SAI clock generation**

# Chapter 6
# Reset and Boot

## 6.1 Introduction

The following reset sources are supported in this MCU:

**Table 6-1. Reset sources**

| Reset sources | Description |
|---|---|
| POR reset | • Power-on reset (POR) |
| System resets | • External pin reset (PIN)<br>• Low-voltage detect (LVD)<br>• Computer operating properly (COP) watchdog reset<br>• Low leakage wakeup (LLWU) reset<br>• Multipurpose clock generator loss of clock (LOC) reset<br>• Multipurpose clock generator loss of lock (LOL) reset<br>• Stop mode acknowledge error (SACKERR)<br>• Software reset (SW)<br>• Lockup reset (LOCKUP)<br>• EzPort reset<br>• MDM DAP system reset |
| Debug reset | • JTAG reset<br>• nTRST reset |

Each of the system reset sources has an associated bit in the system reset status (SRS) registers. See the Reset Control Module for register details.

The MCU exits reset in functional mode that is controlled by $\overline{\text{EZP\_CS}}$ pin to select between the single chip (default) or serial flash programming (EzPort) modes. See Boot options for more details.

## 6.2 Reset

This section discusses basic reset mechanisms and sources. Some modules that cause resets can be configured to cause interrupts instead. Consult the individual peripheral chapters for more information.

### 6.2.1 Power-on reset (POR)

When power is initially applied to the MCU or when the supply voltage drops below the power-on reset re-arm voltage level ($V_{POR}$), the POR circuit causes a POR reset condition.

As the supply voltage rises, the LVD circuit holds the MCU in reset until the supply has risen above the LVD low threshold ($V_{LVDL}$). The POR and LVD bits in SRS0 register are set following a POR.

### 6.2.2 System reset sources

Resetting the MCU provides a way to start processing from a known set of initial conditions. System reset begins with the on-chip regulator in full regulation and system clocking generation from an internal reference. When the processor exits reset, it performs the following:

- Reads the start SP (SP_main) from vector-table offset 0
- Reads the start PC from vector-table offset 4
- LR is set to 0xFFFF_FFFF

The on-chip peripheral modules are disabled and the non-analog I/O pins are initially configured as disabled. The pins with analog functions assigned to them default to their analog function after reset.

During and following a reset, the JTAG pins have their associated input pins configured as:

- TDI in pull-up (PU)
- TCK in pull-down (PD)
- TMS in PU

and associated output pin configured as:

- TDO with no pull-down or pull-up

Note that the nTRST signal is initially configured as disabled, however once configured to its JTAG functionality its associated input pin is configured as:

- nTRST in PU

### 6.2.2.1 External pin reset (PIN)

On this device, $\overline{\text{RESET}}$ is a dedicated pin. This pin is open drain and has an internal pullup device. Asserting $\overline{\text{RESET}}$ wakes the device from any mode. During a pin reset, the RCM's SRS0[PIN] bit is set.

#### 6.2.2.1.1 Reset pin filter

The $\overline{\text{RESET}}$ pin filter supports filtering from both the 1 kHz LPO clock and the bus clock. A separate filter is implemented for each clock source. In stop and VLPS mode operation, this logic either switches to bypass operation or has continued filtering operation depending on the filtering mode selected. In low leakage stop modes, a separate LPO filter in the LLWU can continue filtering the $\overline{\text{RESET}}$ pin.

The RPFC[RSTFLTSS], RPFC[RSTFLTSRW], and RPFW[RSTFLTSEL] fields in the reset control (RCM) register set control this functionality; see the RCM chapter. The filters are asynchronously reset by Chip POR. The reset value for each filter assumes the RESET pin is negated.

The two clock options for the $\overline{\text{RESET}}$ pin filter when the chip is not in low leakage modes are the LPO (1 kHz) and bus clock. For low leakage modes VLLS3, VLLS2, VLLS1, VLLS0, the LLWU provides control (in the LLWU_RST register) of an optional fixed digital filter running the LPO. When entering VLLS0, the RESET pin filter is disabled and bypassed.

The LPO filter has a fixed filter value of 3. Due to a synchronizer on the input data, there is also some associated latency (2 cycles). As a result, 5 cycles are required to complete a transition from low to high or high to low.

The bus filter initializes to off (logic 1) when the bus filter is not enabled. The bus clock is used when the filter selects bus clock, and the number of counts is controlled by the RCM's RPFW[RSTFLTSEL] field.

### 6.2.2.2 Low-voltage detect (LVD)

The chip includes a system for managing low voltage conditions to protect memory contents and control MCU system states during supply voltage variations. The system consists of a power-on reset (POR) circuit and an LVD circuit with a user-selectable trip voltage. The LVD system is always enabled in normal run, wait, or stop mode. The LVD system is disabled when entering VLPx, LLS, or VLLSx modes.

The LVD can be configured to generate a reset upon detection of a low voltage condition by setting the PMC's LVDSC1[LVDRE] bit to 1. The low voltage detection threshold is determined by the PMC's LVDSC1[LVDV] field. After an LVD reset has occurred, the LVD system holds the MCU in reset until the supply voltage has risen above the low voltage detection threshold. The RCM's SRS0[LVD] bit is set following either an LVD reset or POR.

### 6.2.2.3 Computer operating properly (COP) watchdog timer

The computer operating properly (COP) watchdog timer (WDOG) monitors the operation of the system by expecting periodic communication from the software. This communication is generally known as servicing (or refreshing) the COP watchdog. If this periodic refreshing does not occur, the watchdog issues a system reset. The COP reset causes the RCM's SRS0[WDOG] bit to set.

### 6.2.2.4 Low leakage wakeup (LLWU)

The LLWU module provides the means for a number of external pins, the $\overline{\text{RESET}}$ pin, and a number of internal peripherals to wake the MCU from low leakage power modes. The LLWU module is functional only in low leakage power modes.

- In LLS mode, only the $\overline{\text{RESET}}$ pin via the LLWU can generate a system reset.
- In VLLSx modes, all enabled inputs to the LLWU can generate a system reset.

After a system reset, the LLWU retains the flags indicating the input source of the last wakeup until the user clears them.

**NOTE**

Some flags are cleared in the LLWU and some flags are required to be cleared in the peripheral module. Refer to the individual peripheral chapters for more information.

### 6.2.2.5 Multipurpose clock generator loss-of-clock (LOC)

The MCG module supports an external reference clock.

If the C6[CME] bit in the MCG module is set, the clock monitor is enabled. If the external reference falls below $f_{loc\_low}$ or $f_{loc\_high}$, as controlled by the C2[RANGE] field in the MCG module, the MCU resets. The RCM's SRS0[LOC] bit is set to indicate this reset source.

**NOTE**

To prevent unexpected loss of clock reset events, all clock monitors should be disabled before entering any low power modes, including VLPR and VLPW.

### 6.2.2.6 MCG loss-of-lock (LOL) reset

The MCG includes a PLL loss-of-lock detector. The detector is enabled when configured for PEE and lock has been achieved. If the MCG_C8[LOLRE] bit in the MCG module is set and the PLL lock status bit (MCG_S[LOLS0]) becomes set, the MCU resets. The RCM_SRS0[LOL] bit is set to indicate this reset source.

**NOTE**

This reset source does not cause a reset if the chip is in any stop mode.

### 6.2.2.7 Stop mode acknowledge error (SACKERR)

This reset is generated if the core attempts to enter stop mode, but not all modules acknowledge stop mode within 1025 cycles of the 1 kHz LPO clock.

A module might not acknowledge the entry to stop mode if an error condition occurs. The error can be caused by a failure of an external clock input to a module.

### 6.2.2.8 Software reset (SW)

The SYSRESETREQ bit in the NVIC application interrupt and reset control register can be set to force a software reset on the device. (See ARM's NVIC documentation for the full description of the register fields, especially the VECTKEY field requirements.) Setting SYSRESETREQ generates a software reset request. This reset forces a system reset of all major components except for the debug module. A software reset causes the RCM's SRS1[SW] bit to set.

#### 6.2.2.9 Lockup reset (LOCKUP)

The LOCKUP gives immediate indication of seriously errant kernel software. This is the result of the core being locked because of an unrecoverable exception following the activation of the processor's built in system state protection hardware.

The LOCKUP condition causes a system reset and also causes the RCM's SRS1[LOCKUP] bit to set.

#### 6.2.2.10 Tamper detect reset (TAMPER)

A tamper detect condition can optionally cause a system reset and also causes the RCM's SRS1[TAMPER] bit to set.

#### 6.2.2.11 EzPort reset

The EzPort supports a system reset request via EzPort signaling. The EzPort generates a system reset request following execution of a Reset Chip (RESET) command via the EzPort interface. This method of reset allows the chip to boot from flash memory after it has been programmed by an external source. The EzPort is enabled or disabled by the $\overline{\text{EZP\_CS}}$ pin.

An EzPort reset causes the RCM's SRS1[EZPT] bit to set.

#### 6.2.2.12 MDM-AP system reset request

Set the system reset request bit in the MDM-AP control register to initiate a system reset. This is the primary method for resets via the JTAG/SWD interface. The system reset is held until this bit is cleared.

Set the core hold reset bit in the MDM-AP control register to hold the core in reset as the rest of the chip comes out of system reset.

### 6.2.3 MCU Resets

A variety of resets are generated by the MCU to reset different modules.

#### 6.2.3.1 VBAT POR

The VBAT POR asserts on a VBAT POR reset source. It affects only the modules within the VBAT power domain: RTC, DryIce, and VBAT Register File. These modules are not affected by the other reset types.

#### 6.2.3.2 POR Only

The POR Only reset asserts on the POR reset source only. It resets the PMC and System Register File.

The POR Only reset also causes all other reset types (except VBAT POR) to occur.

#### 6.2.3.3 Chip POR not VLLS

The Chip POR not VLLS reset asserts on POR and LVD reset sources. It resets parts of the SMC and SIM. It also resets the LPTMR.

The Chip POR not VLLS reset also causes these resets to occur: Chip POR, Chip Reset not VLLS, and Chip Reset (including Early Chip Reset).

#### 6.2.3.4 Chip POR

The Chip POR asserts on POR, LVD, and VLLS Wakeup reset sources. It resets the Reset Pin Filter registers and parts of the SIM and MCG.

The Chip POR also causes the Chip Reset (including Early Chip Reset) to occur.

#### 6.2.3.5 Chip Reset not VLLS

The Chip Reset not VLLS reset asserts on all reset sources except a VLLS Wakeup that does not occur via the $\overline{\text{RESET}}$ pin. It resets parts of the SMC, LLWU, and other modules that remain powered during VLLS mode.

The Chip Reset not VLLS reset also causes the Chip Reset (including Early Chip Reset) to occur.

#### 6.2.3.6 Early Chip Reset

The Early Chip Reset asserts on all reset sources. It resets only the flash memory module. It negates before flash memory initialization begins ("earlier" than when the Chip Reset negates).

#### 6.2.3.7 Chip Reset

Chip Reset asserts on all reset sources and only negates after flash initialization has completed and the RESET pin has also negated. It resets the remaining modules (the modules not reset by other reset types).

### 6.2.4 Reset Pin

For all reset sources except a VLLS Wakeup that does not occur via the RESET pin, the RESET pin is driven low by the MCU for at least 128 bus clock cycles and until flash initialization has completed.

After flash initialization has completed, the RESET pin is released, and the internal Chip Reset negates after the RESET pin is pulled high. Keeping the RESET pin asserted externally delays the negation of the internal Chip Reset.

### 6.2.5 Debug resets

The following sections detail the debug resets available on the device.

#### 6.2.5.1 JTAG reset

The JTAG module generate a system reset when certain IR codes are selected. This functional reset is asserted when EzPort, EXTEST, HIGHZ and CLAMP instructions are active. The reset source from the JTAG module is released when any other IR code is selected. A JTAG reset causes the RCM's SRS1[JTAG] bit to set.

### 6.2.5.2 nTRST reset

The nTRST pin causes a reset of the JTAG logic when asserted. Asserting the nTRST pin allows the debugger to gain control of the TAP controller state machine (after exiting LLS or VLLSx) without resetting the state of the debug modules.

The nTRST pin does not cause a system reset.

### 6.2.5.3 Resetting the Debug subsystem

Use the CDBGRSTREQ bit within the SWJ-DP CTRL/STAT register to reset the debug modules. However, as explained below, using the CDBGRSTREQ bit does not reset all debug-related registers.

CDBGRSTREQ resets the debug-related registers within the following modules:

- SWJ-DP
- AHB-AP
- ETM
- ATB replicators
- ATB upsizers
- ATB funnels
- ETB
- TPIU
- MDM-AP (MDM control and status registers)
- MCM (ETB “Almost Full” logic)

CDBGRSTREQ does not reset the debug-related registers within the following modules:

- CM4 core (core debug registers: DHCSR, DCRSR, DCRDR, DEMCR)
- FPB
- DWT
- ITM
- NVIC
- Crossbar bus switch[1]
- AHB-AP[1]
- Private peripheral bus[1]

1. CDBGRSTREQ does not affect AHB resources so that debug resources on the private peripheral bus are available during System Reset.

## 6.3 Boot

This section describes the boot sequence, including sources and options.

### 6.3.1 Boot sources

This device only supports booting from internal flash. Any secondary boot must go through an initialization sequence in flash.

### 6.3.2 Boot options

The device's functional mode is controlled by the state of the EzPort chip select ($\overline{\text{EZP\_CS}}$) pin during reset.

The device can be in single chip (default) or serial flash programming mode (EzPort). While in single chip mode the device can be in run or various low power modes mentioned in Power mode transitions.

**Table 6-2. Mode select decoding**

| EzPort chip select ($\overline{\text{EZP\_CS}}$) | Description |
|---|---|
| 0 | Serial flash programming mode (EzPort) |
| 1 | Single chip (default) |

### 6.3.3 FOPT boot options

The flash option register (FOPT) in the flash memory module allows the user to customize the operation of the MCU at boot time. The register contains read-only bits that are loaded from the NVM's option byte in the flash configuration field. The user can reprogram the option byte in flash to change the FOPT values that are used for subsequent resets. For more details on programming the option byte, refer to the flash memory chapter.

The MCU uses the FOPT register bits to configure the device at reset as shown in the following table.

**Table 6-3. Flash Option Register Bit Definitions**

| Bit Num | Field | Value | Definition |
|---|---|---|---|
| 7-3 | Reserved | Reserved for future expansion. | |
| 2 | NMI_DIS | Enable/disable control for the NMI function. | |
| | | 0 | NMI interrupts are always blocked. The associated pin continues to default to NMI pin controls with internal pullup enabled. |
| | | 1 | NMI pin/interrupts reset default to enabled. |
| 1 | EZPORT_DIS | Enable/disable EzPort function. | |
| | | 0 | EzPort operation is disabled. The device always boots to normal CPU execution and the state of $\overline{\text{EZP\_CS}}$ signal during reset is ignored. This option avoids inadvertent resets into EzPort mode if the $\overline{\text{EZP\_CS}}$/NMI pin is used for its NMI function. |
| | | 1 | EzPort operation is enabled. The state of $\overline{\text{EZP\_CS}}$ pin during reset determines if device enters EzPort mode. |
| 0 | LPBOOT | Control the reset value of OUTDIVx values in SIM_CLKDIV1 register. Larger divide value selections produce lower average power consumption during POR, VLLSx recoveries and reset sequencing and after reset exit. | |
| | | 0 | Low-power boot: OUTDIVx values in SIM_CLKDIV1 register are auto-configured at reset exit for higher divide values that produce lower power consumption at reset exit.<br>• Core and system clock divider (OUTDIV1) and bus clock divider (OUTDIV2) are 0x7 (divide by 8)<br>• Flash clock divider (OUTDIV4) and FlexBus clock divider (OUTDIV3) are 0xF (divide by 16) |
| | | 1 | Normal boot: OUTDIVx values in SIM_CLKDIV1 register are auto-configured at reset exit for higher frequency values that produce faster operating frequencies at reset exit.<br>• Core and system clock divider (OUTDIV1) and bus clock divider (OUTDIV2) are 0x0 (divide by 1)<br>• Flash clock divider (OUTDIV4) and FlexBus clock divider (OUTDIV3) are 0x1 (divide by 2) |

## 6.3.4 Boot sequence

At power up, the on-chip regulator holds the system in a POR state until the input supply is above the POR threshold. The system continues to be held in this static state until the internally regulated supplies have reached a safe operating voltage as determined by the LVD. The Mode Controller reset logic then controls a sequence to exit reset.

1. A system reset is held on internal logic, the $\overline{\text{RESET}}$ pin is driven out low, and the MCG is enabled in its default clocking mode.

2. Required clocks are enabled (Core Clock, System Clock, Flash Clock, and any Bus Clocks that do not have clock gate control).
3. The system reset on internal logic continues to be held, but the Flash Controller is released from reset and begins initialization operation while the Mode Control logic continues to drive the RESET pin out low for a count of ~128 Bus Clock cycles.
4. The RESET pin is released, but the system reset of internal logic continues to be held until the Flash Controller finishes initialization. EzPort mode is selected instead of the normal CPU execution if EZP_CS is low when the internal reset is deasserted. EzPort mode can be disabled by programming the FOPT[EZPORT_DIS] field in the Flash Memory module.
5. When Flash Initialization completes, the RESET pin is observed. If RESET continues to be asserted (an indication of a slow rise time on the RESET pin or external drive in low), the system continues to be held in reset. Once the RESET pin is detected high, the system is released from reset.
6. At release of system reset, clocking is switched to a slow clock if the FOPT[LPBOOT] field in the Flash Memory module is configured for Low Power Boot
7. When the system exits reset, the processor sets up the stack, program counter (PC), and link register (LR). The processor reads the start SP (SP_main) from vector-table offset 0. The core reads the start PC from vector-table offset 4. LR is set to 0xFFFF_FFFF. What happens next depends on the NMI input and the FOPT[NMI_DIS] field in the Flash Memory module:
   - If the NMI input is high or the NMI function is disabled in the NMI_DIS field, the CPU begins execution at the PC location.
   - If the NMI input is low and the NMI function is enabled in the NMI_DIS field, this results in an NMI interrupt. The processor executes an Exception Entry and reads the NMI interrupt handler address from vector-table offset 8. The CPU begins execution at the NMI interrupt handler.
8. If FlexNVM is enabled, the flash controller continues to restore the FlexNVM data. This data is not available immediately out of reset and the system should not access this data until the flash controller completes this initialization step as indicated by the EEERDY flag.

Subsequent system resets follow this reset flow beginning with the step where system clocks are enabled.

# Chapter 7
# Power Management

## 7.1 Introduction

This chapter describes the various chip power modes and functionality of the individual modules in these modes.

## 7.2 Power Modes Description

The power management controller (PMC) provides multiple power options to allow the user to optimize power consumption for the level of functionality needed.

Depending on the stop requirements of the user application, a variety of stop modes are available that provide state retention, partial power down or full power down of certain logic and/or memory. I/O states are held in all modes of operation. The following table compares the various power modes available.

For Run and VLPR mode there is a corresponding wait and stop mode. Wait modes are similar to ARM sleep modes. Stop modes (VLPS, STOP) are similar to ARM sleep deep mode. The very low power run (VLPR) operating mode can drastically reduce runtime power when the maximum bus frequency is not required to handle the application needs.

The three primary modes of operation are run, wait and stop. The WFI instruction invokes both wait and stop modes for the chip. The primary modes are augmented in a number of ways to provide lower power based on application needs.

**Table 7-1. Chip power modes**

| Chip mode | Description | Core mode | Normal recovery method |
|---|---|---|---|
| Normal run | Allows maximum performance of chip. Default mode out of reset; on-chip voltage regulator is on. | Run | - |

*Table continues on the next page...*

## Table 7-1. Chip power modes (continued)

| Chip mode | Description | Core mode | Normal recovery method |
|---|---|---|---|
| Normal Wait - via WFI | Allows peripherals to function while the core is in sleep mode, reducing power. NVIC remains sensitive to interrupts; peripherals continue to be clocked. | Sleep | Interrupt |
| Normal Stop - via WFI | Places chip in static state. Lowest power mode that retains all registers while maintaining LVD protection. NVIC is disabled; AWIC is used to wake up from interrupt; peripheral clocks are stopped. | Sleep Deep | Interrupt |
| VLPR (Very Low Power Run) | On-chip voltage regulator is in a low power mode that supplies only enough power to run the chip at a reduced frequency. Reduced frequency Flash access mode (0.8 MHz); LVD off; internal oscillator provides a low power 4 MHz source for the core, the bus and the peripheral clocks. | Run | Interrupt |
| VLPW (Very Low Power Wait) -via WFI | Same as VLPR but with the core in sleep mode to further reduce power; NVIC remains sensitive to interrupts (FCLK = ON). On-chip voltage regulator is in a low power mode that supplies only enough power to run the chip at a reduced frequency. | Sleep | Interrupt |
| VLPS (Very Low Power Stop)-via WFI | Places chip in static state with LVD operation off. Lowest power mode with ADC and pin interrupts functional. Peripheral clocks are stopped, but LPTimer, RTC, CMP, DAC can be used. NVIC is disabled (FCLK = OFF); AWIC is used to wake up from interrupt. On-chip voltage regulator is in a low power mode that supplies only enough power to run the chip at a reduced frequency. All SRAM is operating (content retained and I/O states held). | Sleep Deep | Interrupt |
| LLS (Low Leakage Stop) | State retention power mode. Most peripherals are in state retention mode (with clocks stopped), but LLWU, LPTimer, RTC, CMP, DAC can be used. NVIC is disabled; LLWU is used to wake up.<br>**NOTE:** The LLWU interrupt must not be masked by the interrupt controller to avoid a scenario where the system does not fully exit stop mode on an LLS recovery.<br>All SRAM is operating (content retained and I/O states held). | Sleep Deep | Wakeup Interrupt[1] |
| VLLS3 (Very Low Leakage Stop3) | Most peripherals are disabled (with clocks stopped), but LLWU, LPTimer, RTC, CMP, DAC can be used. NVIC is disabled; LLWU is used to wake up.<br>SRAM_U and SRAM_L remain powered on (content retained and I/O states held). | Sleep Deep | Wakeup Reset[2] |
| VLLS2 (Very Low Leakage Stop2) | Most peripherals are disabled (with clocks stopped), but LLWU, LPTimer, RTC, CMP, DAC can be used. NVIC is disabled; LLWU is used to wake up.<br>SRAM_L is powered off. A portion of SRAM_U remains powered on (content retained and I/O states held). | Sleep Deep | Wakeup Reset[2] |
| VLLS1 (Very Low Leakage Stop1) | Most peripherals are disabled (with clocks stopped), but LLWU, LPTimer, RTC, CMP, DAC can be used. NVIC is disabled; LLWU is used to wake up.<br>All of SRAM_U and SRAM_L are powered off. The 32-byte system register file and the 32-byteVBAT register file remain powered for customer-critical data. | Sleep Deep | Wakeup Reset[2] |

*Table continues on the next page...*

**Table 7-1. Chip power modes (continued)**

| Chip mode | Description | Core mode | Normal recovery method |
|---|---|---|---|
| VLLS0 (Very Low Leakage Stop 0) | Most peripherals are disabled (with clocks stopped), but LLWU and RTC can be used. NVIC is disabled; LLWU is used to wake up.<br>All of SRAM_U and SRAM_L are powered off. The 32-byte system register file and the 32-byte VBAT register file remain powered for customer-critical data.<br>The POR detect circuit can be optionally powered off. | Sleep Deep | Wakeup Reset[2] |
| BAT (backup battery only) | The chip is powered down except for the VBAT supply. The RTC and the 32-byte VBAT register file for customer-critical data remain powered. | Off | Power-up Sequence |

1. Resumes normal run mode operation by executing the LLWU interrupt service routine.
2. Follows the reset flow with the LLWU interrupt flag set for the NVIC.

## 7.3 Entering and exiting power modes

The WFI instruction invokes wait and stop modes for the chip. The processor exits the low-power mode via an interrupt. The Nested Vectored Interrupt Controller (NVIC) describes interrupt operation and what peripherals can cause interrupts.

**NOTE**

The WFE instruction can have the side effect of entering a low-power mode, but that is not its intended usage. See ARM documentation for more on the WFE instruction.

Recovery from VLLSx is through the wake-up Reset event. The chip wake-ups from VLLSx by means of reset, an enabled pin or enabled module. See the table "LLWU inputs" in the LLWU configuration section for a list of the sources.

The wake-up flow from VLLSx is through reset. The wakeup bit in the SRS registers in the RCM is set indicating that the chip is recovering from a low power mode. Code execution begins; however, the I/O pins are held in their pre low power mode entry states, and the system oscillator and MCG registers are reset (even if EREFSTEN had been set before entering VLLSx). Software must clear this hold by writing a 1 to the ACKISO bit in the Regulator Status and Control Register in the PMC module.

**NOTE**

To avoid unwanted transitions on the pins, software must re-initialize the I/O pins to their pre-low-power mode entry states *before* releasing the hold.

If the oscillator was configured to continue running during VLLSx modes, it must be re-configured before the ACKISO bit is cleared. The oscillator configuration within the MCG is cleared after VLLSx recovery and the oscillator will stop when ACKISO is cleared unless the register is re-configured.

## 7.4 Power mode transitions

The following figure shows the power mode transitions. Any reset always brings the chip back to the normal run state. In run, wait, and stop modes active power regulation is enabled. The VLPx modes offer a lower power operating mode than normal modes. VLPR and VLPW are limited in frequency. The LLS and VLLSx modes are the lowest power stop modes based on amount of logic or memory that is required to be retained by the application.

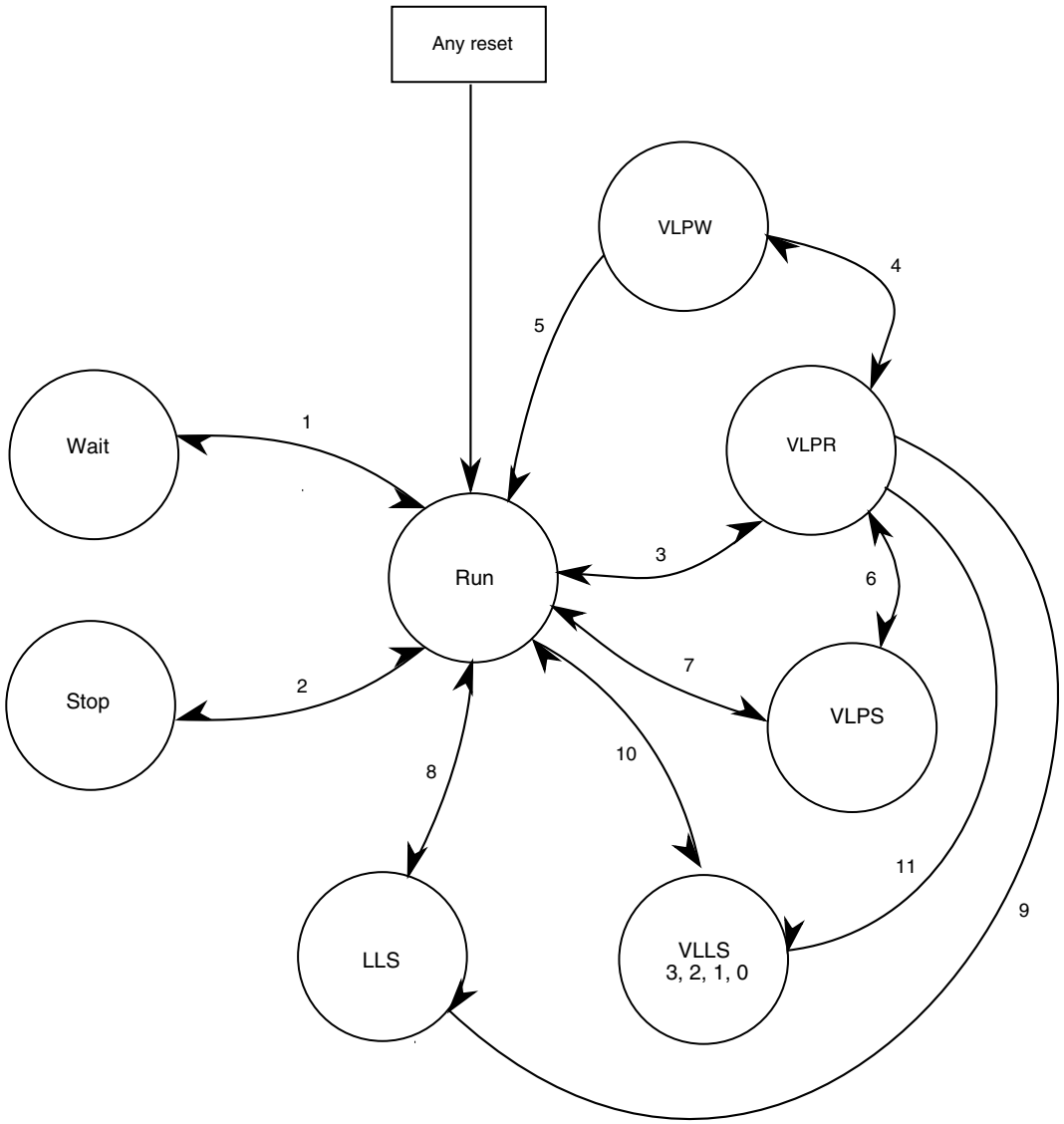



**Figure 7-1. Power mode state transition diagram**

## 7.5 Power modes shutdown sequencing

When entering stop or other low-power modes, the clocks are shut off in an orderly sequence to safely place the chip in the targeted low-power state. All low-power entry sequences are initiated by the core executing an WFI instruction. The ARM core's outputs, SLEEPDEEP and SLEEPING, trigger entry to the various low-power modes:

- System level wait and VLPW modes equate to: SLEEPING & $\overline{\text{SLEEPDEEP}}$
- All other low power modes equate to: SLEEPING & SLEEPDEEP

When entering the non-wait modes, the chip performs the following sequence:

- Shuts off Core Clock and System Clock to the ARM Cortex-M4 core immediately.
- Polls stop acknowledge indications from the non-core crossbar masters (DMA), supporting peripherals (SPI, PIT, RNG) and the Flash Controller for indications that System Clocks, Bus Clock and/or Flash Clock need to be left enabled to complete a previously initiated operation, effectively stalling entry to the targeted low power mode. When all acknowledges are detected, System Clock, Bus Clock and Flash Clock are turned off at the same time.
- MCG and Mode Controller shut off clock sources and/or the internal supplies driven from the on-chip regulator as defined for the targeted low power mode.

In wait modes, most of the system clocks are not affected by the low power mode entry. The Core Clock to the ARM Cortex-M4 core is shut off. Some modules support stop-in-wait functionality and have their clocks disabled under these configurations.

The debugger modules support a transition from stop, wait, VLPS, and VLPW back to a halted state when the debugger is enabled. This transition is initiated by setting the Debug Request bit in MDM-AP control register. As part of this transition, system clocking is re-established and is equivalent to normal run/VLPR mode clocking configuration.

## 7.6 Module Operation in Low Power Modes

The following table illustrates the functionality of each module while the chip is in each of the low power modes. (Debug modules are discussed separately; see Debug in Low Power Modes.) Number ratings (such as 2 MHz and 1 Mbit/s) represent the maximum frequencies or maximum data rates per mode. Also, these terms are used:

- FF = Full functionality. In VLPR and VLPW the system frequency is limited, but if a module does not have a limitation in its functionality, it is still listed as FF.
- static = Module register states and associated memories are retained.
- powered = Memory is powered to retain contents.
- low power = Flash has a low power state that retains configuration registers to support faster wakeup.
- OFF = Modules are powered off; module is in reset state upon wakeup.
- wakeup = Modules can serve as a wakeup source for the chip.

**Table 7-2. Module operation in low power modes**

| Modules | Stop | VLPR | VLPW | VLPS | LLS | VLLSx |
|---|---|---|---|---|---|---|
| **Core modules** | | | | | | |
| NVIC | static | FF | FF | static | static | OFF |
| **System modules** | | | | | | |
| Mode Controller | FF | FF | FF | FF | FF | FF |

*Table continues on the next page...*

## Table 7-2. Module operation in low power modes (continued)

| Modules | Stop | VLPR | VLPW | VLPS | LLS | VLLSx |
|---|---|---|---|---|---|---|
| LLWU[1] | static | static | static | static | FF | FF[2] |
| Regulator | ON | low power | low power | low power | low power | low power in VLLS2/3, OFF in VLLS0/1 |
| LVD | ON | disabled | disabled | disabled | disabled | disabled |
| Brown-out Detection | ON | ON | ON | ON | ON | ON in VLLS1/2/3, optionally disabled in VLLS0[3] |
| DMA | static | FF | FF | static | static | OFF |
| Watchdog | FF | FF | FF | FF | static | OFF |
| EWM | static | FF | static | static | static | OFF |
| **Clocks** | | | | | | |
| 1kHz LPO | ON | ON | ON | ON | ON | ON in VLLS1/2/3, OFF in VLLS0 |
| System oscillator (OSC) | OSCERCLK optional | OSCERCLK max of 4 MHz crystal | OSCERCLK max of 4 MHz crystal | OSCERCLK max of 4 MHz crystal | limited to low range/low power | limited to low range/low power in VLLS1/2/3, OFF in VLLS0 |
| MCG | static - MCGIRCLK optional; PLL optionally on but gated | MHz IRC | MHz IRC | static - no clock output | static - no clock output | OFF |
| Core clock | OFF | MHz max | OFF | OFF | OFF | OFF |
| System clock | OFF | MHz max | MHz max | OFF | OFF | OFF |
| Bus clock | OFF | MHz max | MHz max | OFF | OFF | OFF |
| **Memory and memory interfaces** | | | | | | |
| Flash | powered | 0.8 MHz max access - no pgm | low power | low power | OFF | OFF |
| Portion of SRAM_U[4] | low power | low power | low power | low power | low power | low power in VLLS3,2; otherwise OFF |
| Remaining SRAM_U and all of SRAM_L | low power | low power | low power | low power | low power | low power in VLLS3; otherwise OFF |
| FlexMemory | low power | low power[5] | low power | low power | low power | OFF |
| VBAT Register file[6] | powered | powered | powered | powered | powered | powered |
| System Register files | powered | powered | powered | powered | powered | powered |
| FlexBus | static | FF | FF | static | static | OFF |
| EzPort | disabled | disabled | disabled | disabled | disabled | disabled |
| **Communication interfaces** | | | | | | |

*Table continues on the next page...*

## Table 7-2. Module operation in low power modes (continued)

| Modules | Stop | VLPR | VLPW | VLPS | LLS | VLLSx |
|---|---|---|---|---|---|---|
| USB FS/LS | static | static | static | static | static | OFF |
| USB DCD | static | FF | FF | static | static | OFF |
| USB Voltage Regulator | optional | optional | optional | optional | optional | optional |
| UART | static, wakeup on edge | 125 kbit/s | 125 kbit/s | static, wakeup on edge | static | OFF |
| SPI | static | 1 Mbit/s | 1 Mbit/s | static | static | OFF |
| $I^2C$ | static, address match wakeup | 100 kbit/s | 100 kbit/s | static, address match wakeup | static | OFF |
| CAN | wakeup | 256 kbit/s | 256 kbit/s | wakeup | static | OFF |
| $I^2S$ | FF with external clock[7] | FF | FF | FF with external clock[7] | static | OFF |
| SDHC | wakeup | FF | FF | wakeup | static | OFF |
| **Security** | | | | | | |
| CRC | static | FF | FF | static | static | OFF |
| RNG | static | FF | static | static | static | OFF |
| DryIce[6] | FF | FF | FF | FF | FF | FF |
| **Timers** | | | | | | |
| FTM | static | FF | FF | static | static | OFF |
| PIT | static | FF | FF | static | static | OFF |
| PDB | static | FF | FF | static | static | OFF |
| LPTMR | FF | FF | FF | FF | FF | FF[8] |
| RTC - 32kHz OSC[6] | FF | FF | FF | FF | FF[9] | FF[9] |
| CMT | static | FF | FF | static | static | OFF |
| **Analog** | | | | | | |
| 16-bit ADC | ADC internal clock only | FF | FF | ADC internal clock only | static | OFF |
| CMP[10] | HS or LS level compare | FF | FF | HS or LS level compare | LS level compare | LS level compare in VLLS1/2/3, OFF in VLLS0 |
| 6-bit DAC | static | FF | FF | static | static | static |
| VREF | FF | FF | FF | FF | static | OFF |
| 12-bit DAC | static | FF | FF | static | static | static |
| **Human-machine interfaces** | | | | | | |
| GPIO | wakeup | FF | FF | wakeup | static, pins latched | OFF, pins latched |

1. Using the LLWU module, the external pins available for this chip do not require the associated peripheral function to be enabled. It only requires the function controlling the pin (GPIO or peripheral) to be configured as an input to allow a transition to occur to the LLWU.
2. Since LPO clock source is disabled, filters will be bypassed during VLLS0
3. The VLLSCTRL[PORPO] bit in the SMC module controls this option.
4. A KB portion of SRAM_U block is left powered on in low power mode VLLS2.

5. FlexRAM enabled as EEPROM is not writable in VLPR and writes are ignored. Read accesses to FlexRAM as EEPROM while in VLPR are allowed. There are no access restrictions for FlexRAM configured as traditional RAM.
6. These components remain powered in BAT power mode.
7. Use an externally generated bit clock or an externally generated audio master clock (including EXTAL).
8. System OSC and LPO clock sources are not available in VLLS0
9. RTC_CLKOUT is not available.
10. CMP in stop or VLPS supports high speed or low speed external pin to pin or external pin to DAC compares. CMP in LLS or VLLSx only supports low speed external pin to pin or external pin to DAC compares. Windowed, sampled & filtered modes of operation are not available while in stop, VLPS, LLS, or VLLSx modes.

## 7.7 Clock Gating

To conserve power, the clocks to most modules can be turned off using the SCGCx registers in the SIM module. These bits are cleared after any reset, which disables the clock to the corresponding module. Prior to initializing a module, set the corresponding bit in the SCGCx register to enable the clock. Before turning off the clock, make sure to disable the module. For more details, refer to the clock distribution and SIM chapters.

# Chapter 8
# Security

## 8.1 Introduction

This device implements security based on the mode selected from the flash module. The following sections provide an overview of flash security and details the effects of security on non-flash modules.

## 8.2 Flash Security

The flash module provides security information to the MCU based on the state held by the FSEC[SEC] bits. The MCU, in turn, confirms the security request and limits access to flash resources. During reset, the flash module initializes the FSEC register using data read from the security byte of the flash configuration field.

**NOTE**

The security features apply only to external accesses via debug and EzPort. CPU accesses to the flash are not affected by the status of FSEC.

In the unsecured state all flash commands are available to the programming interfaces (JTAG and EzPort), as well as user code execution of Flash Controller commands. When the flash is secured (FSEC[SEC] = 00, 01, or 11), programmer interfaces are only allowed to launch mass erase operations and have no access to memory locations.

Further information regarding the flash security options and enabling/disabling flash security is available in the Flash Memory Module.

## 8.3 Security Interactions with other Modules

The flash security settings are used by the SoC to determine what resources are available. The following sections describe the interactions between modules and the flash security settings or the impact that the flash security has on non-flash modules.

### 8.3.1 Security interactions with FlexBus

When flash security is enabled, SIM_SOPT2[FBSL] enables/disables off-chip accesses through the FlexBus interface. The FBSL bitfield also has an option to allow opcode and operand accesses or only operand accesses.

### 8.3.2 Security Interactions with EzPort

When flash security is active the MCU can still boot in EzPort mode. The EzPort holds the flash logic in NVM special mode and thus limits flash operation when flash security is active. While in EzPort mode and security is active, flash bulk erase (BE) can still be executed. The write FCCOB registers (WRFCCOB) command is limited to the mass erase (Erase All Blocks) and verify all 1s (Read 1s All Blocks) commands. Read accesses to internal memories via the EzPort are blocked when security is enabled.

The mass erase can be used to disable flash security, but all of the flash contents are lost in the process. A mass erase via the EzPort is allowed even when some memory locations are protected.

When mass erase has been disabled, mass erase via the EzPort is blocked and cannot be defeated.

### 8.3.3 Security Interactions with Debug

When flash security is active the JTAG port cannot access the memory resources of the MCU. Boundary scan chain operations work, but debugging capabilities are disabled so that the debug port cannot read flash contents.

Although most debug functions are disabled, the debugger can write to the Flash Mass Erase in Progress bit in the MDM-AP Control register to trigger a mass erase (Erase All Blocks) command. A mass erase via the debugger is allowed even when some memory locations are protected.

When mass erase is disabled, mass erase via the debugger is blocked.

# Chapter 9
# Debug

## 9.1 Introduction

This device's debug is based on the ARM coresight architecture and is configured in each device to provide the maximum flexibility as allowed by the restrictions of the pinout and other available resources.

Four debug interfaces are supported:

- IEEE 1149.1 JTAG
- IEEE 1149.7 JTAG (cJTAG)
- Serial Wire Debug (SWD)
- ARM Real-Time Trace Interface

The basic Cortex-M4 debug architecture is very flexible. The following diagram shows the topology of the core debug architecture and its components.

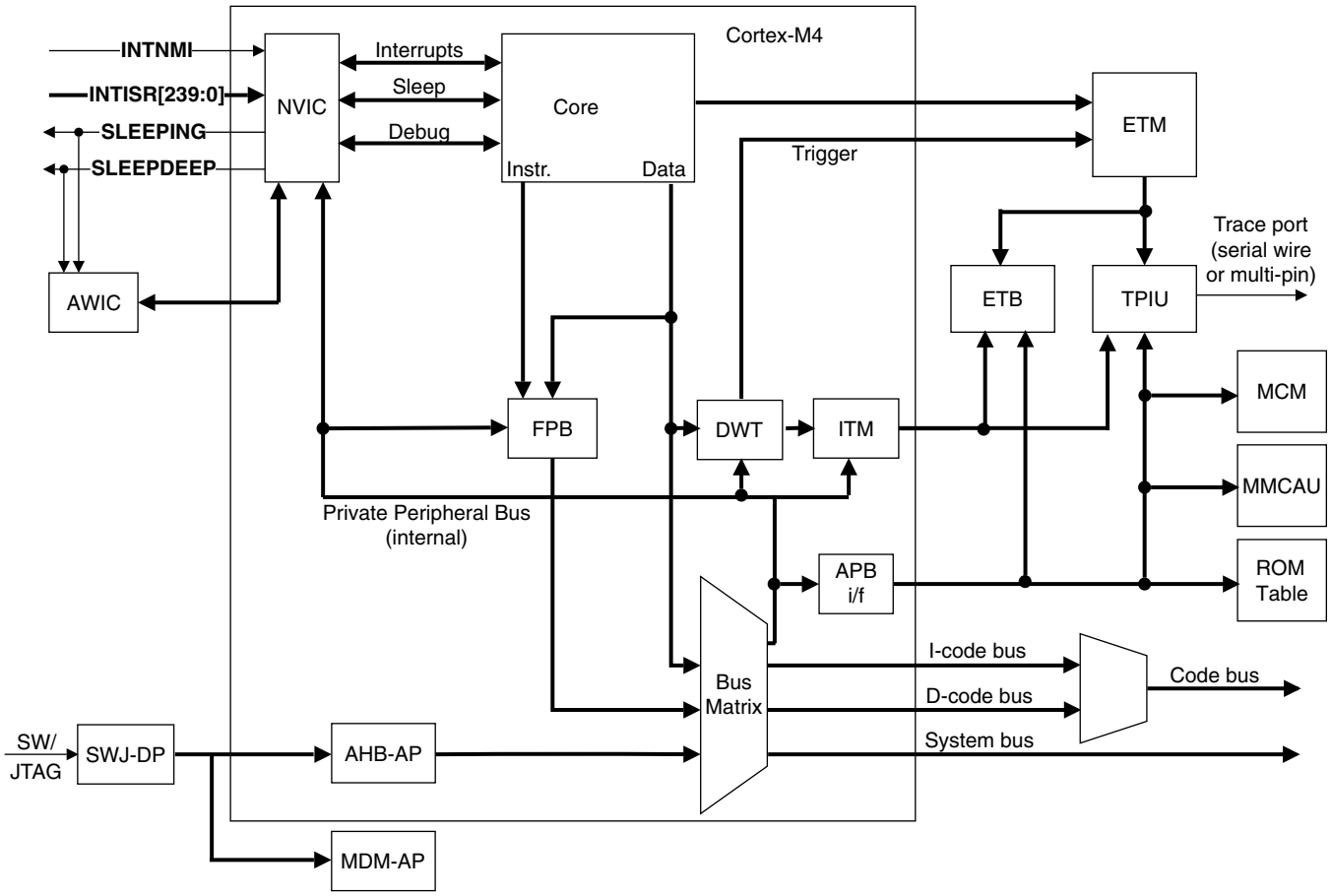



**Figure 9-1. Cortex-M4 Debug Topology**

The following table presents a brief description of each one of the debug components.

**Table 9-1. Debug Components Description**

| Module | Description |
|---|---|
| SWJ-DP+ cJTAG | Modified Debug Port with support for SWD, JTAG, cJTAG |
| AHB-AP | AHB Master Interface from JTAG to debug module and SOC system memory maps |
| MDM-AP | Provides centralized control and status registers for an external debugger to control the device. |
| ROM Table | Identifies which debug IP is available. |
| Core Debug | Singlestep, Register Access, Run, Core Status |
| CoreSight Trace Funnel (not shown in figure) | The CSTF combines multiple trace streams onto a single ATB bus. |
| CoreSight Trace Replicator (not shown in figure) | The ATB replicator enables two trace sinks to be wired together and operate from the same incoming trace stream. |
| ETM (Embedded Trace Macrocell) | ETMv3.5 Architecture |
| CoreSight ETB (Embedded Trace Buffer) | Memory mapped buffer used to store trace data. |
| ITM | S/W Instrumentation Messaging + Simple Data Trace Messaging + Watchpoint Messaging |

*Table continues on the next page...*

**Table 9-1. Debug Components Description (continued)**

| Module | Description |
|---|---|
| DWT (Data and Address Watchpoints) | 4 data and address watchpoints |
| FPB (Flash Patch and Breakpoints) | The FPB implements hardware breakpoints and patches code and data from code space to system space.<br>The FPB unit contains two literal comparators for matching against literal loads from Code space, and remapping to a corresponding area in System space.<br>The FBP also contains six instruction comparators for matching against instruction fetches from Code space, and remapping to a corresponding area in System space. Alternatively, the six instruction comparators can individually configure the comparators to return a Breakpoint Instruction (BKPT) to the processor core on a match, so providing hardware breakpoint capability. |
| TPIU (Trace Port Inteface Unit) | Asynchronous Mode (1-pin) = TRACE_SWO (available on JTAG_TDO) |
| MCM (Miscellaneous Control Module) | The MCM provides miscellaneous control functions including control of the ETB and trace path switching. |

### 9.1.1 References

For more information on ARM debug components, see these documents:

- ARMv7-M Architecture Reference Manual
- ARM Debug Interface v5.1
- ARM CoreSight Architecture Specification
- ARM ETM Architecture Specification v3.5

## 9.2 The Debug Port

The configuration of the cJTAG module, JTAG controller, and debug port is illustrated in the following figure:

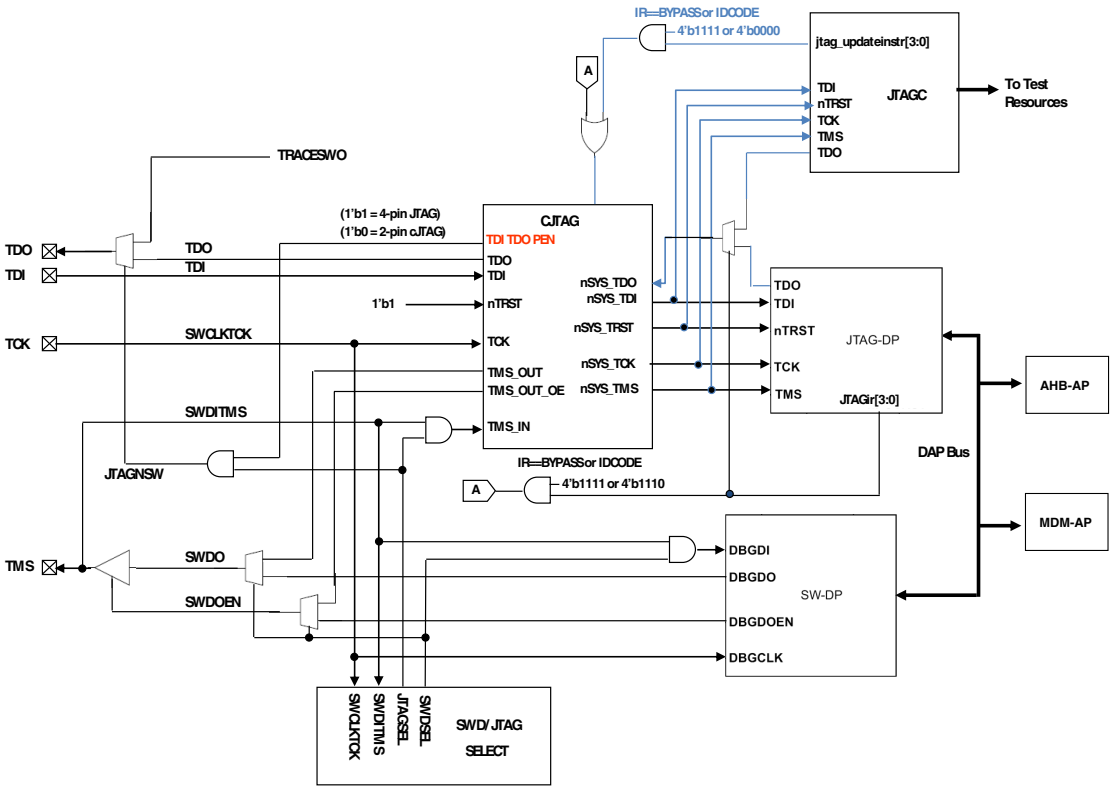



**Figure 9-2. Modified Debug Port**

The debug port comes out of reset in standard JTAG mode and is switched into either cJTAG or SWD mode by the following sequences. Once the mode has been changed, unused debug pins can be reassigned to any of their alternative muxed functions.

## 9.2.1 JTAG-to-SWD change sequence

1. Send more than 50 TCK cycles with TMS (SWDIO) =1
2. Send the 16-bit sequence on TMS (SWDIO) = 0111_1001_1110_0111 (MSB transmitted first)
3. Send more than 50 TCK cycles with TMS (SWDIO) =1

**NOTE**

See the ARM documentation for the CoreSight DAP Lite for restrictions.

## 9.2.2 JTAG-to-cJTAG change sequence

1. Reset the debug port

2. Set the control level to 2 via zero-bit scans
3. Execute the Store Format (STFMT) command (00011) to set the scan format register to 1149.7 scan format

## 9.3 Debug Port Pin Descriptions

The debug port pins default after POR to their JTAG functionality with the exception of JTAG_TRST_b and can be later reassigned to their alternate functionalities. In cJTAG and SWD modes JTAG_TDI and JTAG_TRST_b can be configured to alternate GPIO functions.

**Table 9-2. Debug port pins**

| Pin Name | JTAG Debug Port | | cJTAG Debug Port | | SWD Debug Port | | Internal Pull-up\Down |
|---|---|---|---|---|---|---|---|
| | Type | Description | Type | Description | Type | Description | |
| JTAG_TMS/ SWD_DIO | I/O | JTAG Test Mode Selection | I/O | cJTAG Data | I/O | Serial Wire Data | Pull-up |
| JTAG_TCLK/ SWD_CLK | I | JTAG Test Clock | I | cJTAG Clock | I | Serial Wire Clock | Pull-down |
| JTAG_TDI | I | JTAG Test Data Input | - | - | - | - | Pull-up |
| JTAG_TDO/ TRACE_SWO | O | JTAG Test Data Output | O | Trace output over a single pin | O | Trace output over a single pin | N/C |
| JTAG_TRST_b | I | JTAG Reset | I | cJTAG Reset | - | - | Pull-up |

## 9.4 System TAP connection

The system JTAG controller is connected in parallel to the ARM TAP controller. The system JTAG controller IR codes overlay the ARM JTAG controller IR codes without conflict. Refer to the IR codes table for a list of the available IR codes. The output of the TAPs (TDO) are muxed based on the IR code which is selected. This design is fully JTAG compliant and appears to the JTAG chain as a single TAP. At power on reset, ARM's IDCODE (IR=4'b1110) is selected.

### 9.4.1 IR Codes

**Table 9-3. JTAG Instructions**

| Instruction | Code[3:0] | Instruction Summary |
|---|---|---|
| IDCODE | 0000 | Selects device identification register for shift |
| SAMPLE/PRELOAD | 0010 | Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation |
| SAMPLE | 0011 | Selects boundary scan register for shifting and sampling without disturbing functional operation |
| EXTEST | 0100 | Selects boundary scan register while applying preloaded values to output pins and asserting functional reset |
| HIGHZ | 1001 | Selects bypass register while three-stating all output pins and asserting functional reset |
| CLAMP | 1100 | Selects bypass register while applying preloaded values to output pins and asserting functional reset |
| EZPORT | 1101 | Enables the EZPORT function for the SoC and asserts functional reset. |
| ARM_IDCODE | 1110 | ARM JTAG-DP Instruction |
| BYPASS | 1111 | Selects bypass register for data operations |
| Factory debug reserved | 0101, 0110, 0111 | Intended for factory debug only |
| ARM JTAG-DP Reserved | 1000, 1010, 1011, 1110 | These instructions will go the ARM JTAG-DP controller. Please look at ARM JTAG-DP documentation for more information on these instructions. |
| Reserved [1] | All other opcodes | Decoded to select bypass register |

1. The manufacturer reserves the right to change the decoding of reserved instruction codes in the future

## 9.5 JTAG status and control registers

Through the ARM Debug Access Port (DAP), the debugger has access to the status and control elements, implemented as registers on the DAP bus as shown in the following figure. These registers provide additional control and status for low power mode recovery and typical run-control scenarios. The status register bits also provide a means for the debugger to get updated status of the core without having to initiate a bus transaction across the crossbar switch, thus remaining less intrusive during a debug session.

It is important to note that these DAP control and status registers are not memory mapped within the system memory map and are only accessible via the Debug Access Port (DAP) using JTAG, cJTAG, or SWD. The MDM-AP is accessible as Debug Access Port 1 with the available registers shown in the table below.

**Table 9-4. MDM-AP Register Summary**

| Address | Register | Description |
|---|---|---|

*Table continues on the next page...*

**Table 9-4. MDM-AP Register Summary (continued)**

| | | |
|---|---|---|
| 0x0100_0000 | Status | See MDM-AP Status Register |
| 0x0100_0004 | Control | See MDM-AP Control Register |
| 0x0100_00FC | ID | Read-only identification register that always reads as 0x001C_0000 |

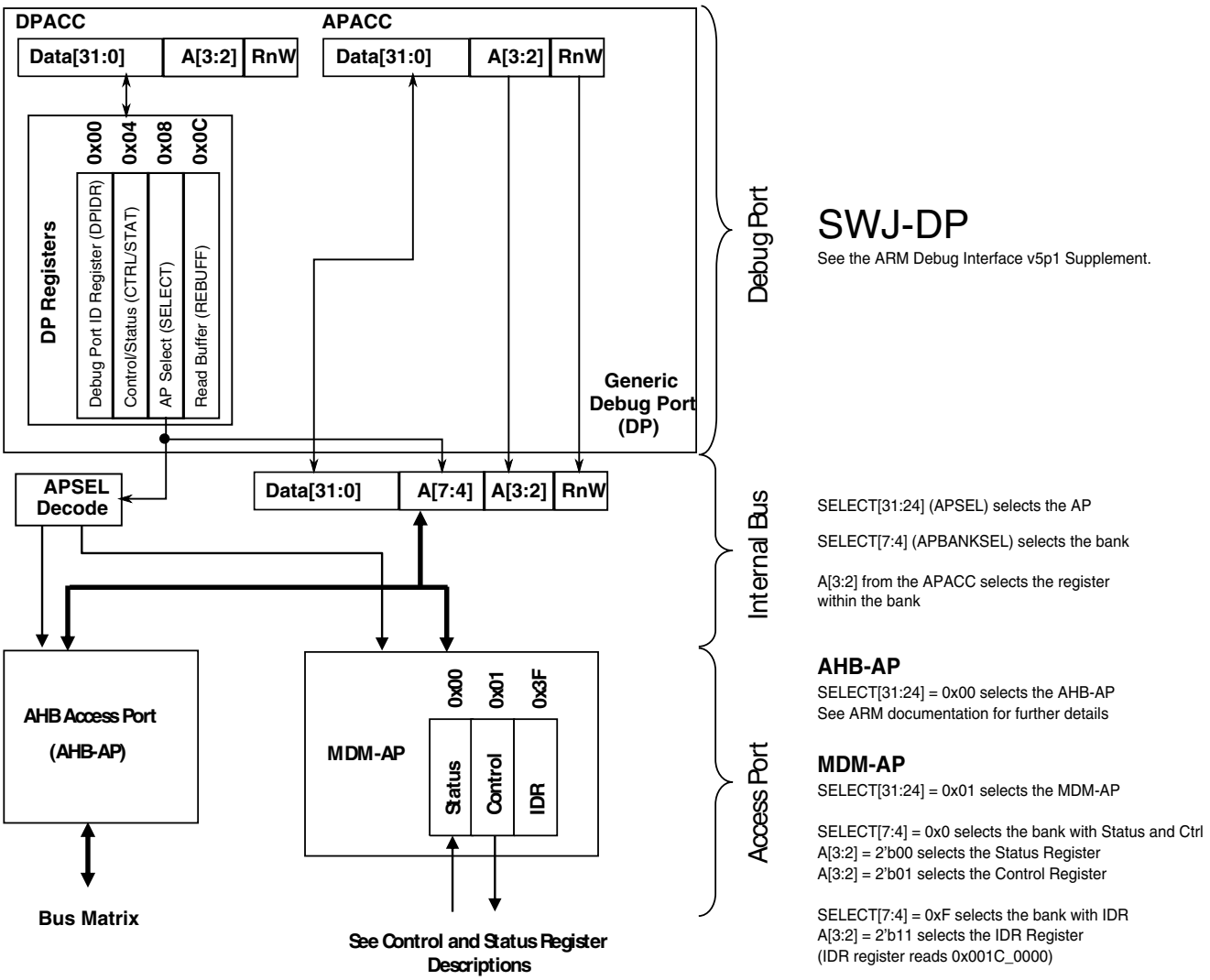



**Figure 9-3. MDM AP Addressing**

## 9.5.1 MDM-AP Control Register

**Table 9-5. MDM-AP Control register assignments**

| Bit | Name | Secure[1] | Description |
|---|---|---|---|
| 0 | Flash Mass Erase in Progress | Y | Set to cause mass erase. Cleared by hardware after mass erase operation completes.<br>When mass erase is disabled (via MEEN and SEC settings), the erase request does not occur and the Flash Mass Erase in Progress bit continues to assert until the next system reset. |
| 1 | Debug Disable | N | Set to disable debug. Clear to allow debug operation. When set it overrides the C_DEBUGEN bit within the DHCSR and force disables Debug logic. |
| 2 | Debug Request | N | Set to force the Core to halt.<br>If the Core is in a stop or wait mode, this bit can be used to wakeup the core and transition to a halted state. |
| 3 | System Reset Request | N | Set to force a system reset. The system remains held in reset until this bit is cleared. |
| 4 | Core Hold Reset | N | Configuration bit to control Core operation at the end of system reset sequencing.<br>0 Normal operation - release the Core from reset along with the rest of the system at the end of system reset sequencing.<br>1 Suspend operation - hold the Core in reset at the end of reset sequencing. Once the system enters this suspended state, clearing this control bit immediately releases the Core from reset and CPU operation begins. |
| 5 | VLLSx Debug Request (VLLDBGREQ) | N | Set to configure the system to be held in reset after the next recovery from a VLLSx mode.<br>This bit holds the in reset when VLLSx modes are exited to allow the debugger time to re-initialize debug IP before the debug session continues.<br>The Mode Controller captures this bit logic on entry to VLLSx modes. Upon exit from VLLSx modes, the Mode Controller will hold the in reset until VLLDBGACK is asserted.<br>The VLLDBGREQ bit clears automatically due to the POR reset generated as part of the VLLSx recovery. |
| 6 | VLLSx Debug Acknowledge (VLLDBGACK) | N | Set to release a being held in reset following a VLLSx recovery<br>This bit is used by the debugger to release the system reset when it is being held on VLLSx mode exit. The debugger re-initializes all debug IP and then assert this control bit to allow the Mode Controller to release the from reset and allow CPU operation to begin.<br>The VLLDBGACK bit is cleared by the debugger or can be left set because it clears automatically due to the POR reset generated as part of the next VLLSx recovery. |
| 7 | LLS, VLLSx Status Acknowledge | N | Set this bit to acknowledge the DAP LLS and VLLS Status bits have been read. This acknowledge automatically clears the status bits.<br>This bit is used by the debugger to clear the sticky LLS and VLLSx mode entry status bits. This bit is asserted and cleared by the debugger. |

*Table continues on the next page...*

**Table 9-5. MDM-AP Control register assignments (continued)**

| Bit | Name | Secure[1] | Description |
|---|---|---|---|
| 8 | Timestamp Disable | N | Set this bit to disable the 48-bit global trace timestamp counter during debug halt mode when the core is halted.<br>0 The timestamp counter continues to count assuming trace is enabled and the ETM is enabled. (default)<br>1 The timestamp counter freezes when the core has halted (debug halt mode). |
| 9 – 31 | Reserved for future use | N | |

1. Command available in secure mode

## 9.5.2 MDM-AP Status Register

**Table 9-6. MDM-AP Status register assignments**

| Bit | Name | Description |
|---|---|---|
| 0 | Flash Mass Erase Acknowledge | The Flash Mass Erase Acknowledge bit is cleared after any system reset. The bit is also cleared at launch of a mass erase command due to write of Flash Mass Erase in Progress bit in MDM AP Control Register. The Flash Mass Erase Acknowledge is set after Flash control logic has started the mass erase operation.<br>When mass erase is disabled (via MEEN and SEC settings), an erase request due to seting of Flash Mass Erase in Progress bit is not acknowledged. |
| 1 | Flash Ready | Indicate Flash has been initialized and debugger can be configured even if system is continuing to be held in reset via the debugger. |
| 2 | System Security | Indicates the security state. When secure, the debugger does not have access to the system bus or any memory mapped peripherals. This bit indicates when the part is locked and no system bus access is possible. |
| 3 | System Reset | Indicates the system reset state.<br>0 System is in reset<br>1 System is not in reset |
| 4 | Reserved | |
| 5 | Mass Erase Enable | Indicates if the MCU can be mass erased or not<br>0 Mass erase is disabled<br>1 Mass erase is enabled |
| 6 | Backdoor Access Key Enable | Indicates if the MCU has the backdoor access key enabled.<br>0 Disabled<br>1 Enabled |

*Table continues on the next page...*

**Table 9-6. MDM-AP Status register assignments (continued)**

| Bit | Name | Description |
|---|---|---|
| 7 | LP Enabled | Decode of LPLLSM control bits to indicate that VLPS, LLS, or VLLSx are the selected power mode the next time the ARM Core enters Deep Sleep.<br>0 Low Power Stop Mode is not enabled<br>1 Low Power Stop Mode is enabled<br>Usage intended for debug operation in which Run to VLPS is attempted. Per debug definition, the system actually enters the Stop state. A debugger should interpret deep sleep indication (with SLEEPDEEP and SLEEPING asserted), in conjuntion with this bit asserted as the debugger-VLPS status indication. |
| 8 | Very Low Power Mode | Indicates current power mode is VLPx. This bit is not 'sticky' and should always represent whether VLPx is enabled or not.<br>This bit is used to throttle JTAG TCK frequency up/down. |
| 9 | LLS Mode Exit | This bit indicates an exit from LLS mode has occurred. The debugger will lose communication while the system is in LLS (including access to this register). Once communication is reestablished, this bit indicates that the system had been in LLS. Since the debug modules held their state during LLS, they do not need to be reconfigured.<br>This bit is set during the LLS recovery sequence. The LLS Mode Exit bit is held until the debugger has had a chance to recognize that LLS was exited and is cleared by a write of 1 to the LLS, VLLSx Status Acknowledge bit in MDM AP Control register. |
| 10 | VLLSx Modes Exit | This bit indicates an exit from VLLSx mode has occurred. The debugger will lose communication while the system is in VLLSx (including access to this register). Once communication is reestablished, this bit indicates that the system had been in VLLSx. Since the debug modules lose their state during VLLSx modes, they need to be reconfigured.<br>This bit is set during the VLLSx recovery sequence. The VLLSx Mode Exit bit is held until the debugger has had a chance to recognize that a VLLS mode was exited and is cleared by a write of 1 to the LLS, VLLSx Status Acknowledge bit in MDM AP Control register. |
| 11 – 15 | Reserved for future use | Always read 0. |
| 16 | Core Halted | Indicates the Core has entered debug halt mode |
| 17 | Core SLEEPDEEP | Indicates the Core has entered a low power mode |
| 18 | Core SLEEPING | SLEEPING==1 and SLEEPDEEP==0 indicates wait or VLPW mode.<br>SLEEPING==1 and SLEEPDEEP==1 indicates stop or VLPS mode. |
| 19 – 31 | Reserved for future use | Always read 0. |

## 9.6 Debug Resets

The debug system receives the following sources of reset:

- JTAG_TRST_b from an external signal. This signal is optional and may not be available in all packages.

- Debug reset (CDBGRSTREQ bit within the SWJ-DP CTRL/STAT register) in the TCLK domain that allows the debugger to reset the debug logic.
- TRST asserted via the cJTAG escape command.
- System POR reset

Conversely the debug system is capable of generating system reset using the following mechanism:

- A system reset in the DAP control register which allows the debugger to hold the system in reset.
- SYSRESETREQ bit in the NVIC application interrupt and reset control register
- A system reset in the DAP control register which allows the debugger to hold the Core in reset.

## 9.7 AHB-AP

AHB-AP provides the debugger access to all memory and registers in the system, including processor registers through the NVIC. System access is independent of the processor status. AHB-AP does not do back-to-back transactions on the bus, so all transactions are non-sequential. AHB-AP can perform unaligned and bit-band transactions. AHB-AP transactions bypass the FPB, so the FPB cannot remap AHB-AP transactions. SWJ/SW-DP-initiated transaction aborts drive an AHB-AP-supported sideband signal called HABORT. This signal is driven into the Bus Matrix, which resets the Bus Matrix state, so that AHB-AP can access the Private Peripheral Bus for last ditch debugging such as read/stop/reset the core. AHB-AP transactions are little endian.

The MPU includes default settings and protections for the Region Descriptor 0 (RGD0) such that the Debugger always has access to the entire address space and those rights cannot be changed by the core or any other bus master.

For a short period at the start of a system reset event the system security status is being determined and debugger access to all AHB-AP transactions is blocked. The MDM-AP Status register is accessible and can be monitored to determine when this initial period is completed. After this initial period, if system reset is held via assertion of the RESET pin, the debugger has access via the bus matrix to the private peripheral bus to configure the debug IP even while system reset is asserted. While in system reset, access to other memory and register resources, accessed over the Crossbar Switch, is blocked.

## 9.8 ITM

The ITM is an application-driven trace source that supports printf style debugging to trace Operating System (OS) and application events, and emits diagnostic system information. The ITM emits trace information as packets. There are four sources that can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The four sources in decreasing order of priority are:

1. Software trace -- Software can write directly to ITM stimulus registers. This emits packets.
2. Hardware trace -- The DWT generates these packets, and the ITM emits them.
3. Time stamping -- Timestamps are emitted relative to packets. The ITM contains a 21-bit counter to generate the timestamp. The Cortex-M4 clock or the bitclock rate of the Serial Wire Viewer (SWV) output clocks the counter.
4. Global system timestamping. Timestamps can optionally be generated using a system-wide 48-bit count value. The same count value can be used to insert timestamps in the ETM trace stream, allowing coarse-grain correlation.

## 9.9 Core Trace Connectivity

The ETM and ITM can route its data to the ETB or the TPIU. (See the MCM (Miscellaneous Control Module) for controlling the routing to the TPIU.) This configuration enables the use of trace with low cost tools while maintaining the compatibility with trace probes. The arbitration between the ETM and ITM is performed inside the TPIU.

## 9.10 Embedded Trace Macrocell v3.5 (ETM)

The Cortex-M4 Embedded Trace Macrocell (ETM-M4) is a debug component that enables a debugger to reconstruct program execution. The CoreSight ETM-M4 supports only instruction trace. You can use it either with the Cortex-M4 Trace Port Interface Unit (M4-TPIU), or with the CoreSight ETB.

The main features of an ETM are:

- tracing of 16-bit and 32-bit Thumb instructions
- four EmbeddedICE watchpoint inputs
- a Trace Start/Stop block with EmbeddedICE inputs

- one reduced function counter
- two external inputs
- a 24-byte FIFO queue
- global timestamping

## 9.11 Coresight Embedded Trace Buffer (ETB)

The ETB provides on-chip storage of trace data using 32-bit RAM. The ETB accepts trace data from any CoreSight-compliant component trace source with an ATB master port, such as a trace source or a trace funnel. It is included in this device to remove dependencies from the trace pin pad speed, and enable low cost trace solutions. The TraceRAM size is 2 KB.

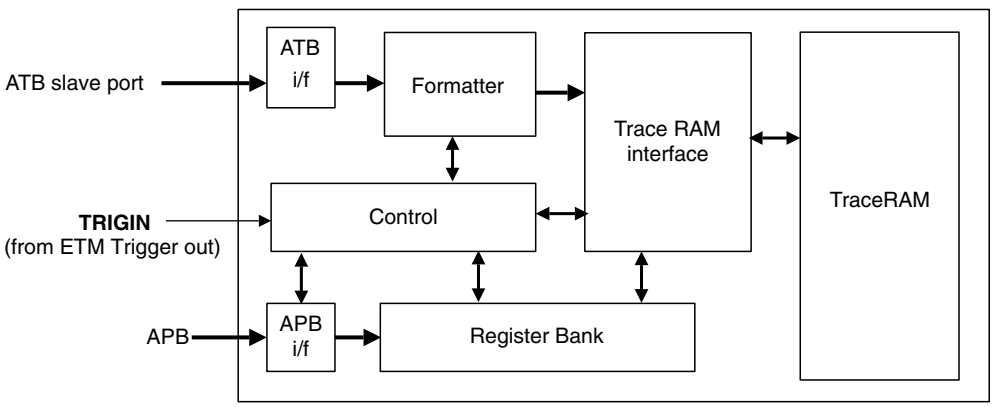



**Figure 9-4. ETB Block Diagram**

The ETB contains the following blocks:

- Formatter -- Inserts source ID signals into the data packet stream so that trace data can be re-associated with its trace source after the data is read back out of the ETB.
- Control -- Control registers for trace capture and flushing.
- APB interface -- Read, write, and data pointers provide access to ETB registers. In addition, the APB interface supports wait states through the use of a PREADYDBG signal output by the ETB. The APB interface is synchronous to the ATB domain.
- Register bank -- Contains the management, control, and status registers for triggers, flushing behavior, and external control.
- Trace RAM interface -- Controls reads and writes to the Trace RAM.

### 9.11.1 Performance Profiling with the ETB

To create a performance profile (e.g. gprof) for the target application, a means to collect trace over a long period of time is needed. The ETB buffer is too small to capture a meaningful profile in just one take. What is needed is to collect and concatenate data from the ETB buffer for multiple sequential runs. Using the ETB packet counter (described in Miscellaneous Control Module (MCM)), the trace analysis tool can capture multiple sequential runs by executing code until the ETB is almost full, and halting or executing an interrupt handler to allow the buffer to be emptied, and then continuing executing code. The target halts or executes an interrupt handler when the buffer is almost full to empty the data and then the debugger runs the target again.

### 9.11.2 ETB Counter Control

The ETB packet counter is controlled by the ETB counter control register, ETB reload register, and ETB counter value register implemented in the Miscellaneous Control Module (MCM) accessible via the Private Peripheral Bus. Via the ETB counter control register the ETB control logic can be configured to cause an MCM Alert Interrupt, an NMI Interrupt, or cause a Debug halt when the down counter reaches 0. Other features of the ETB control logic include:

- Down counter to count as many as 512 x 32-bit packets.
- Reload request transfers reload value to counter.
- ATB valid and ready signals used to form counter decrement.
- The counter disarms itself when the count reaches 0.

## 9.12 TPIU

The TPIU acts as a bridge between the on-chip trace data from the Embedded Trace Macrocell (ETM) and the Instrumentation Trace Macrocell (ITM), with separate IDs, to a data stream, encapsulating IDs where required, that is then captured by a Trace Port Analyzer (TPA). The TPIU is specially designed for low-cost debug.

## 9.13 DWT

The DWT is a unit that performs the following debug functionality:

- It contains four comparators that you can configure as a hardware watchpoint, an ETM trigger, a PC sampler event trigger, or a data address sampler event trigger. The first comparator, DWT_COMP0, can also compare against the clock cycle counter, CYCCNT. The second comparator, DWT_COMP1, can also be used as a data comparator.
- The DWT contains counters for:
  - Clock cycles (CYCCNT)
  - Folded instructions
  - Load store unit (LSU) operations
  - Sleep cycles
  - CPI (all instruction cycles except for the first cycle)
  - Interrupt overhead

**NOTE**

An event is emitted each time a counter overflows.

- The DWT can be configured to emit PC samples at defined intervals, and to emit interrupt event information.

## 9.14 Debug in Low Power Modes

In low power modes in which the debug modules are kept static or powered off, the debugger cannot gather any debug data for the duration of the low power mode. In the case that the debugger is held static, the debug port returns to full functionality as soon as the low power mode exits and the system returns to a state with active debug. In the case that the debugger logic is powered off, the debugger is reset on recovery and must be reconfigured once the low power mode is exited.

Power mode entry logic monitors Debug Power Up and System Power Up signals from the debug port as indications that a debugger is active. These signals can be changed in RUN, VLPR, WAIT and VLPW. If the debug signal is active and the system attempts to enter stop or VLPS, FCLK continues to run to support core register access. In these modes in which FCLK is left active the debug modules have access to core registers but not to system memory resources accessed via the crossbar.

With debug enabled, transitions from Run directly to VLPS are not allowed and result in the system entering Stop mode instead. Status bits within the MDM-AP Status register can be evaluated to determine this pseudo-VLPS state. Note with the debug enabled, transitions from Run--> VLPR --> VLPS are still possible but also result in the system entering Stop mode instead.

In VLLS mode all debug modules are powered off and reset at wakeup. In LLS mode, the debug modules retain their state but no debug activity is possible.

**NOTE**

When using cJTAG and entering LLS mode, the cJTAG controller must be reset on exit from LLS mode.

Going into a VLLSx mode causes all the debug controls and settings to be reset. To give time to the debugger to sync up with the HW, the MDM-AP Control register can be configured hold the system in reset on recovery so that the debugger can regain control and reconfigure debug logic prior to the system exiting reset and resuming operation.

### 9.14.1 Debug Module State in Low Power Modes

The following table shows the state of the debug modules in low power modes. These terms are used:

- FF = Full functionality. In VLPR and VLPW the system frequency is limited, but if a module does not have a limitation in its functionality, it is still listed as FF.
- static = Module register states and associated memories are retained.
- OFF = Modules are powered off; module is in reset state upon wakeup.

**Table 9-7. Debug Module State in Low Power Modes**

| Module | STOP | VLPR | VLPW | VLPS | LLS | VLLSx |
|---|---|---|---|---|---|---|
| Debug Port | FF | FF | FF | OFF | static | OFF |
| AHB-AP | FF | FF | FF | OFF | static | OFF |
| ITM | FF | FF | FF | OFF | static | OFF |
| TPIU | FF | FF | FF | OFF | static | OFF |
| DWT | FF | FF | FF | OFF | static | OFF |

## 9.15 Debug & Security

When security is enabled (FSEC[SEC] != 10), the debug port capabilities are limited in order to prevent exploitation of secure data. In the secure state the debugger still has access to the MDM-AP Status Register and can determine the current security state of the device. In the case of a secure device, the debugger also has the capability of performing a mass erase operation via writes to the MDM-AP Control Register. In the case of a secure device that has mass erase disabled (FSEC[MEEN] = 10), attempts to mass erase via the debug interface are blocked.

# Chapter 10
# Signal Multiplexing and Signal Descriptions

## 10.1 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.

The Port Control block controls which signal is present on the external pin. Reference that chapter to find which register controls the operation of a specific pin.

## 10.2 Signal Multiplexing Integration

This section summarizes how the module is integrated into the device. For a comprehensive description of the module itself, see the module's dedicated chapter.

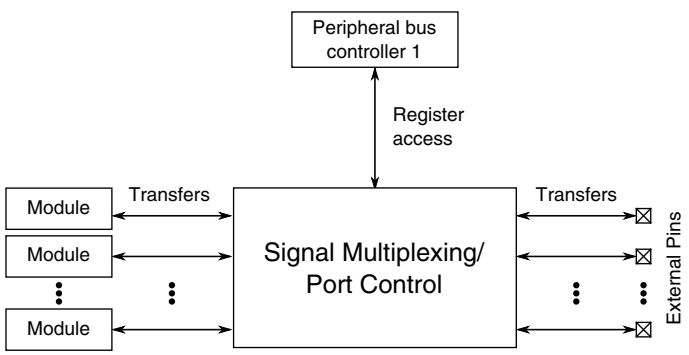



**Figure 10-1. Signal multiplexing integration**

**Table 10-1. Reference links to related information**

| Topic | Related module | Reference |
|---|---|---|
| Full description | Port control | Port control |
| System memory map | | System memory map |

*Table continues on the next page...*

**Table 10-1. Reference links to related information (continued)**

| Topic | Related module | Reference |
|---|---|---|
| Clocking | | Clock Distribution |
| Register access | Peripheral bus controller | Peripheral bridge |

## 10.2.1 Port control and interrupt module features

- Five 32-pin ports

**NOTE**

Not all pins are available on the device. See the following section for details.

- Each 32-pin port is assigned one interrupt.

## 10.2.2 PCRn reset values for port A

PCRn bit reset values for port A are 1 for the following bits:

- For PCR0: bits 1, 6, 8, 9, and 10.
- For PCR1 to PCR4: bits 0, 1, 6, 8, 9, and 10.
- For PCR5 : bits 0, 1, and 6.

All other PCRn bit reset values for port A are 0.

## 10.2.3 Clock gating

The clock to the port control module can be gated on and off using the SCGC5[PORTx] bits in the SIM module. These bits are cleared after any reset, which disables the clock to the corresponding module to conserve power. Prior to initializing the corresponding module, set SCGC5[PORTx] in the SIM module to enable the clock. Before turning off the clock, make sure to disable the module. For more details, refer to the clock distribution chapter.

## 10.2.4 Signal multiplexing constraints

1. A given peripheral function must be assigned to a maximum of one package pin. Do not program the same function to more than one pin.

2. To ensure the best signal timing for a given peripheral's interface, choose the pins in closest proximity to each other.

## 10.3 Pinout

### 10.3.1 K21 Signal Multiplexing and Pin Assignments

The following table shows the signals available on each pin and the locations of these pins on the devices supported by this document. The Port Control Module is responsible for selecting which ALT functionality is available on each pin.

**NOTE**

- The analog input signals ADC0_DP2 and ADC0_DM2 on PTE2 and PTE3 are available only for K21 and K22 devices and are not present on K10 and K20 devices.
- The TRACE signals on PTE0, PTE1, PTE2, PTE3, and PTE4 are available only for K11, K12, K21, and K22 devices and are not present on K10 and K20 devices.
- If the VBAT pin is not used, the VBAT pin should be left floating. Do not connect VBAT pin to VSS.
- The FTM_CLKIN signals on PTB16 and PTB17 are available only for K11, K12, K21, and K22 devices and is not present on K10 and K20 devices. For K22D devices this signal is on ALT7, and for K22F devices, this signal is on ALT4.
- The FTM0_CH2 signal on PTC5/LLWU_P9 is available only for K11, K12, K21, and K22 devices and is not present on K10 and K20 devices.
- The I2C0_SCL signal on PTD2/LLWU_P13 and I2C0_SDA signal on PTD3 are available only for K11, K12, K21, and K22 devices and are not present on K10 and K20 devices.

| 144 MAP BGA | 144 LQFP | Pin Name | Default | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 | ALT6 | ALT7 | EzPort |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D3 | 1 | PTE0 | ADC1_SE4a | ADC1_SE4a | PTE0 | SPI1_PCS1 | UART1_TX | SDHC0_D1 | TRACE_CLKOUT | I2C1_SDA | RTC_CLKOUT | |
| D2 | 2 | PTE1/LLWU_P0 | ADC1_SE5a | ADC1_SE5a | PTE1/LLWU_P0 | SPI1_SOUT | UART1_RX | SDHC0_D0 | TRACE_D3 | I2C1_SCL | SPI1_SIN | |
| D1 | 3 | PTE2/LLWU_P1 | ADC0_DP2/ADC1_SE6a | ADC0_DP2/ADC1_SE6a | PTE2/LLWU_P1 | SPI1_SCK | UART1_CTS_b | SDHC0_DCLK | TRACE_D2 | | | |

| 144 MAP BGA | 144 LQFP | Pin Name | Default | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 | ALT6 | ALT7 | EzPort |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E4 | 4 | PTE3 | ADC0_DM2/ ADC1_SE7a | ADC0_DM2/ ADC1_SE7a | PTE3 | SPI1_SIN | UART1_RTS_ b | SDHC0_CMD | TRACE_D1 | | SPI1_SOUT | |
| E5 | 5 | VDD | VDD | VDD | | | | | | | | |
| F6 | 6 | VSS | VSS | VSS | | | | | | | | |
| E3 | 7 | PTE4/ LLWU_P2 | DISABLED | | PTE4/ LLWU_P2 | SPI1_PCS0 | UART3_TX | SDHC0_D3 | TRACE_D0 | | | |
| E2 | 8 | PTE5 | DISABLED | | PTE5 | SPI1_PCS2 | UART3_RX | SDHC0_D2 | | FTM3_CH0 | | |
| E1 | 9 | PTE6 | DISABLED | | PTE6 | SPI1_PCS3 | UART3_CTS_ b | I2S0_MCLK | | FTM3_CH1 | USB_SOF_ OUT | |
| F4 | 10 | PTE7 | DISABLED | | PTE7 | | UART3_RTS_ b | I2S0_RXD0 | | FTM3_CH2 | | |
| F3 | 11 | PTE8 | DISABLED | | PTE8 | I2S0_RXD1 | UART5_TX | I2S0_RX_FS | | FTM3_CH3 | | |
| F2 | 12 | PTE9 | DISABLED | | PTE9 | I2S0_TXD1 | UART5_RX | I2S0_RX_ BCLK | | FTM3_CH4 | | |
| F1 | 13 | PTE10 | DISABLED | | PTE10 | | UART5_CTS_ b | I2S0_TXD0 | | FTM3_CH5 | | |
| G4 | 14 | PTE11 | DISABLED | | PTE11 | | UART5_RTS_ b | I2S0_TX_FS | | FTM3_CH6 | | |
| G3 | 15 | PTE12 | DISABLED | | PTE12 | | | I2S0_TX_ BCLK | | FTM3_CH7 | | |
| E6 | 16 | VDD | VDD | VDD | | | | | | | | |
| F7 | 17 | VSS | VSS | VSS | | | | | | | | |
| H3 | 18 | VSS | VSS | VSS | | | | | | | | |
| H1 | 19 | USB0_DP | USB0_DP | USB0_DP | | | | | | | | |
| H2 | 20 | USB0_DM | USB0_DM | USB0_DM | | | | | | | | |
| G1 | 21 | VOUT33 | VOUT33 | VOUT33 | | | | | | | | |
| G2 | 22 | VREGIN | VREGIN | VREGIN | | | | | | | | |
| J1 | 23 | ADC0_DP1 | ADC0_DP1 | ADC0_DP1 | | | | | | | | |
| J2 | 24 | ADC0_DM1 | ADC0_DM1 | ADC0_DM1 | | | | | | | | |
| K1 | 25 | ADC1_DP1 | ADC1_DP1 | ADC1_DP1 | | | | | | | | |
| K2 | 26 | ADC1_DM1 | ADC1_DM1 | ADC1_DM1 | | | | | | | | |
| L1 | 27 | ADC0_DP0/ ADC1_DP3 | ADC0_DP0/ ADC1_DP3 | ADC0_DP0/ ADC1_DP3 | | | | | | | | |
| L2 | 28 | ADC0_DM0/ ADC1_DM3 | ADC0_DM0/ ADC1_DM3 | ADC0_DM0/ ADC1_DM3 | | | | | | | | |
| M1 | 29 | ADC1_DP0/ ADC0_DP3 | ADC1_DP0/ ADC0_DP3 | ADC1_DP0/ ADC0_DP3 | | | | | | | | |
| M2 | 30 | ADC1_DM0/ ADC0_DM3 | ADC1_DM0/ ADC0_DM3 | ADC1_DM0/ ADC0_DM3 | | | | | | | | |
| H5 | 31 | VDDA | VDDA | VDDA | | | | | | | | |
| G5 | 32 | VREFH | VREFH | VREFH | | | | | | | | |
| G6 | 33 | VREFL | VREFL | VREFL | | | | | | | | |
| H6 | 34 | VSSA | VSSA | VSSA | | | | | | | | |

| 144 MAP BGA | 144 LQFP | Pin Name | Default | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 | ALT6 | ALT7 | EzPort |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| K3 | 35 | ADC1_SE16/ CMP2_IN2/ ADC0_SE22 | ADC1_SE16/ CMP2_IN2/ ADC0_SE22 | ADC1_SE16/ CMP2_IN2/ ADC0_SE22 | | | | | | | | |
| J3 | 36 | ADC0_SE16/ CMP1_IN2/ ADC0_SE21 | ADC0_SE16/ CMP1_IN2/ ADC0_SE21 | ADC0_SE16/ CMP1_IN2/ ADC0_SE21 | | | | | | | | |
| M3 | — | VREF_OUT/ CMP1_IN5/ CMP0_IN5/ ADC1_SE18 | VREF_OUT/ CMP1_IN5/ CMP0_IN5/ ADC1_SE18 | VREF_OUT/ CMP1_IN5/ CMP0_IN5/ ADC1_SE18 | | | | | | | | |
| L3 | — | DAC0_OUT/ CMP1_IN3/ ADC0_SE23 | DAC0_OUT/ CMP1_IN3/ ADC0_SE23 | DAC0_OUT/ CMP1_IN3/ ADC0_SE23 | | | | | | | | |
| L4 | — | DAC1_OUT/ CMP0_IN4/ CMP2_IN3/ ADC1_SE23 | DAC1_OUT/ CMP0_IN4/ CMP2_IN3/ ADC1_SE23 | DAC1_OUT/ CMP0_IN4/ CMP2_IN3/ ADC1_SE23 | | | | | | | | |
| L5 | 37 | TAMPER0/ RTC_ WAKEUP_B | TAMPER0/ RTC_ WAKEUP_B | TAMPER0/ RTC_ WAKEUP_B | | | | | | | | |
| K5 | 38 | TAMPER1 | TAMPER1 | TAMPER1 | | | | | | | | |
| K4 | 39 | TAMPER2 | TAMPER2 | TAMPER2 | | | | | | | | |
| J4 | — | TAMPER3 | TAMPER3 | TAMPER3 | | | | | | | | |
| H4 | — | TAMPER4 | TAMPER4 | TAMPER4 | | | | | | | | |
| M4 | — | TAMPER5 | TAMPER5 | TAMPER5 | | | | | | | | |
| M7 | 40 | XTAL32 | XTAL32 | XTAL32 | | | | | | | | |
| M6 | 41 | EXTAL32 | EXTAL32 | EXTAL32 | | | | | | | | |
| L6 | 42 | VBAT | VBAT | VBAT | | | | | | | | |
| — | 43 | VDD | VDD | VDD | | | | | | | | |
| — | 44 | VSS | VSS | VSS | | | | | | | | |
| — | 45 | PTE24 | ADC0_SE17 | ADC0_SE17 | PTE24 | | UART4_TX | | | EWM_OUT_b | | |
| — | 46 | PTE25 | ADC0_SE18 | ADC0_SE18 | PTE25 | | UART4_RX | | | EWM_IN | | |
| — | 47 | PTE26 | DISABLED | | PTE26 | | UART4_CTS_ b | | | RTC_CLKOUT | USB_CLKIN | |
| — | 48 | PTE27 | DISABLED | | PTE27 | | UART4_RTS_ b | | | | | |
| — | 49 | PTE28 | DISABLED | | PTE28 | | | | | | | |
| J5 | 50 | PTA0 | JTAG_TCLK/ SWD_CLK/ EZP_CLK | | PTA0 | UART0_CTS_ b/ UART0_COL_ b | FTM0_CH5 | | | | JTAG_TCLK/ SWD_CLK | EZP_CLK |
| J6 | 51 | PTA1 | JTAG_TDI/ EZP_DI | | PTA1 | UART0_RX | FTM0_CH6 | | | | JTAG_TDI | EZP_DI |
| K6 | 52 | PTA2 | JTAG_TDO/ TRACE_SWO/ EZP_DO | | PTA2 | UART0_TX | FTM0_CH7 | | | | JTAG_TDO/ TRACE_SWO | EZP_DO |

| 144 MAP BGA | 144 LQFP | Pin Name | Default | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 | ALT6 | ALT7 | EzPort |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| K7 | 53 | PTA3 | JTAG_TMS/ SWD_DIO | | PTA3 | UART0_RTS_ b | FTM0_CH0 | | | | JTAG_TMS/ SWD_DIO | |
| L7 | 54 | PTA4/ LLWU_P3 | NMI_b/ EZP_CS_b | | PTA4/ LLWU_P3 | | FTM0_CH1 | | | | NMI_b | EZP_CS_b |
| M8 | 55 | PTA5 | DISABLED | | PTA5 | USB_CLKIN | FTM0_CH2 | | CMP2_OUT | I2S0_TX_ BCLK | JTAG_TRST_ b | |
| E7 | 56 | VDD | VDD | VDD | | | | | | | | |
| G7 | 57 | VSS | VSS | VSS | | | | | | | | |
| J7 | 58 | PTA6 | DISABLED | | PTA6 | | FTM0_CH3 | | CLKOUT | | TRACE_ CLKOUT | |
| J8 | 59 | PTA7 | ADC0_SE10 | ADC0_SE10 | PTA7 | | FTM0_CH4 | | | | TRACE_D3 | |
| K8 | 60 | PTA8 | ADC0_SE11 | ADC0_SE11 | PTA8 | | FTM1_CH0 | | | FTM1_QD_ PHA | TRACE_D2 | |
| L8 | 61 | PTA9 | DISABLED | | PTA9 | | FTM1_CH1 | | | FTM1_QD_ PHB | TRACE_D1 | |
| M9 | 62 | PTA10 | DISABLED | | PTA10 | | FTM2_CH0 | | | FTM2_QD_ PHA | TRACE_D0 | |
| L9 | 63 | PTA11 | DISABLED | | PTA11 | | FTM2_CH1 | | I2C2_SDA | FTM2_QD_ PHB | | |
| K9 | 64 | PTA12 | CMP2_IN0 | CMP2_IN0 | PTA12 | CAN0_TX | FTM1_CH0 | | I2C2_SCL | I2S0_TXD0 | FTM1_QD_ PHA | |
| J9 | 65 | PTA13/ LLWU_P4 | CMP2_IN1 | CMP2_IN1 | PTA13/ LLWU_P4 | CAN0_RX | FTM1_CH1 | | I2C2_SDA | I2S0_TX_FS | FTM1_QD_ PHB | |
| L10 | 66 | PTA14 | DISABLED | | PTA14 | SPI0_PCS0 | UART0_TX | | I2C2_SCL | I2S0_RX_ BCLK | I2S0_TXD1 | |
| L11 | 67 | PTA15 | DISABLED | | PTA15 | SPI0_SCK | UART0_RX | | | I2S0_RXD0 | | |
| K10 | 68 | PTA16 | DISABLED | | PTA16 | SPI0_SOUT | UART0_CTS_ b/ UART0_COL_ b | | | I2S0_RX_FS | I2S0_RXD1 | |
| K11 | 69 | PTA17 | ADC1_SE17 | ADC1_SE17 | PTA17 | SPI0_SIN | UART0_RTS_ b | | | I2S0_MCLK | | |
| E8 | 70 | VDD | VDD | VDD | | | | | | | | |
| G8 | 71 | VSS | VSS | VSS | | | | | | | | |
| M12 | 72 | PTA18 | EXTAL0 | EXTAL0 | PTA18 | | FTM0_FLT2 | FTM_CLKIN0 | | | | |
| M11 | 73 | PTA19 | XTAL0 | XTAL0 | PTA19 | | FTM1_FLT0 | FTM_CLKIN1 | | LPTMR0_ ALT1 | | |
| L12 | 74 | RESET_b | RESET_b | RESET_b | | | | | | | | |
| K12 | 75 | PTA24 | DISABLED | | PTA24 | | | | | FB_A29 | | |
| J12 | 76 | PTA25 | DISABLED | | PTA25 | | | | | FB_A28 | | |
| J11 | 77 | PTA26 | DISABLED | | PTA26 | | | | | FB_A27 | | |
| J10 | 78 | PTA27 | DISABLED | | PTA27 | | | | | FB_A26 | | |
| H12 | 79 | PTA28 | DISABLED | | PTA28 | | | | | FB_A25 | | |
| H11 | 80 | PTA29 | DISABLED | | PTA29 | | | | | FB_A24 | | |

| 144 MAP BGA | 144 LQFP | Pin Name | Default | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 | ALT6 | ALT7 | EzPort |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| H10 | 81 | PTB0/ LLWU_P5 | ADC0_SE8/ ADC1_SE8 | ADC0_SE8/ ADC1_SE8 | PTB0/ LLWU_P5 | I2C0_SCL | FTM1_CH0 | | | FTM1_QD_ PHA | | |
| H9 | 82 | PTB1 | ADC0_SE9/ ADC1_SE9 | ADC0_SE9/ ADC1_SE9 | PTB1 | I2C0_SDA | FTM1_CH1 | | | FTM1_QD_ PHB | | |
| G12 | 83 | PTB2 | ADC0_SE12 | ADC0_SE12 | PTB2 | I2C0_SCL | UART0_RTS_ b | | | FTM0_FLT3 | | |
| G11 | 84 | PTB3 | ADC0_SE13 | ADC0_SE13 | PTB3 | I2C0_SDA | UART0_CTS_ b/ UART0_COL_ b | | | FTM0_FLT0 | | |
| G10 | 85 | PTB4 | ADC1_SE10 | ADC1_SE10 | PTB4 | | | | | FTM1_FLT0 | | |
| G9 | 86 | PTB5 | ADC1_SE11 | ADC1_SE11 | PTB5 | | | | | FTM2_FLT0 | | |
| F12 | 87 | PTB6 | ADC1_SE12 | ADC1_SE12 | PTB6 | | | | FB_AD23 | | | |
| F11 | 88 | PTB7 | ADC1_SE13 | ADC1_SE13 | PTB7 | | | | FB_AD22 | | | |
| F10 | 89 | PTB8 | DISABLED | | PTB8 | | UART3_RTS_ b | | FB_AD21 | | | |
| F9 | 90 | PTB9 | DISABLED | | PTB9 | SPI1_PCS1 | UART3_CTS_ b | | FB_AD20 | | | |
| E12 | 91 | PTB10 | ADC1_SE14 | ADC1_SE14 | PTB10 | SPI1_PCS0 | UART3_RX | | FB_AD19 | FTM0_FLT1 | | |
| E11 | 92 | PTB11 | ADC1_SE15 | ADC1_SE15 | PTB11 | SPI1_SCK | UART3_TX | | FB_AD18 | FTM0_FLT2 | | |
| H7 | 93 | VSS | VSS | VSS | | | | | | | | |
| F5 | 94 | VDD | VDD | VDD | | | | | | | | |
| E10 | 95 | PTB16 | DISABLED | | PTB16 | SPI1_SOUT | UART0_RX | FTM_CLKIN0 | FB_AD17 | EWM_IN | | |
| E9 | 96 | PTB17 | DISABLED | | PTB17 | SPI1_SIN | UART0_TX | FTM_CLKIN1 | FB_AD16 | EWM_OUT_b | | |
| D12 | 97 | PTB18 | DISABLED | | PTB18 | CAN0_TX | FTM2_CH0 | I2S0_TX_ BCLK | FB_AD15 | FTM2_QD_ PHA | | |
| D11 | 98 | PTB19 | DISABLED | | PTB19 | CAN0_RX | FTM2_CH1 | I2S0_TX_FS | FB_OE_b | FTM2_QD_ PHB | | |
| D10 | 99 | PTB20 | DISABLED | | PTB20 | SPI2_PCS0 | | | FB_AD31 | CMP0_OUT | | |
| D9 | 100 | PTB21 | DISABLED | | PTB21 | SPI2_SCK | | | FB_AD30 | CMP1_OUT | | |
| C12 | 101 | PTB22 | DISABLED | | PTB22 | SPI2_SOUT | | | FB_AD29 | CMP2_OUT | | |
| C11 | 102 | PTB23 | DISABLED | | PTB23 | SPI2_SIN | SPI0_PCS5 | | FB_AD28 | | | |
| B12 | 103 | PTC0 | ADC0_SE14 | ADC0_SE14 | PTC0 | SPI0_PCS4 | PDB0_EXTRG | | FB_AD14 | I2S0_TXD1 | | |
| B11 | 104 | PTC1/ LLWU_P6 | ADC0_SE15 | ADC0_SE15 | PTC1/ LLWU_P6 | SPI0_PCS3 | UART1_RTS_ b | FTM0_CH0 | FB_AD13 | I2S0_TXD0 | | |
| A12 | 105 | PTC2 | ADC0_SE4b/ CMP1_IN0 | ADC0_SE4b/ CMP1_IN0 | PTC2 | SPI0_PCS2 | UART1_CTS_ b | FTM0_CH1 | FB_AD12 | I2S0_TX_FS | | |
| A11 | 106 | PTC3/ LLWU_P7 | CMP1_IN1 | CMP1_IN1 | PTC3/ LLWU_P7 | SPI0_PCS1 | UART1_RX | FTM0_CH2 | CLKOUT | I2S0_TX_ BCLK | | |
| H8 | 107 | VSS | VSS | VSS | | | | | | | | |
| — | 108 | VDD | VDD | VDD | | | | | | | | |
| A9 | 109 | PTC4/ LLWU_P8 | DISABLED | | PTC4/ LLWU_P8 | SPI0_PCS0 | UART1_TX | FTM0_CH3 | FB_AD11 | CMP1_OUT | | |
| D8 | 110 | PTC5/ LLWU_P9 | DISABLED | | PTC5/ LLWU_P9 | SPI0_SCK | LPTMR0_ ALT2 | I2S0_RXD0 | FB_AD10 | CMP0_OUT | FTM0_CH2 | |

| 144 MAP BGA | 144 LQFP | Pin Name | Default | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 | ALT6 | ALT7 | EzPort |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C8 | 111 | PTC6/ LLWU_P10 | CMP0_IN0 | CMP0_IN0 | PTC6/ LLWU_P10 | SPI0_SOUT | PDB0_EXTRG | I2S0_RX_ BCLK | FB_AD9 | I2S0_MCLK | | |
| B8 | 112 | PTC7 | CMP0_IN1 | CMP0_IN1 | PTC7 | SPI0_SIN | USB_SOF_ OUT | I2S0_RX_FS | FB_AD8 | | | |
| A8 | 113 | PTC8 | ADC1_SE4b/ CMP0_IN2 | ADC1_SE4b/ CMP0_IN2 | PTC8 | | FTM3_CH4 | I2S0_MCLK | FB_AD7 | | | |
| D7 | 114 | PTC9 | ADC1_SE5b/ CMP0_IN3 | ADC1_SE5b/ CMP0_IN3 | PTC9 | | FTM3_CH5 | I2S0_RX_ BCLK | FB_AD6 | FTM2_FLT0 | | |
| C7 | 115 | PTC10 | ADC1_SE6b | ADC1_SE6b | PTC10 | I2C1_SCL | FTM3_CH6 | I2S0_RX_FS | FB_AD5 | | | |
| B7 | 116 | PTC11/ LLWU_P11 | ADC1_SE7b | ADC1_SE7b | PTC11/ LLWU_P11 | I2C1_SDA | FTM3_CH7 | I2S0_RXD1 | FB_RW_b | | | |
| A7 | 117 | PTC12 | DISABLED | | PTC12 | | UART4_RTS_ b | | FB_AD27 | FTM3_FLT0 | | |
| D6 | 118 | PTC13 | DISABLED | | PTC13 | | UART4_CTS_ b | | FB_AD26 | | | |
| C6 | 119 | PTC14 | DISABLED | | PTC14 | | UART4_RX | | FB_AD25 | | | |
| B6 | 120 | PTC15 | DISABLED | | PTC15 | | UART4_TX | | FB_AD24 | | | |
| — | 121 | VSS | VSS | VSS | | | | | | | | |
| — | 122 | VDD | VDD | VDD | | | | | | | | |
| A6 | 123 | PTC16 | DISABLED | | PTC16 | | UART3_RX | | FB_CS5_b/ FB_TSIZ1/ FB_BE23_16_ BLS15_8_b | | | |
| D5 | 124 | PTC17 | DISABLED | | PTC17 | | UART3_TX | | FB_CS4_b/ FB_TSIZ0/ FB_BE31_24_ BLS7_0_b | | | |
| C5 | 125 | PTC18 | DISABLED | | PTC18 | | UART3_RTS_ b | | FB_TBST_b/ FB_CS2_b/ FB_BE15_8_ BLS23_16_b | | | |
| B5 | 126 | PTC19 | DISABLED | | PTC19 | | UART3_CTS_ b | | FB_CS3_b/ FB_BE7_0_ BLS31_24_b | FB_TA_b | | |
| A5 | 127 | PTD0/ LLWU_P12 | DISABLED | | PTD0/ LLWU_P12 | SPI0_PCS0 | UART2_RTS_ b | FTM3_CH0 | FB_ALE/ FB_CS1_b/ FB_TS_b | | | |
| D4 | 128 | PTD1 | ADC0_SE5b | ADC0_SE5b | PTD1 | SPI0_SCK | UART2_CTS_ b | FTM3_CH1 | FB_CS0_b | | | |
| C4 | 129 | PTD2/ LLWU_P13 | DISABLED | | PTD2/ LLWU_P13 | SPI0_SOUT | UART2_RX | FTM3_CH2 | FB_AD4 | | I2C0_SCL | |
| B4 | 130 | PTD3 | DISABLED | | PTD3 | SPI0_SIN | UART2_TX | FTM3_CH3 | FB_AD3 | | I2C0_SDA | |
| A4 | 131 | PTD4/ LLWU_P14 | DISABLED | | PTD4/ LLWU_P14 | SPI0_PCS1 | UART0_RTS_ b | FTM0_CH4 | FB_AD2 | EWM_IN | | |
| A3 | 132 | PTD5 | ADC0_SE6b | ADC0_SE6b | PTD5 | SPI0_PCS2 | UART0_CTS_ b/ UART0_COL_ b | FTM0_CH5 | FB_AD1 | EWM_OUT_b | | |

| 144 MAP BGA | 144 LQFP | Pin Name | Default | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 | ALT6 | ALT7 | EzPort |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A2 | 133 | PTD6/ LLWU_P15 | ADC0_SE7b | ADC0_SE7b | PTD6/ LLWU_P15 | SPI0_PCS3 | UART0_RX | FTM0_CH6 | FB_AD0 | FTM0_FLT0 | | |
| M10 | 134 | VSS | VSS | VSS | | | | | | | | |
| F8 | 135 | VDD | VDD | VDD | | | | | | | | |
| A1 | 136 | PTD7 | DISABLED | | PTD7 | CMT_IRO | UART0_TX | FTM0_CH7 | | FTM0_FLT1 | | |
| C9 | 137 | PTD8 | DISABLED | | PTD8 | I2C0_SCL | UART5_RX | | | FB_A16 | | |
| B9 | 138 | PTD9 | DISABLED | | PTD9 | I2C0_SDA | UART5_TX | | | FB_A17 | | |
| B3 | 139 | PTD10 | DISABLED | | PTD10 | | UART5_RTS_ b | | | FB_A18 | | |
| B2 | 140 | PTD11 | DISABLED | | PTD11 | SPI2_PCS0 | UART5_CTS_ b | SDHC0_ CLKIN | | FB_A19 | | |
| B1 | 141 | PTD12 | DISABLED | | PTD12 | SPI2_SCK | FTM3_FLT0 | SDHC0_D4 | | FB_A20 | | |
| C3 | 142 | PTD13 | DISABLED | | PTD13 | SPI2_SOUT | | SDHC0_D5 | | FB_A21 | | |
| C2 | 143 | PTD14 | DISABLED | | PTD14 | SPI2_SIN | | SDHC0_D6 | | FB_A22 | | |
| C1 | 144 | PTD15 | DISABLED | | PTD15 | SPI2_PCS1 | | SDHC0_D7 | | FB_A23 | | |
| M5 | — | NC | NC | NC | | | | | | | | |
| A10 | — | NC | NC | NC | | | | | | | | |
| B10 | — | NC | NC | NC | | | | | | | | |
| C10 | — | NC | NC | NC | | | | | | | | |

## 10.3.2 K21 Pinouts

The below figure shows the pinout diagram for the devices supported by this document. Many signals may be multiplexed onto a single pin. To determine what signals can be used on which pin, see the previous section.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | PTD7 | PTD6/ LLWU_P15 | PTD5 | PTD4/ LLWU_P14 | PTD0/ LLWU_P12 | PTC16 | PTC12 | PTC8 | PTC4/ LLWU_P8 | NC | PTC3/ LLWU_P7 | PTC2 | A |
| B | PTD12 | PTD11 | PTD10 | PTD3 | PTC19 | PTC15 | PTC11/ LLWU_P11 | PTC7 | PTD9 | NC | PTC1/ LLWU_P6 | PTC0 | B |
| C | PTD15 | PTD14 | PTD13 | PTD2/ LLWU_P13 | PTC18 | PTC14 | PTC10 | PTC6/ LLWU_P10 | PTD8 | NC | PTB23 | PTB22 | C |
| D | PTE2/ LLWU_P1 | PTE1/ LLWU_P0 | PTE0 | PTD1 | PTC17 | PTC13 | PTC9 | PTC5/ LLWU_P9 | PTB21 | PTB20 | PTB19 | PTB18 | D |
| E | PTE6 | PTE5 | PTE4/ LLWU_P2 | PTE3 | VDD | VDD | VDD | VDD | PTB17 | PTB16 | PTB11 | PTB10 | E |
| F | PTE10 | PTE9 | PTE8 | PTE7 | VDD | VSS | VSS | VDD | PTB9 | PTB8 | PTB7 | PTB6 | F |
| G | VOUT33 | VREGIN | PTE12 | PTE11 | VREFH | VREFL | VSS | VSS | PTB5 | PTB4 | PTB3 | PTB2 | G |
| H | USB0_DP | USB0_DM | VSS | TAMPER4 | VDDA | VSSA | VSS | VSS | PTB1 | PTB0/ LLWU_P5 | PTA29 | PTA28 | H |
| J | ADC0_DP1 | ADC0_DM1 | ADC0_SE16/ CMP1_IN2/ ADC0_SE21 | TAMPER3 | PTA0 | PTA1 | PTA6 | PTA7 | PTA13/ LLWU_P4 | PTA27 | PTA26 | PTA25 | J |
| K | ADC1_DP1 | ADC1_DM1 | ADC1_SE16/ CMP2_IN2/ ADC0_SE22 | TAMPER2 | TAMPER1 | PTA2 | PTA3 | PTA8 | PTA12 | PTA16 | PTA17 | PTA24 | K |
| L | ADC0_DP0/ ADC1_DP3 | ADC0_DM0/ ADC1_DM3 | DAC0_OUT/ CMP1_IN3/ ADC0_SE23 | DAC1_OUT/ CMP0_IN4/ CMP2_IN3/ ADC1_SE23 | TAMPER0/ RTC_ WAKEUP_B | VBAT | PTA4/ LLWU_P3 | PTA9 | PTA11 | PTA14 | PTA15 | RESET_b | L |
| M | ADC1_DP0/ ADC0_DP3 | ADC1_DM0/ ADC0_DM3 | VREF_OUT/ CMP1_IN5/ CMP0_IN5/ ADC1_SE18 | TAMPER5 | NC | EXTAL32 | XTAL32 | PTA5 | PTA10 | VSS | PTA19 | PTA18 | M |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | |

**Figure 10-2. K21 144 MAPBGA Pinout Diagram**

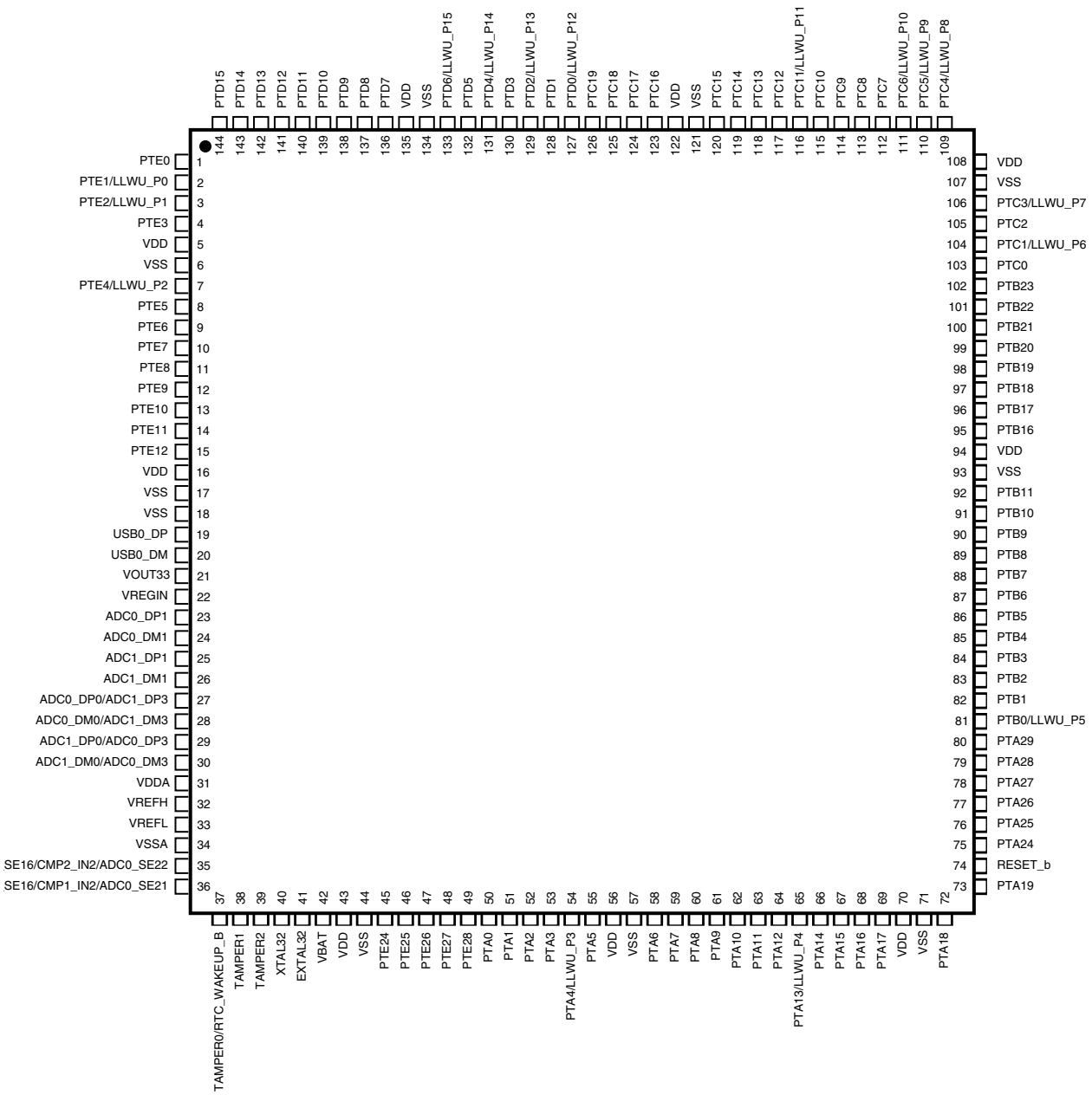



**Figure 10-3. K21 144 LQFP Pinout Diagram**

## 10.4 Module Signal Description Tables

The following sections correlate the chip-level signal name with the signal name used in the module's chapter. They also briefly describe the signal function and direction.

## 10.4.1 Core Modules

**Table 10-2. JTAG Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| JTAG_TMS | JTAG_TMS/ SWD_DIO | JTAG Test Mode Selection | I/O |
| JTAG_TCLK | JTAG_TCLK/ SWD_CLK | JTAG Test Clock | I |
| JTAG_TDI | JTAG_TDI | JTAG Test Data Input | I |
| JTAG_TDO | JTAG_TDO/ TRACE_SWO | JTAG Test Data Output | O |
| JTAG_TRST | JTAG_TRST_b | JTAG Reset | I |

**Table 10-3. SWD Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| SWD_DIO | JTAG_TMS/ SWD_DIO | Serial Wire Data | I/O |
| SWD_CLK | JTAG_TCLK/ SWD_CLK | Serial Wire Clock | I |

**Table 10-4. TPIU Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| TRACE_SWO | JTAG_TDO/ TRACE_SWO | Trace output data from the ARM CoreSight debug block over a single pin | O |

## 10.4.2 System Modules

**Table 10-5. System Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| $\overline{\text{NMI}}$ | — | Non-maskable interrupt<br>**NOTE:** Driving the $\overline{\text{NMI}}$ signal low forces a non-maskable interrupt, if the $\overline{\text{NMI}}$ function is selected on the corresponding pin. | I |
| $\overline{\text{RESET}}$ | — | Reset bi-directional signal | I/O |
| VDD | — | MCU power | I |
| VSS | — | MCU ground | I |

**Table 10-6. EWM Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| EWM_IN | EWM_in | EWM input for safety status of external safety circuits. The polarity of EWM_in is programmable using the EWM_CTRL[ASSIN] bit. The default polarity is active-low. | I |
| $\overline{\text{EWM\_OUT}}$ | $\overline{\text{EWM\_out}}$ | EWM reset out signal | O |

## 10.4.3 Clock Modules

**Table 10-7. OSC Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| EXTAL0 | EXTAL | External clock/Oscillator input | I |
| XTAL0 | XTAL | Oscillator output | O |

**Table 10-8. RTC OSC Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| EXTAL32 | EXTAL32 | 32.768 kHz oscillator input | I |
| XTAL32 | XTAL32 | 32.768 kHz oscillator output | O |

## 10.4.4 Memories and Memory Interfaces

**Table 10-9. EzPort Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| EZP_CLK | EZP_CK | EzPort Clock | Input |
| $\overline{\text{EZP\_CS}}$ | $\overline{\text{EZP\_CS}}$ | EzPort Chip Select | Input |
| EZP_DI | EZP_D | EzPort Serial Data In | Input |
| EZP_DO | EZP_Q | EzPort Serial Data Out | Output |

## Table 10-10. FlexBus Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| CLKOUT | FB_CLK | O | FlexBus Clock Output |
| FB_A[29:16] | FB_A[29:16] | Address Bus<br>When FlexBus is used in a nonmultiplexed configuration, this is the address bus. When FlexBus is used in a multiplexed configuration, this bus is not used. | O |
| FB_AD[31:0] | FB_D31–FB_D0 | Data Bus—During the first cycle, this bus drives the upper address byte, addr[31:24].<br>When FlexBus is used in a nonmultiplexed configuration, this is the data bus, FB_D. When FlexBus is used in a multiplexed configuration, this is the address and data bus, FB_AD.<br>The number of byte lanes carrying the data is determined by the port size associated with the matching chip-select.<br>When FlexBus is used in a multiplexed configuration, the full 32-bit address is driven on the first clock of a bus cycle (address phase). After the first clock, the data is driven on the bus (data phase). During the data phase, the address is driven on the pins not used for data. For example, in 16-bit mode, the lower address is driven on FB_AD15–FB_AD0, and in 8-bit mode, the lower address is driven on FB_AD23–FB_AD0. | I/O |
| FB_CS[5:0] | FB_CS5–FB_CS0 | General Purpose Chip-Selects—Indicate which external memory or peripheral is selected. A particular chip-select is asserted when the transfer address is within the external memory's or peripheral's address space, as defined in CSAR[BA] and CSMR[BAM]. | O |
| FB_BE31_24_BLS7_0,<br>FB_BE23_16_BLS15_8,<br>FB_BE15_8_BLS23_16,<br>FB_BE7_0_BLS31_24 | FB_BE_31_24<br>FB_BE_23_16<br>FB_BE_15_8<br>FB_BE_7_0 | Byte Enables—Indicate that data is to be latched or driven onto a specific byte lane of the data bus. CSCR[BEM] determines if these signals are asserted on reads and writes or on writes only.<br>For external SRAM or flash devices, the FB_BE outputs should be connected to individual byte strobe signals. | O |
| FB_OE | FB_OE | Output Enable—Sent to the external memory or peripheral to enable a read transfer. This signal is asserted during read accesses only when a chip-select matches the current address decode. | O |
| FB_R W | FB_R/W | Read/Write—Indicates whether the current bus operation is a read operation (FB_R/W high) or a write operation (FB_R/W low). | O |
| FB_TS/ FB_ALE | FB_TS | Transfer Start—Indicates that the chip has begun a bus transaction and that the address and attributes are valid.<br>An inverted FB_TS is available as an address latch enable (FB_ALE), which indicates when the address is being driven on the FB_AD bus.<br>FB_TS/FB_ALE is asserted for one bus clock cycle.<br>The chip can extend this signal until the first positive clock edge after FB_CS asserts. See CSCR[EXTS] and Extended Transfer Start/Address Latch Enable. | O |

*Table continues on the next page...*

## Table 10-10. FlexBus Signal Descriptions (continued)

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| FB_TSIZ[1:0] | FB_TSIZ1–FB_TSIZ0 | Transfer Size—Indicates (along with $\overline{\text{FB\_TBST}}$) the data transfer size of the current bus operation. The interface supports 8-, 16-, and 32-bit operand transfers and allows accesses to 8-, 16-, and 32-bit data ports.<br>• 00b = 4 bytes<br>• 01b = 1 byte<br>• 10b = 2 bytes<br>• 11b = 16 bytes (line)<br><br>For misaligned transfers, FB_TSIZ1–FB_TSIZ0 indicate the size of each transfer. For example, if a 32-bit access through a 32-bit port device occurs at a misaligned offset of 1h, 8 bits are transferred first (FB_TSIZ1–FB_TSIZ0 = 01b), 16 bits are transferred next at offset 2h (FB_TSIZ1–FB_TSIZ0 = 10b), and the final 8 bits are transferred at offset 4h (FB_TSIZ1–FB_TSIZ0 = 01b).<br>For aligned transfers larger than the port size, FB_TSIZ1–FB_TSIZ0 behave as follows:<br>• If bursting is used, FB_TSIZ1–FB_TSIZ0 are driven to the transfer size.<br>• If bursting is inhibited, FB_TSIZ1–FB_TSIZ0 first show the entire transfer size and then show the port size.<br><br>For burst-inhibited transfers, FB_TSIZ1–FB_TSIZ0 change with each $\overline{\text{FB\_TS}}$ assertion to reflect the next transfer size.<br>For transfers to port sizes smaller than the transfer size, FB_TSIZ1–FB_TSIZ0 indicate the size of the entire transfer on the first access and the size of the current port transfer on subsequent transfers. For example, for a 32-bit write to an 8-bit port, FB_TSIZ1–FB_TSIZ0 are 00b for the first transaction and 01b for the next three transactions. If bursting is used for a 32-bit write to an 8-bit port, FB_TSIZ1–FB_TSIZ0 are driven to 00b for the entire transfer. | O |

*Table continues on the next page...*

**Table 10-10. FlexBus Signal Descriptions (continued)**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| FB_TA | FB_TA | Transfer Acknowledge—Indicates that the external data transfer is complete. When FB_TA is asserted during a read transfer, FlexBus latches the data and then terminates the transfer. When FB_TA is asserted during a write transfer, the transfer is terminated.<br><br>If auto-acknowledge is disabled (CSCR[AA] = 0), the external memory or peripheral drives FB_TA to terminate the transfer. If auto-acknowledge is enabled (CSCR[AA] = 1), FB_TA is generated internally after a specified number of wait states, or the external memory or peripheral may assert external FB_TA before the wait-state countdown to terminate the transfer early. The chip deasserts FB_CS one cycle after the last FB_TA is asserted. During read transfers, the external memory or peripheral must continue to drive data until FB_TA is recognized. For write transfers, the chip continues driving data one clock cycle after FB_CS is deasserted.<br><br>The number of wait states is determined by CSCR or the external FB_TA input. If the external FB_TA is used, the external memory or peripheral has complete control of the number of wait states.<br><br>**Note:** External memory or peripherals should assert FB_TA only while the FB_CS signal to the external memory or peripheral is asserted.<br><br>The CSPMCR register controls muxing of FB_TA with other signals. If auto-acknowledge is not used and CSPMCR does not allow FB_TA control, FlexBus may hang. | I |
| FB_TBST | FB_TBST | Transfer Burst—Indicates that a burst transfer is in progress as driven by the chip. A burst transfer can be 2 to 16 beats depending on FB_TSIZ1–FB_TSIZ0 and the port size.<br><br>**Note:** When a burst transfer is in progress (FB_TBST = 0b), the transfer size is 16 bytes (FB_TSIZ1–FB_TSIZ0 = 11b), and the address is misaligned within the 16-byte boundary, the external memory or peripheral must be able to wrap around the address. | O |

## 10.4.5 Security Modules

**Table 10-11. Tamper Detect Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| TAMPER[5:0] | TAMPER[7:0] | External tamper input or active tamper output | I/O |

## 10.4.6 Analog

**Table 10-12. ADC 0 Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| ADC0_DP[3:0] | DADP3–DADP0 | Differential Analog Channel Inputs | I |
| ADC0_DM[3:0] | DADM3–DADM0 | Differential Analog Channel Inputs | I |
| ADC0_SE[18:4] | AD*n* | Single-Ended Analog Channel Inputs | I |
| VREFH | $V_{REFSH}$ | Voltage Reference Select High | I |
| VREFL | $V_{REFSL}$ | Voltage Reference Select Low | I |
| VDDA | $V_{DDA}$ | Analog Power Supply | I |
| VSSA | $V_{SSA}$ | Analog Ground | I |

**Table 10-13. ADC 1 Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| ADC1_DP3, ADC1_DP[1:0] | DADP3–DADP0 | Differential Analog Channel Inputs | I |
| ADC1_DM3, ADC1_DM[1:0] | DADM3–DADM0 | Differential Analog Channel Inputs | I |
| ADC1_SE[18:4] | AD*n* | Single-Ended Analog Channel Inputs | I |
| VREFH | $V_{REFSH}$ | Voltage Reference Select High | I |
| VREFL | $V_{REFSL}$ | Voltage Reference Select Low | I |
| VDDA | $V_{DDA}$ | Analog Power Supply | I |
| VSSA | $V_{SSA}$ | Analog Ground | I |

**Table 10-14. CMP 0 Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| CMP0_IN[5:0] | IN[5:0] | Analog voltage inputs | I |
| CMP0_OUT | CMPO | Comparator output | O |

**Table 10-15. CMP 1 Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| CMP1_IN[5:0] | IN[5:0] | Analog voltage inputs | I |
| CMP1_OUT | CMPO | Comparator output | O |

**Table 10-16. CMP 2 Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| CMP2_IN[5:0] | IN[5:0] | Analog voltage inputs | I |
| CMP2_OUT | CMPO | Comparator output | O |

**Table 10-17. DAC 0 Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| DAC0_OUT | — | DAC output | O |

**Table 10-18. DAC 1 Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| DAC1_OUT | — | DAC output | O |

**Table 10-19. VREF Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| VREF_OUT | VREF_OUT | Internally-generated Voltage Reference output | O |

## 10.4.7 Timer Modules

**Table 10-20. FTM 0 Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| FTM_CLKIN[1:0] | EXTCLK | External clock. FTM external clock can be selected to drive the FTM counter. | I |
| FTM0_CH[7:0] | CHn | FTM channel (n), where n can be 7-0 | I/O |
| FTM0_FLT[3:0] | FAULTj | Fault input (j), where j can be 3-0 | I |

**Table 10-21. FTM 1 Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| FTM_CLKIN[1:0] | EXTCLK | External clock. FTM external clock can be selected to drive the FTM counter. | I |
| FTM1_CH[1:0] | CHn | FTM channel (n), where n can be 7-0 | I/O |

*Table continues on the next page...*

## Table 10-21. FTM 1 Signal Descriptions (continued)

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| FTM1_FLT0 | FAULTj | Fault input (j), where j can be 3-0 | I |
| FTM1_QD_PHA | PHA | Quadrature decoder phase A input. Input pin associated with quadrature decoder phase A. | I |
| FTM1_QD_PHB | PHB | Quadrature decoder phase B input. Input pin associated with quadrature decoder phase B. | I |

## Table 10-22. FTM 2 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| FTM_CLKIN[1:0] | EXTCLK | External clock. FTM external clock can be selected to drive the FTM counter. | I |
| FTM2_CH[1:0] | CHn | FTM channel (n), where n can be 7-0 | I/O |
| FTM2_FLT0 | FAULTj | Fault input (j), where j can be 3-0 | I |
| FTM2_QD_PHA | PHA | Quadrature decoder phase A input. Input pin associated with quadrature decoder phase A. | I |
| FTM2_QD_PHB | PHB | Quadrature decoder phase B input. Input pin associated with quadrature decoder phase B. | I |

## Table 10-23. FTM 3 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| FTM_CLKIN[1:0] | EXTCLK | External clock. FTM external clock can be selected to drive the FTM counter. | I |
| FTM3_CH[7:0] | CHn | FTM channel (n), where n can be 7-0 | I/O |
| FTM3_FLT[3:0] | FAULTj | Fault input (j), where j can be 3-0 | I |

## Table 10-24. CMT Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| CMT_IRO | CMT_IRO | Infrared Output | O |

## Table 10-25. PDB 0 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| PDB0_EXTRG | EXTRG | External Trigger Input Source<br>If the PDB is enabled and external trigger input source is selected, a positive edge on the EXTRG signal resets and starts the counter. | I |

**Table 10-26. LPTMR 0 Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| LPTMR0_ALT[:1] | LPTMR_ALT*n* | Pulse Counter Input pin | I |

**Table 10-27. RTC Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| VBAT | — | Backup battery supply for RTC and VBAT register file | I |
| RTC_CLKOUT | RTC_CLKOUT | 1 Hz square-wave output | O |

## 10.4.8 Communication Interfaces

**Table 10-28. USB FS OTG Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| USB0_DM | usb_dm | USB D- analog data signal on the USB bus. | I/O |
| USB0_DP | usb_dp | USB D+ analog data signal on the USB bus. | I/O |
| USB_CLKIN | — | Alternate USB clock input | I |
| USB_SOF_OUT | — | USB start of frame signal. Can be used to make the USB start of frame available for external synchronization. | O |

**Table 10-29. USB VREG Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| VOUT33 | reg33_out | Regulator output voltage | O |
| VREGIN | reg33_in | Unregulated power supply | I |

**Table 10-30. CAN 0 Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| CAN0_RX | CAN Rx | CAN Receive Pin | Input |
| CAN0_TX | CAN Tx | CAN Transmit Pin | Output |

## Table 10-31. SPI 0 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| SPI0_PCS0 | PCS0/$\overline{SS}$ | Peripheral Chip Select 0 output | I/O |
| SPI0_PCS[3:1] | PCS[3:1] | Peripheral Chip Select 1 – 3 | O |
| SPI0_PCS4 | PCS4 | Peripheral Chip Select 4 | O |
| SPI0_PCS5 | PCS5/ $\overline{PCSS}$ | Peripheral Chip Select 5 /Peripheral Chip Select Strobe | O |
| SPI0_SIN | SIN | Serial Data In | I |
| SPI0_SOUT | SOUT | Serial Data Out | O |
| SPI0_SCK | SCK | Master mode: Serial Clock (output) | I/O |

## Table 10-32. SPI 1 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| SPI1_PCS0 | PCS0/$\overline{SS}$ | Peripheral Chip Select 0 output | I/O |
| SPI1_SIN | SIN | Serial Data In | I |
| SPI1_SOUT | SOUT | Serial Data Out | O |
| SPI1_SCK | SCK | Master mode: Serial Clock (output) | I/O |

## Table 10-33. SPI 2 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| SPI2_PCS0 | PCS0/$\overline{SS}$ | Peripheral Chip Select 0 output | I/O |
| SPI2_PCS1 | PCS[3:1] | Peripheral Chip Select 1 – 3 | O |
| SPI2_SIN | SIN | Serial Data In | I |
| SPI2_SOUT | SOUT | Serial Data Out | O |
| SPI2_SCK | SCK | Master mode: Serial Clock (output) | I/O |

## Table 10-34. $I^2C$ 0 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| I2C0_SCL | SCL | Bidirectional serial clock line of the $I^2C$ system. | I/O |
| I2C0_SDA | SDA | Bidirectional serial data line of the $I^2C$ system. | I/O |

## Table 10-35. UART 0 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| $\overline{UART0\_CTS}$ | CTS | Clear to send | I |
| $\overline{UART0\_RTS}$ | $\overline{RTS}$ | Request to send | O |

*Table continues on the next page...*

## Table 10-35. UART 0 Signal Descriptions (continued)

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| UART0_TX | TXD | Transmit data | O |
| UART0_RX | RXD | Receive data | I |

## Table 10-36. UART 1 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| UART1_CTS | CTS | Clear to send | I |
| UART1_RTS | RTS | Request to send | O |
| UART1_TX | TXD | Transmit data | O |
| UART1_RX | RXD | Receive data | I |

## Table 10-37. UART 2 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| UART2_CTS | CTS | Clear to send | I |
| UART2_RTS | RTS | Request to send | O |
| UART2_TX | TXD | Transmit data | O |
| UART2_RX | RXD | Receive data | I |

## Table 10-38. UART 3 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| UART3_CTS | CTS | Clear to send | I |
| UART3_RTS | RTS | Request to send | O |
| UART3_TX | TXD | Transmit data | O |
| UART3_RX | RXD | Receive data | I |

## Table 10-39. UART 4 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| UART4_CTS | CTS | Clear to send | I |
| UART4_RTS | RTS | Request to send | O |
| UART4_TX | TXD | Transmit data | O |
| UART4_RX | RXD | Receive data | I |

## Table 10-40. UART 5 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| UART5_CTS | CTS | Clear to send | I |
| UART5_RTS | RTS | Request to send | O |
| UART5_TX | TXD | Transmit data | O |
| UART5_RX | RXD | Receive data | I |

## Table 10-41. SDHC Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| SDHC0_CLKIN | — | SDHC clock input | I |
| SDHC0_DCLK | SDHC_DCLK | Generated clock used to drive the MMC, SD, SDIO or CE-ATA cards. | O |
| SDHC0_CMD | SDHC_CMD | Send commands to and receive responses from the card. | I/O |
| SDHC0_D0 | SDHC_D0 | DAT0 line or busy-state detect | I/O |
| SDHC0_D1 | SDHC_D1 | 8-bit mode: DAT1 line<br>4-bit mode: DAT1 line or interrupt detect<br>1-bit mode: Interrupt detect | I/O |
| SDHC0_D2 | SDHC_D2 | 4-/8-bit mode: DAT2 line or read wait<br>1-bit mode: Read wait | I/O |
| SDHC0_D3 | SDHC_D3 | 4-/8-bit mode: DAT3 line or configured as card detection pin<br>1-bit mode: May be configured as card detection pin | I/O |
| SDHC0_D4 | SDHC_D4 | DAT4 line in 8-bit mode<br>Not used in other modes | I/O |
| SDHC0_D5 | SDHC_D5 | DAT5 line in 8-bit mode<br>Not used in other modes | I/O |
| SDHC0_D6 | SDHC_D6 | DAT6 line in 8-bit mode<br>Not used in other modes | I/O |
| SDHC0_D7 | SDHC_D7 | DAT7 line in 8-bit mode<br>Not used in other modes | I/O |

## Table 10-42. $I^2S0$ Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| I2S0_MCLK | SAI_MCLK | Audio Master Clock | I/O |
| I2S0_RX_BCLK | SAI_RX_BCLK | Receive Bit Clock | I/O |
| I2S0_RX_FS | SAI_RX_SYNC | Receive Frame Sync | I/O |
| I2S0_RXD | SAI_RX_DATA[1:0] | Receive Data | I |
| I2S0_TX_BCLK | SAI_TX_BCLK | Transmit Bit Clock | I/O |

*Table continues on the next page...*

**Table 10-42. I²S0 Signal Descriptions (continued)**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| I2S0_TX_FS | SAI_TX_SYNC | Transmit Frame Sync | I/O |
| I2S0_TXD | SAI_TX_DATA[1:0] | Transmit Data | O |

## 10.4.9 Human-Machine Interfaces (HMI)

**Table 10-43. GPIO Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| PTA[31:0][1] | PORTA31–PORTA0 | General-purpose input/output | I/O |
| PTB[31:0][1] | PORTB31–PORTB0 | General-purpose input/output | I/O |
| PTC[31:0][1] | PORTC31–PORTC0 | General-purpose input/output | I/O |
| PTD[31:0][1] | PORTD31–PORTD0 | General-purpose input/output | I/O |
| PTE[31:0][1] | PORTE31–PORTE0 | General-purpose input/output | I/O |

1. The available GPIO pins depends on the specific package. See the signal multiplexing section for which exact GPIO signals are available.

# Chapter 11
# Port control and interrupts (PORT)

## 11.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

## 11.2 Overview

The port control and interrupt (PORT) module provides support for port control, digital filtering, and external interrupt functions. Most functions can be configured independently for each pin in the 32-bit port and affect the pin regardless of its pin muxing state.

There is one instance of the PORT module for each port. Not all pins within each port are implemented on a specific device.

### 11.2.1 Features

The PORT module has the following features:

- Pin interrupt
  - Interrupt flag and enable registers for each pin
  - Support for edge sensitive (rising, falling, both) or level sensitive (low, high) configured per pin
  - Support for interrupt or DMA request configured per pin
  - Asynchronous wakeup in Low-Power modes
  - Pin interrupt is functional in all digital Pin Muxing modes
- Digital input filter on selected pins

- Digital input filter for each pin, usable by any digital peripheral muxed onto the pin
- Individual enable or bypass control field per pin
- Selectable clock source for digital input filter with a five bit resolution on filter size
- Functional in all digital Pin Muxing modes

- Port control
  - Individual pull control fields with pullup, pulldown, and pull-disable support
  - Individual drive strength field supporting high and low drive strength
  - Individual slew rate field supporting fast and slow slew rates
  - Individual input passive filter field supporting enable and disable of the individual input passive filter
  - Individual open drain field supporting enable and disable of the individual open drain output
  - Individual mux control field supporting analog or pin disabled, GPIO, and up to six chip-specific digital functions
  - Pad configuration fields are functional in all digital Pin Muxing modes

## 11.2.2 Modes of operation

### 11.2.2.1 Run mode

In Run mode, the PORT operates normally.

### 11.2.2.2 Wait mode

In Wait mode, PORT continues to operate normally and may be configured to exit the Low-Power mode if an enabled interrupt is detected. DMA requests are still generated during the Wait mode, but do not cause an exit from the Low-Power mode.

### 11.2.2.3 Stop mode

In Stop mode, the PORT can be configured to exit the Low-Power mode via an asynchronous wakeup signal if an enabled interrupt is detected.

In Stop mode, the digital input filters are bypassed unless they are configured to run from the 1 kHz LPO clock source.

#### 11.2.2.4 Debug mode

In Debug mode, PORT operates normally.

## 11.3 External signal description

The following table describes the PORT external signal.

**Table 11-1. Signal properties**

| Name | Function | I/O | Reset | Pull |
|---|---|---|---|---|
| PORTx[31:0] | External interrupt | I/O | 0 | - |

**NOTE**

Not all pins within each port are implemented on each device.

## 11.4 Detailed signal description

The following table contains the detailed signal description for the PORT interface.

**Table 11-2. PORT interface—detailed signal description**

| Signal | I/O | Description | |
|---|---|---|---|
| PORTx[31:0] | I/O | External interrupt. | |
| | | State meaning | Asserted—pin is logic one.<br>Negated—pin is logic zero. |
| | | Timing | Assertion—may occur at any time and can assert asynchronously to the system clock.<br>Negation—may occur at any time and can assert asynchronously to the system clock. |

## 11.5 Memory map and register definition

Any read or write access to the PORT memory space that is outside the valid memory map results in a bus error. All register accesses complete with zero wait states.

## PORT memory map

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4004_9000 | Pin Control Register n (PORTA_PCR0) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9004 | Pin Control Register n (PORTA_PCR1) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9008 | Pin Control Register n (PORTA_PCR2) | 32 | R/W | See section | 11.5.1/260 |
| 4004_900C | Pin Control Register n (PORTA_PCR3) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9010 | Pin Control Register n (PORTA_PCR4) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9014 | Pin Control Register n (PORTA_PCR5) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9018 | Pin Control Register n (PORTA_PCR6) | 32 | R/W | See section | 11.5.1/260 |
| 4004_901C | Pin Control Register n (PORTA_PCR7) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9020 | Pin Control Register n (PORTA_PCR8) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9024 | Pin Control Register n (PORTA_PCR9) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9028 | Pin Control Register n (PORTA_PCR10) | 32 | R/W | See section | 11.5.1/260 |
| 4004_902C | Pin Control Register n (PORTA_PCR11) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9030 | Pin Control Register n (PORTA_PCR12) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9034 | Pin Control Register n (PORTA_PCR13) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9038 | Pin Control Register n (PORTA_PCR14) | 32 | R/W | See section | 11.5.1/260 |
| 4004_903C | Pin Control Register n (PORTA_PCR15) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9040 | Pin Control Register n (PORTA_PCR16) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9044 | Pin Control Register n (PORTA_PCR17) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9048 | Pin Control Register n (PORTA_PCR18) | 32 | R/W | See section | 11.5.1/260 |
| 4004_904C | Pin Control Register n (PORTA_PCR19) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9050 | Pin Control Register n (PORTA_PCR20) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9054 | Pin Control Register n (PORTA_PCR21) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9058 | Pin Control Register n (PORTA_PCR22) | 32 | R/W | See section | 11.5.1/260 |
| 4004_905C | Pin Control Register n (PORTA_PCR23) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9060 | Pin Control Register n (PORTA_PCR24) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9064 | Pin Control Register n (PORTA_PCR25) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9068 | Pin Control Register n (PORTA_PCR26) | 32 | R/W | See section | 11.5.1/260 |
| 4004_906C | Pin Control Register n (PORTA_PCR27) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9070 | Pin Control Register n (PORTA_PCR28) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9074 | Pin Control Register n (PORTA_PCR29) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9078 | Pin Control Register n (PORTA_PCR30) | 32 | R/W | See section | 11.5.1/260 |
| 4004_907C | Pin Control Register n (PORTA_PCR31) | 32 | R/W | See section | 11.5.1/260 |
| 4004_9080 | Global Pin Control Low Register (PORTA_GPCLR) | 32 | W (always reads 0) | 0000_0000h | 11.5.2/262 |
| 4004_9084 | Global Pin Control High Register (PORTA_GPCHR) | 32 | W (always reads 0) | 0000_0000h | 11.5.3/263 |
| 4004_90A0 | Interrupt Status Flag Register (PORTA_ISFR) | 32 | w1c | 0000_0000h | 11.5.4/264 |

*Table continues on the next page...*

## PORT memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4004_90C0 | Digital Filter Enable Register (PORTA_DFER) | 32 | R/W | 0000_0000h | 11.5.5/264 |
| 4004_90C4 | Digital Filter Clock Register (PORTA_DFCR) | 32 | R/W | 0000_0000h | 11.5.6/265 |
| 4004_90C8 | Digital Filter Width Register (PORTA_DFWR) | 32 | R/W | 0000_0000h | 11.5.7/265 |
| 4004_A000 | Pin Control Register n (PORTB_PCR0) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A004 | Pin Control Register n (PORTB_PCR1) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A008 | Pin Control Register n (PORTB_PCR2) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A00C | Pin Control Register n (PORTB_PCR3) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A010 | Pin Control Register n (PORTB_PCR4) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A014 | Pin Control Register n (PORTB_PCR5) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A018 | Pin Control Register n (PORTB_PCR6) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A01C | Pin Control Register n (PORTB_PCR7) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A020 | Pin Control Register n (PORTB_PCR8) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A024 | Pin Control Register n (PORTB_PCR9) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A028 | Pin Control Register n (PORTB_PCR10) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A02C | Pin Control Register n (PORTB_PCR11) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A030 | Pin Control Register n (PORTB_PCR12) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A034 | Pin Control Register n (PORTB_PCR13) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A038 | Pin Control Register n (PORTB_PCR14) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A03C | Pin Control Register n (PORTB_PCR15) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A040 | Pin Control Register n (PORTB_PCR16) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A044 | Pin Control Register n (PORTB_PCR17) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A048 | Pin Control Register n (PORTB_PCR18) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A04C | Pin Control Register n (PORTB_PCR19) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A050 | Pin Control Register n (PORTB_PCR20) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A054 | Pin Control Register n (PORTB_PCR21) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A058 | Pin Control Register n (PORTB_PCR22) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A05C | Pin Control Register n (PORTB_PCR23) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A060 | Pin Control Register n (PORTB_PCR24) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A064 | Pin Control Register n (PORTB_PCR25) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A068 | Pin Control Register n (PORTB_PCR26) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A06C | Pin Control Register n (PORTB_PCR27) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A070 | Pin Control Register n (PORTB_PCR28) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A074 | Pin Control Register n (PORTB_PCR29) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A078 | Pin Control Register n (PORTB_PCR30) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A07C | Pin Control Register n (PORTB_PCR31) | 32 | R/W | See section | 11.5.1/260 |
| 4004_A080 | Global Pin Control Low Register (PORTB_GPCLR) | 32 | W (always reads 0) | 0000_0000h | 11.5.2/262 |

*Table continues on the next page...*

## PORT memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4004_A084 | Global Pin Control High Register (PORTB_GPCHR) | 32 | W (always reads 0) | 0000_0000h | 11.5.3/263 |
| 4004_A0A0 | Interrupt Status Flag Register (PORTB_ISFR) | 32 | w1c | 0000_0000h | 11.5.4/264 |
| 4004_A0C0 | Digital Filter Enable Register (PORTB_DFER) | 32 | R/W | 0000_0000h | 11.5.5/264 |
| 4004_A0C4 | Digital Filter Clock Register (PORTB_DFCR) | 32 | R/W | 0000_0000h | 11.5.6/265 |
| 4004_A0C8 | Digital Filter Width Register (PORTB_DFWR) | 32 | R/W | 0000_0000h | 11.5.7/265 |
| 4004_B000 | Pin Control Register n (PORTC_PCR0) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B004 | Pin Control Register n (PORTC_PCR1) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B008 | Pin Control Register n (PORTC_PCR2) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B00C | Pin Control Register n (PORTC_PCR3) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B010 | Pin Control Register n (PORTC_PCR4) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B014 | Pin Control Register n (PORTC_PCR5) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B018 | Pin Control Register n (PORTC_PCR6) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B01C | Pin Control Register n (PORTC_PCR7) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B020 | Pin Control Register n (PORTC_PCR8) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B024 | Pin Control Register n (PORTC_PCR9) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B028 | Pin Control Register n (PORTC_PCR10) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B02C | Pin Control Register n (PORTC_PCR11) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B030 | Pin Control Register n (PORTC_PCR12) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B034 | Pin Control Register n (PORTC_PCR13) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B038 | Pin Control Register n (PORTC_PCR14) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B03C | Pin Control Register n (PORTC_PCR15) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B040 | Pin Control Register n (PORTC_PCR16) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B044 | Pin Control Register n (PORTC_PCR17) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B048 | Pin Control Register n (PORTC_PCR18) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B04C | Pin Control Register n (PORTC_PCR19) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B050 | Pin Control Register n (PORTC_PCR20) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B054 | Pin Control Register n (PORTC_PCR21) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B058 | Pin Control Register n (PORTC_PCR22) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B05C | Pin Control Register n (PORTC_PCR23) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B060 | Pin Control Register n (PORTC_PCR24) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B064 | Pin Control Register n (PORTC_PCR25) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B068 | Pin Control Register n (PORTC_PCR26) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B06C | Pin Control Register n (PORTC_PCR27) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B070 | Pin Control Register n (PORTC_PCR28) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B074 | Pin Control Register n (PORTC_PCR29) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B078 | Pin Control Register n (PORTC_PCR30) | 32 | R/W | See section | 11.5.1/260 |

*Table continues on the next page...*

## PORT memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4004_B07C | Pin Control Register n (PORTC_PCR31) | 32 | R/W | See section | 11.5.1/260 |
| 4004_B080 | Global Pin Control Low Register (PORTC_GPCLR) | 32 | W (always reads 0) | 0000_0000h | 11.5.2/262 |
| 4004_B084 | Global Pin Control High Register (PORTC_GPCHR) | 32 | W (always reads 0) | 0000_0000h | 11.5.3/263 |
| 4004_B0A0 | Interrupt Status Flag Register (PORTC_ISFR) | 32 | w1c | 0000_0000h | 11.5.4/264 |
| 4004_B0C0 | Digital Filter Enable Register (PORTC_DFER) | 32 | R/W | 0000_0000h | 11.5.5/264 |
| 4004_B0C4 | Digital Filter Clock Register (PORTC_DFCR) | 32 | R/W | 0000_0000h | 11.5.6/265 |
| 4004_B0C8 | Digital Filter Width Register (PORTC_DFWR) | 32 | R/W | 0000_0000h | 11.5.7/265 |
| 4004_C000 | Pin Control Register n (PORTD_PCR0) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C004 | Pin Control Register n (PORTD_PCR1) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C008 | Pin Control Register n (PORTD_PCR2) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C00C | Pin Control Register n (PORTD_PCR3) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C010 | Pin Control Register n (PORTD_PCR4) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C014 | Pin Control Register n (PORTD_PCR5) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C018 | Pin Control Register n (PORTD_PCR6) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C01C | Pin Control Register n (PORTD_PCR7) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C020 | Pin Control Register n (PORTD_PCR8) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C024 | Pin Control Register n (PORTD_PCR9) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C028 | Pin Control Register n (PORTD_PCR10) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C02C | Pin Control Register n (PORTD_PCR11) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C030 | Pin Control Register n (PORTD_PCR12) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C034 | Pin Control Register n (PORTD_PCR13) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C038 | Pin Control Register n (PORTD_PCR14) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C03C | Pin Control Register n (PORTD_PCR15) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C040 | Pin Control Register n (PORTD_PCR16) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C044 | Pin Control Register n (PORTD_PCR17) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C048 | Pin Control Register n (PORTD_PCR18) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C04C | Pin Control Register n (PORTD_PCR19) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C050 | Pin Control Register n (PORTD_PCR20) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C054 | Pin Control Register n (PORTD_PCR21) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C058 | Pin Control Register n (PORTD_PCR22) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C05C | Pin Control Register n (PORTD_PCR23) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C060 | Pin Control Register n (PORTD_PCR24) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C064 | Pin Control Register n (PORTD_PCR25) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C068 | Pin Control Register n (PORTD_PCR26) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C06C | Pin Control Register n (PORTD_PCR27) | 32 | R/W | See section | 11.5.1/260 |

*Table continues on the next page...*

## PORT memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4004_C070 | Pin Control Register n (PORTD_PCR28) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C074 | Pin Control Register n (PORTD_PCR29) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C078 | Pin Control Register n (PORTD_PCR30) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C07C | Pin Control Register n (PORTD_PCR31) | 32 | R/W | See section | 11.5.1/260 |
| 4004_C080 | Global Pin Control Low Register (PORTD_GPCLR) | 32 | W (always reads 0) | 0000_0000h | 11.5.2/262 |
| 4004_C084 | Global Pin Control High Register (PORTD_GPCHR) | 32 | W (always reads 0) | 0000_0000h | 11.5.3/263 |
| 4004_C0A0 | Interrupt Status Flag Register (PORTD_ISFR) | 32 | w1c | 0000_0000h | 11.5.4/264 |
| 4004_C0C0 | Digital Filter Enable Register (PORTD_DFER) | 32 | R/W | 0000_0000h | 11.5.5/264 |
| 4004_C0C4 | Digital Filter Clock Register (PORTD_DFCR) | 32 | R/W | 0000_0000h | 11.5.6/265 |
| 4004_C0C8 | Digital Filter Width Register (PORTD_DFWR) | 32 | R/W | 0000_0000h | 11.5.7/265 |
| 4004_D000 | Pin Control Register n (PORTE_PCR0) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D004 | Pin Control Register n (PORTE_PCR1) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D008 | Pin Control Register n (PORTE_PCR2) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D00C | Pin Control Register n (PORTE_PCR3) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D010 | Pin Control Register n (PORTE_PCR4) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D014 | Pin Control Register n (PORTE_PCR5) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D018 | Pin Control Register n (PORTE_PCR6) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D01C | Pin Control Register n (PORTE_PCR7) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D020 | Pin Control Register n (PORTE_PCR8) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D024 | Pin Control Register n (PORTE_PCR9) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D028 | Pin Control Register n (PORTE_PCR10) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D02C | Pin Control Register n (PORTE_PCR11) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D030 | Pin Control Register n (PORTE_PCR12) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D034 | Pin Control Register n (PORTE_PCR13) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D038 | Pin Control Register n (PORTE_PCR14) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D03C | Pin Control Register n (PORTE_PCR15) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D040 | Pin Control Register n (PORTE_PCR16) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D044 | Pin Control Register n (PORTE_PCR17) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D048 | Pin Control Register n (PORTE_PCR18) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D04C | Pin Control Register n (PORTE_PCR19) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D050 | Pin Control Register n (PORTE_PCR20) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D054 | Pin Control Register n (PORTE_PCR21) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D058 | Pin Control Register n (PORTE_PCR22) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D05C | Pin Control Register n (PORTE_PCR23) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D060 | Pin Control Register n (PORTE_PCR24) | 32 | R/W | See section | 11.5.1/260 |

*Table continues on the next page...*

## PORT memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4004_D064 | Pin Control Register n (PORTE_PCR25) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D068 | Pin Control Register n (PORTE_PCR26) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D06C | Pin Control Register n (PORTE_PCR27) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D070 | Pin Control Register n (PORTE_PCR28) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D074 | Pin Control Register n (PORTE_PCR29) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D078 | Pin Control Register n (PORTE_PCR30) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D07C | Pin Control Register n (PORTE_PCR31) | 32 | R/W | See section | 11.5.1/260 |
| 4004_D080 | Global Pin Control Low Register (PORTE_GPCLR) | 32 | W (always reads 0) | 0000_0000h | 11.5.2/262 |
| 4004_D084 | Global Pin Control High Register (PORTE_GPCHR) | 32 | W (always reads 0) | 0000_0000h | 11.5.3/263 |
| 4004_D0A0 | Interrupt Status Flag Register (PORTE_ISFR) | 32 | w1c | 0000_0000h | 11.5.4/264 |
| 4004_D0C0 | Digital Filter Enable Register (PORTE_DFER) | 32 | R/W | 0000_0000h | 11.5.5/264 |
| 4004_D0C4 | Digital Filter Clock Register (PORTE_DFCR) | 32 | R/W | 0000_0000h | 11.5.6/265 |
| 4004_D0C8 | Digital Filter Width Register (PORTE_DFWR) | 32 | R/W | 0000_0000h | 11.5.7/265 |

### 11.5.1 Pin Control Register n (PORTx_PCR*n*)

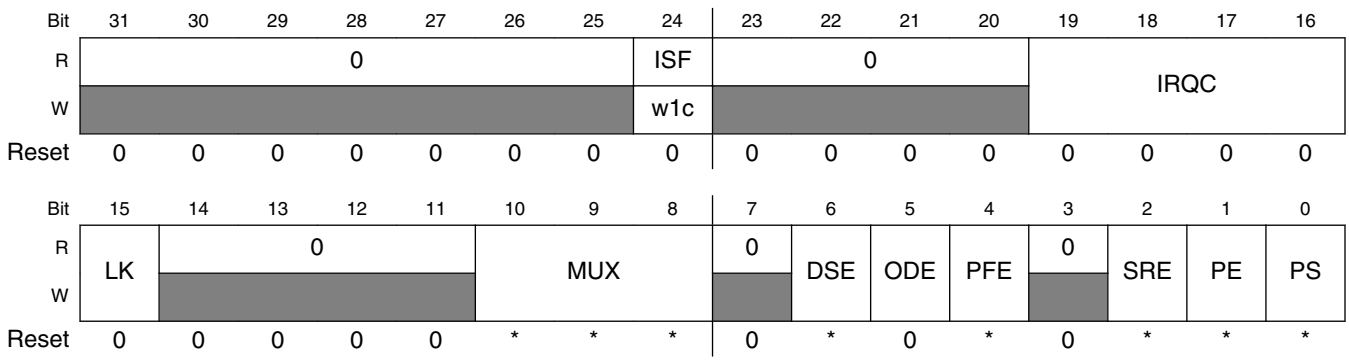
**NOTE**

Refer to the Signal Multiplexing and Pin Assignment chapter for the reset value of this device.

See the GPIO Configuration section for details on the available functions for each pin.

Do not modify pin configuration registers associated with pins not available in your selected package. All un-bonded pins not available in your package will default to DISABLE state for lowest power consumption.

Address: Base address + 0h offset + (4d × i), where i=0d to 31d




* Notes:
- MUX field: Varies by port. See Signal Multiplexing and Signal Descriptions chapter for reset values per port.
- DSE field: Varies by port. See Signal Multiplexing and Signal Descriptions chapter for reset values per port.
- PFE field: Varies by port. See Signal Multiplexing and Signal Descriptions chapter for reset values per port.
- SRE field: Varies by port. See Signal Multiplexing and Signal Descriptions chapter for reset values per port.
- PE field: Varies by port. See Signal Multiplexing and Signal Descriptions chapter for reset values per port.
- PS field: Varies by port. See Signal Multiplexing and Signal Descriptions chapter for reset values per port.

**PORTx_PCR*n* field descriptions**

| Field | Description |
|---|---|
| 31–25<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 24<br>ISF | Interrupt Status Flag<br><br>The pin interrupt configuration is valid in all digital pin muxing modes. |

*Table continues on the next page...*

## PORTx_PCR*n* field descriptions (continued)

| Field | Description |
|---|---|
| | 0 Configured interrupt is not detected.<br>1 Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic one is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |
| 23–20<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 19–16<br>IRQC | Interrupt Configuration<br>The pin interrupt configuration is valid in all digital pin muxing modes. The corresponding pin is configured to generate interrupt/DMA request as follows:<br>0000 Interrupt/DMA request disabled.<br>0001 DMA request on rising edge.<br>0010 DMA request on falling edge.<br>0011 DMA request on either edge.<br>1000 Interrupt when logic zero.<br>1001 Interrupt on rising edge.<br>1010 Interrupt on falling edge.<br>1011 Interrupt on either edge.<br>1100 Interrupt when logic one.<br>Others Reserved. |
| 15<br>LK | Lock Register<br>0 Pin Control Register fields [15:0] are not locked.<br>1 Pin Control Register fields [15:0] are locked and cannot be updated until the next system reset. |
| 14–11<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 10–8<br>MUX | Pin Mux Control<br>Not all pins support all pin muxing slots. Unimplemented pin muxing slots are reserved and may result in configuring the pin for a different pin muxing slot.<br>The corresponding pin is configured in the following pin muxing slot as follows:<br>000 Pin disabled (analog).<br>001 Alternative 1 (GPIO).<br>010 Alternative 2 (chip-specific).<br>011 Alternative 3 (chip-specific).<br>100 Alternative 4 (chip-specific).<br>101 Alternative 5 (chip-specific).<br>110 Alternative 6 (chip-specific).<br>111 Alternative 7 (chip-specific). |
| 7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6<br>DSE | Drive Strength Enable<br>Drive strength configuration is valid in all digital pin muxing modes. |

*Table continues on the next page...*

**PORTx_PCR*n* field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Low drive strength is configured on the corresponding pin, if pin is configured as a digital output.<br>1 High drive strength is configured on the corresponding pin, if pin is configured as a digital output. |
| 5<br>ODE | Open Drain Enable<br>Open drain configuration is valid in all digital pin muxing modes.<br>0 Open drain output is disabled on the corresponding pin.<br>1 Open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. |
| 4<br>PFE | Passive Filter Enable<br>Passive filter configuration is valid in all digital pin muxing modes.<br>0 Passive input filter is disabled on the corresponding pin.<br>1 Passive input filter is enabled on the corresponding pin, if the pin is configured as a digital input. Refer to the device data sheet for filter characteristics. |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>SRE | Slew Rate Enable<br>Slew rate configuration is valid in all digital pin muxing modes.<br>0 Fast slew rate is configured on the corresponding pin, if the pin is configured as a digital output.<br>1 Slow slew rate is configured on the corresponding pin, if the pin is configured as a digital output. |
| 1<br>PE | Pull Enable<br>Pull configuration is valid in all digital pin muxing modes.<br>0 Internal pullup or pulldown resistor is not enabled on the corresponding pin.<br>1 Internal pullup or pulldown resistor is enabled on the corresponding pin, if the pin is configured as a digital input. |
| 0<br>PS | Pull Select<br>Pull configuration is valid in all digital pin muxing modes.<br>0 Internal pulldown resistor is enabled on the corresponding pin, if the corresponding Port Pull Enable field is set.<br>1 Internal pullup resistor is enabled on the corresponding pin, if the corresponding Port Pull Enable field is set. |

## 11.5.2 Global Pin Control Low Register (PORTx_GPCLR)

Only 32-bit writes are supported to this register.

Address: Base address + 80h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | 0 | | | | | | | | | | | | | | | |
| W | GPWE | | | | | | | | | | | | | | | | GPWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PORTx_GPCLR field descriptions**

| Field | Description |
|---|---|
| 31–16<br>GPWE | Global Pin Write Enable<br><br>Selects which Pin Control Registers (15 through 0) bits [15:0] update with the value in GPWD. If a selected Pin Control Register is locked then the write to that register is ignored.<br><br>0 Corresponding Pin Control Register is not updated with the value in GPWD.<br>1 Corresponding Pin Control Register is updated with the value in GPWD. |
| 15–0<br>GPWD | Global Pin Write Data<br><br>Write value that is written to all Pin Control Registers bits [15:0] that are selected by GPWE. |

## 11.5.3 Global Pin Control High Register (PORTx_GPCHR)

Only 32-bit writes are supported to this register.

Address: Base address + 84h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | 0 | | | | | | | | | | | | | | | |
| W | GPWE | | | | | | | | | | | | | | | | GPWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PORTx_GPCHR field descriptions**

| Field | Description |
|---|---|
| 31–16<br>GPWE | Global Pin Write Enable<br><br>Selects which Pin Control Registers (31 through 16) bits [15:0] update with the value in GPWD. If a selected Pin Control Register is locked then the write to that register is ignored.<br><br>0 Corresponding Pin Control Register is not updated with the value in GPWD.<br>1 Corresponding Pin Control Register is updated with the value in GPWD. |
| 15–0<br>GPWD | Global Pin Write Data<br><br>Write value that is written to all Pin Control Registers bits [15:0] that are selected by GPWE. |

## 11.5.4 Interrupt Status Flag Register (PORTx_ISFR)

The pin interrupt configuration is valid in all digital pin muxing modes. The Interrupt Status Flag for each pin is also visible in the corresponding Pin Control Register, and each flag can be cleared in either location.

Address: Base address + A0h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ISF | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | w1c | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PORTx_ISFR field descriptions**

| Field | Description |
|---|---|
| 31–0<br>ISF | Interrupt Status Flag<br><br>Each bit in the field indicates the detection of the configured interrupt of the same number as the field.<br><br>0 Configured interrupt is not detected.<br>1 Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic one is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |

## 11.5.5 Digital Filter Enable Register (PORTx_DFER)

The corresponding bit is read only for pins that do not support a digital filter. Refer to the Chapter of Signal Multiplexing and Signal Descriptions for the pins that support digital filter.

The digital filter configuration is valid in all digital pin muxing modes.

Address: Base address + C0h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DFE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PORTx_DFER field descriptions**

| Field | Description |
|---|---|
| 31–0<br>DFE | Digital Filter Enable |

**PORTx_DFER field descriptions (continued)**

| Field | Description |
|---|---|
| | The digital filter configuration is valid in all digital pin muxing modes. The output of each digital filter is reset to zero at system reset and whenever the digital filter is disabled. Each bit in the field enables the digital filter of the same number as the field.<br><br>0 Digital filter is disabled on the corresponding pin and output of the digital filter is reset to zero.<br>1 Digital filter is enabled on the corresponding pin, if the pin is configured as a digital input. |

## 11.5.6 Digital Filter Clock Register (PORTx_DFCR)

This register is read only for ports that do not support a digital filter.

The digital filter configuration is valid in all digital pin muxing modes.

Address: Base address + C4h offset

| Bit | 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 |

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

**PORTx_DFCR field descriptions**

| Field | Description |
|---|---|
| 31–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>CS | Clock Source<br><br>The digital filter configuration is valid in all digital pin muxing modes. Configures the clock source for the digital input filters. Changing the filter clock source must be done only when all digital filters are disabled.<br><br>0 Digital filters are clocked by the bus clock.<br>1 Digital filters are clocked by the 1 kHz LPO clock. |

## 11.5.7 Digital Filter Width Register (PORTx_DFWR)

This register is read only for ports that do not support a digital filter.

The digital filter configuration is valid in all digital pin muxing modes.

Address: Base address + C8h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | FILT | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PORTx_DFWR field descriptions**

| Field | Description |
|---|---|
| 31–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4–0<br>FILT | Filter Length<br><br>The digital filter configuration is valid in all digital pin muxing modes. Configures the maximum size of the glitches, in clock cycles, that the digital filter absorbs for the enabled digital filters. Glitches that are longer than this register setting will pass through the digital filter, and glitches that are equal to or less than this register setting are filtered. Changing the filter length must be done only after all filters are disabled. |

# 11.6 Functional description

## 11.6.1 Pin control

Each port pin has a corresponding pin control register, PORT_PCRn, associated with it.

The upper half of the pin control register configures the pin's capability to either interrupt the CPU or request a DMA transfer, on a rising/falling edge or both edges as well as a logic level occurring on the port pin. It also includes a flag to indicate that an interrupt has occurred.

The lower half of the pin control register configures the following functions for each pin within the 32-bit port.

- Pullup or pulldown enable
- Drive strength and slew rate configuration
- Open drain enable
- Passive input filter enable
- Pin Muxing mode

The functions apply across all digital Pin Muxing modes and individual peripherals do not override the configuration in the pin control register. For example, if an $I^2C$ function is enabled on a pin, that does not override the pullup or open drain configuration for that pin.

When the Pin Muxing mode is configured for analog or is disabled, all the digital functions on that pin are disabled. This includes the pullup and pulldown enables, digital output buffer enable, digital input buffer enable, and passive filter enable.

A lock field also exists that allows the configuration for each pin to be locked until the next system reset. When locked, writes to the lower half of that pin control register are ignored, although a bus error is not generated on an attempted write to a locked register.

The configuration of each pin control register is retained when the PORT module is disabled.

### 11.6.2 Global pin control

The two global pin control registers allow a single register write to update the lower half of the pin control register on up to sixteen pins, all with the same value. Registers that are locked cannot be written using the global pin control registers.

The global pin control registers are designed to enable software to quickly configure multiple pins within the one port for the same peripheral function. However, the interrupt functions cannot be configured using the global pin control registers.

The global pin control registers are write-only registers, that always read as zero.

### 11.6.3 External interrupts

The external interrupt capability of the PORT module is available in all digital pin muxing modes provided the PORT module is enabled.

Each pin can be individually configured for any of the following external interrupt modes:

- Interrupt disabled, default out of reset
- Active high level sensitive interrupt
- Active low level sensitive interrupt
- Rising edge sensitive interrupt
- Falling edge sensitive interrupt
- Rising and falling edge sensitive interrupt
- Rising edge sensitive DMA request
- Falling edge sensitive DMA request
- Rising and falling edge sensitive DMA request

The interrupt status flag is set when the configured edge or level is detected on the pin or at the output of the digital input filter, if the digital input digital filter is enabled. When not in Stop mode, the input is first synchronized to the bus clock to detect the configured level or edge transition.

The PORT module generates a single interrupt that asserts when the interrupt status flag is set for any enabled interrupt for that port. The interrupt negates after the interrupt status flags for all enabled interrupts have been cleared by writing a logic 1 to the ISF flag in either the PORT_ISFR or PORT_PCRn registers.

The PORT module generates a single DMA request that asserts when the interrupt status flag is set for any enabled DMA request in that port. The DMA request negates after the DMA transfer is completed, because that clears the interrupt status flags for all enabled DMA requests.

During Stop mode, the interrupt status flag for any enabled interrupt is asynchronously set if the required level or edge is detected. This also generates an asynchronous wakeup signal to exit the Low-Power mode.

### 11.6.4 Digital filter

The digital filter capabilities of the PORT module are available in all digital Pin Muxing modes if the PORT module is enabled.

The clock used for all digital filters within one port can be configured between the bus clock or the 1 kHz LPO clock. This selection must be changed only when all digital filters for that port are disabled. If the digital filters for a port are configured to use the bus clock, then the digital filters are bypassed for the duration of Stop mode. While the digital filters are bypassed, the output of each digital filter always equals the input pin, but the internal state of the digital filters remains static and does not update due to any change on the input pin.

The filter width in clock size is the same for all enabled digital filters within one port and must be changed only when all digital filters for that port are disabled.

The output of each digital filter is logic zero after system reset and whenever a digital filter is disabled. After a digital filter is enabled, the input is synchronized to the filter clock, either the bus clock or the 1 kHz LPO clock. If the synchronized input and the output of the digital filter remain different for a number of filter clock cycles equal to the filter width register configuration, then the output of the digital filter updates to equal the synchronized filter input.

The minimum latency through a digital filter equals two or three filter clock cycles plus the filter width configuration register.

# Chapter 12
# System Integration Module (SIM)

## 12.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The System Integration Module (SIM) provides system control and chip configuration registers.

### 12.1.1 Features

Features of the SIM include:

- System clocking configuration
  - System clock divide values
  - Architectural clock gating control
  - USB clock selection and divide values
  - SDHC clock source selection
- Flash and system RAM size configuration
- USB regulator configuration
- FlexTimer external clock, hardware trigger, and fault source selection
- UART0 and UART1 receive/transmit source selection/configuration

## 12.2 Memory map and register definition

The SIM module contains many fields for selecting the clock source and dividers for various module clocks. See the Clock Distribution chapter for more information, including block diagrams and clock definitions.

**NOTE**

The SIM_SOPT1 and SIM_SOPT1CFG registers are located at a different base address than the other SIM registers.

**SIM memory map**

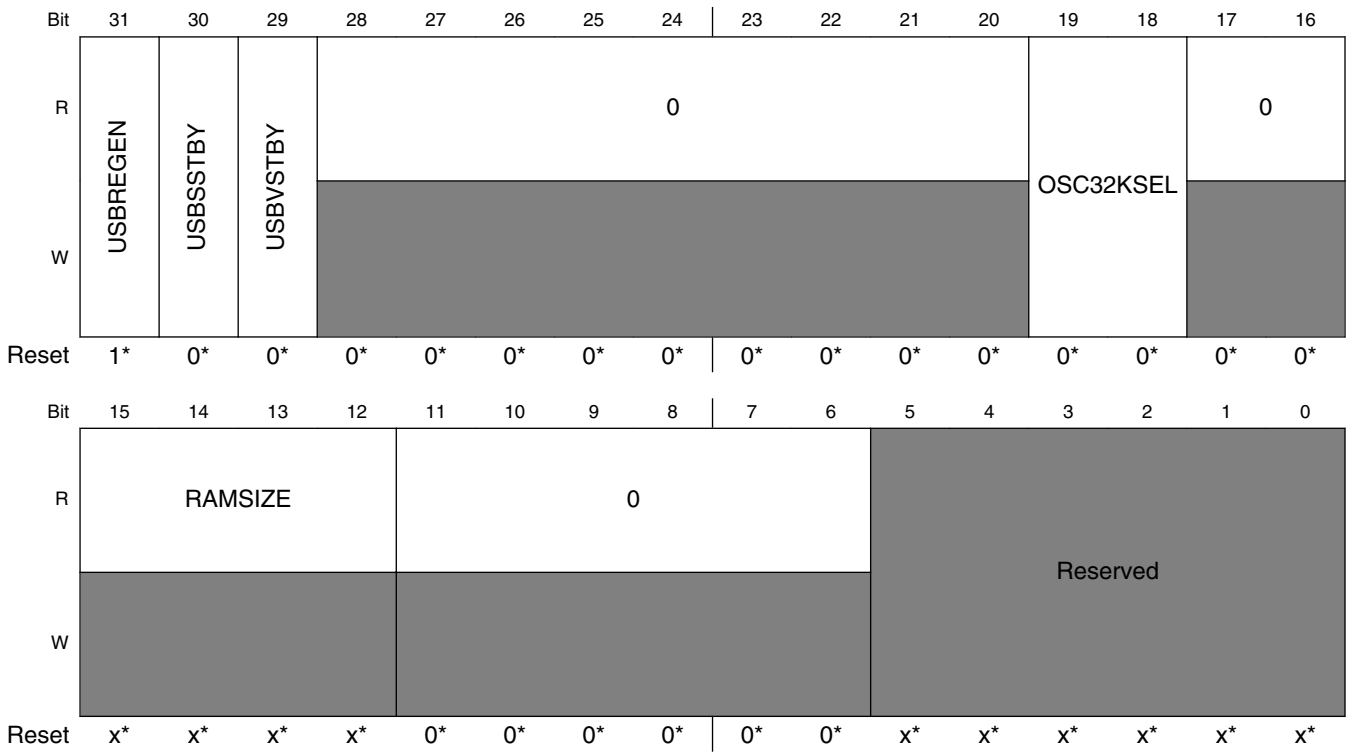
| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4004_7000 | System Options Register 1 (SIM_SOPT1) | 32 | R/W | See section | 12.2.1/271 |
| 4004_7004 | SOPT1 Configuration Register (SIM_SOPT1CFG) | 32 | R/W | 0000_0000h | 12.2.2/273 |
| 4004_8004 | System Options Register 2 (SIM_SOPT2) | 32 | R/W | 0000_1000h | 12.2.3/274 |
| 4004_800C | System Options Register 4 (SIM_SOPT4) | 32 | R/W | 0000_0000h | 12.2.4/276 |
| 4004_8010 | System Options Register 5 (SIM_SOPT5) | 32 | R/W | 0000_0000h | 12.2.5/279 |
| 4004_8018 | System Options Register 7 (SIM_SOPT7) | 32 | R/W | 0000_0000h | 12.2.6/281 |
| 4004_8024 | System Device Identification Register (SIM_SDID) | 32 | R | See section | 12.2.7/283 |
| 4004_8028 | System Clock Gating Control Register 1 (SIM_SCGC1) | 32 | R/W | 0000_0000h | 12.2.8/284 |
| 4004_802C | System Clock Gating Control Register 2 (SIM_SCGC2) | 32 | R/W | 0000_0000h | 12.2.9/285 |
| 4004_8030 | System Clock Gating Control Register 3 (SIM_SCGC3) | 32 | R/W | 0000_0000h | 12.2.10/287 |
| 4004_8034 | System Clock Gating Control Register 4 (SIM_SCGC4) | 32 | R/W | F010_0030h | 12.2.11/289 |
| 4004_8038 | System Clock Gating Control Register 5 (SIM_SCGC5) | 32 | R/W | 0004_0182h | 12.2.12/291 |
| 4004_803C | System Clock Gating Control Register 6 (SIM_SCGC6) | 32 | R/W | 4000_0001h | 12.2.13/293 |
| 4004_8040 | System Clock Gating Control Register 7 (SIM_SCGC7) | 32 | R/W | 0000_0006h | 12.2.14/296 |
| 4004_8044 | System Clock Divider Register 1 (SIM_CLKDIV1) | 32 | R/W | See section | 12.2.15/296 |
| 4004_8048 | System Clock Divider Register 2 (SIM_CLKDIV2) | 32 | R/W | 0000_0000h | 12.2.16/299 |
| 4004_804C | Flash Configuration Register 1 (SIM_FCFG1) | 32 | R | See section | 12.2.17/300 |
| 4004_8050 | Flash Configuration Register 2 (SIM_FCFG2) | 32 | R | See section | 12.2.18/303 |
| 4004_8054 | Unique Identification Register High (SIM_UIDH) | 32 | R | See section | 12.2.19/304 |
| 4004_8058 | Unique Identification Register Mid-High (SIM_UIDMH) | 32 | R | See section | 12.2.20/305 |
| 4004_805C | Unique Identification Register Mid Low (SIM_UIDML) | 32 | R | See section | 12.2.21/305 |
| 4004_8060 | Unique Identification Register Low (SIM_UIDL) | 32 | R | See section | 12.2.22/306 |

## 12.2.1 System Options Register 1 (SIM_SOPT1)

**NOTE**

The SOPT1 register is only reset on POR or LVD.

Address: 4004_7000h base + 0h offset = 4004_7000h




* Notes:

- Reset value loaded during System Reset from Flash IFR.
- x = Undefined at reset.

**SIM_SOPT1 field descriptions**

| Field | Description |
|---|---|
| 31<br>USBREGEN | USB voltage regulator enable<br><br>Controls whether the USB voltage regulator is enabled.<br><br>0 USB voltage regulator is disabled.<br>1 USB voltage regulator is enabled. |
| 30<br>USBSSTBY | USB voltage regulator in standby mode during Stop, VLPS, LLS and VLLS modes.<br><br>Controls whether the USB voltage regulator is placed in standby mode during Stop, VLPS, LLS and VLLS modes. |

*Table continues on the next page...*

## SIM_SOPT1 field descriptions (continued)

| Field | Description |
|---|---|
| | 0 USB voltage regulator not in standby during Stop, VLPS, LLS and VLLS modes.<br>1 USB voltage regulator in standby during Stop, VLPS, LLS and VLLS modes. |
| 29<br>USBVSTBY | USB voltage regulator in standby mode during VLPR and VLPW modes<br><br>Controls whether the USB voltage regulator is placed in standby mode during VLPR and VLPW modes.<br><br>0 USB voltage regulator not in standby during VLPR and VLPW modes.<br>1 USB voltage regulator in standby during VLPR and VLPW modes. |
| 28–20<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 19–18<br>OSC32KSEL | 32K oscillator clock select<br><br>Selects the 32 kHz clock source (ERCLK32K) for LPTMR. This field is reset only on POR/LVD.<br><br>00 System oscillator (OSC32KCLK)<br>01 Reserved<br>10 RTC 32.768kHz oscillator<br>11 LPO 1 kHz |
| 17–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–12<br>RAMSIZE | RAM size<br><br>This field specifies the amount of system RAM available on the device.<br><br>0001 8 KB<br>0011 16 KB<br>0100 24 KB<br>0101 32 KB<br>0110 48 KB<br>0111 64 KB<br>1000 96 KB<br>1001 128 KB |
| 11–6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5–0<br>Reserved | This field is reserved. |

## 12.2.2 SOPT1 Configuration Register (SIM_SOPT1CFG)

**NOTE**

The SOPT1CFG register is reset on System Reset not VLLS.

Address: 4004_7000h base + 4h offset = 4004_7004h

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

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

### SIM_SOPT1CFG field descriptions

| Field | Description |
|---|---|
| 31–27<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 26<br>USSWE | USB voltage regulator stop standby write enable<br><br>Writing one to the USSWE bit allows the SOPT1 USBSSTBY bit to be written. This register bit clears after a write to USBSSTBY.<br><br>0 SOPT1 USBSSTBY cannot be written.<br>1 SOPT1 USBSSTBY can be written. |
| 25<br>UVSWE | USB voltage regulator VLP standby write enable<br><br>Writing one to the UVSWE bit allows the SOPT1 USBVSTBY bit to be written. This register bit clears after a write to USBVSTBY.<br><br>0 SOPT1 USBVSTBY cannot be written.<br>1 SOPT1 USBVSTBY can be written. |
| 24<br>URWE | USB voltage regulator enable write enable<br><br>Writing one to the URWE bit allows the SOPT1 USBREGEN bit to be written. This register bit clears after a write to USBREGEN.<br><br>0 SOPT1 USBREGEN cannot be written.<br>1 SOPT1 USBREGEN can be written. |

*Table continues on the next page...*

**SIM_SOPT1CFG field descriptions (continued)**

| Field | Description |
|---|---|
| 23–10<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 9–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 12.2.3 System Options Register 2 (SIM_SOPT2)

SOPT2 contains the controls for selecting many of the module clock source options on this device. See the Clock Distribution chapter for more information including clocking diagrams and definitions of device clocks.

Address: 4004_7000h base + 1004h offset = 4004_8004h

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

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | TRACECLKSEL | PTD7PAD | 0 | FBSL | | CLKOUTSEL | | | RTCCLKOUTSEL | 0 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SIM_SOPT2 field descriptions**

| Field | Description |
|---|---|
| 31–30<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 29–28<br>SDHCSRC | SDHC clock source select<br><br>Selects the clock source for the SDHC clock .<br><br>00 Core/system clock.<br>01 MCGPLLCLK/MCGFLLCLK clock<br>10 OSCERCLK clock<br>11 External bypass clock (SDHC0_CLKIN) |

*Table continues on the next page...*

## SIM_SOPT2 field descriptions (continued)

| Field | Description |
|---|---|
| 27–26<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 25–24<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 23–22<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 21–19<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 18<br>USBSRC | USB clock source select<br><br>Selects the clock source for the USB 48 MHz clock.<br><br>0 External bypass clock (USB_CLKIN).<br>1 MCGPLLCLK/MCGFLLCLK clock divided by the USB fractional divider. See the SIM_CLKDIV2[USBFRAC, USBDIV] descriptions. |
| 17<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 16<br>PLLFLLSEL | PLL/FLL clock select<br><br>Selects the MCGPLLCLK or MCGFLLCLK clock for various peripheral clocking options.<br><br>0 MCGFLLCLK clock<br>1 MCGPLLCLK clock |
| 15–13<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 12<br>TRACECLKSEL | Debug trace clock select<br><br>Selects the core/system clock or MCG output clock (MCGOUTCLK) as the trace clock source.<br><br>0 MCGOUTCLK<br>1 Core/system clock |
| 11<br>PTD7PAD | PTD7 pad drive strength<br><br>Controls the output drive strength of the PTD7 pin by selecting either one or two pads to drive it.<br><br>0 Single-pad drive strength for PTD7.<br>1 Double pad drive strength for PTD7. |
| 10<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 9–8<br>FBSL | FlexBus security level<br><br>If flash security is enabled, then this field affects what CPU operations can access off-chip via the FlexBus interface. This field has no effect if flash security is not enabled.<br><br>00 All off-chip accesses (instruction and data) via the FlexBus are disallowed.<br>01 All off-chip accesses (instruction and data) via the FlexBus are disallowed.<br>10 Off-chip instruction accesses are disallowed. Data accesses are allowed.<br>11 Off-chip instruction accesses and data accesses are allowed. |
| 7–5<br>CLKOUTSEL | CLKOUT select |

*Table continues on the next page...*

**SIM_SOPT2 field descriptions (continued)**

| Field | Description |
|---|---|
| | Selects the clock to output on the CLKOUT pin.<br><br>000 FlexBus CLKOUT<br>001 Reserved<br>010 Flash clock<br>011 LPO clock (1 kHz)<br>100 MCGIRCLK<br>101 RTC 32.768kHz clock<br>110 OSCERCLK0<br>111 Reserved |
| 4<br>RTCCLKOUTSEL | RTC clock out select<br><br>Selects either the RTC 1 Hz clock or the 32.768kHz clock to be output on the RTC_CLKOUT pin.<br><br>0 RTC 1 Hz clock is output on the RTC_CLKOUT pin.<br>1 RTC 32.768kHz clock is output on the RTC_CLKOUT pin. |
| 3–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 12.2.4 System Options Register 4 (SIM_SOPT4)

Address: 4004_7000h base + 100Ch offset = 4004_800Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FTM3TRG1SRC | FTM3TRG0SRC | FTM0TRG1SRC | FTM0TRG0SRC | FTM3CLKSEL | FTM2CLKSEL | FTM1CLKSEL | FTM0CLKSEL | 0 | 0 | FTM2CH0SRC | | FTM1CH0SRC | | 0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | FTM3FLT0 | 0 | | | FTM2FLT0 | 0 | | | FTM1FLT0 | 0 | FTM0FLT2 | FTM0FLT1 | FTM0FLT0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SIM_SOPT4 field descriptions**

| Field | Description |
|---|---|
| 31<br>FTM3TRG1SRC | FlexTimer 3 Hardware Trigger 1 Source Select<br><br>Selects the source of FTM3 hardware trigger 1. |

*Table continues on the next page...*

## SIM_SOPT4 field descriptions (continued)

| Field | Description |
|---|---|
| | 0 Reserved<br>1 FTM2 channel match drives FTM3 hardware trigger 1 |
| 30<br>FTM3TRG0SRC | FlexTimer 3 Hardware Trigger 0 Source Select<br><br>Selects the source of FTM3 hardware trigger 0.<br><br>0 Reserved<br>1 FTM1 channel match drives FTM3 hardware trigger 0 |
| 29<br>FTM0TRG1SRC | FlexTimer 0 Hardware Trigger 1 Source Select<br><br>Selects the source of FTM0 hardware trigger 1.<br><br>0 PDB output trigger 1 drives FTM0 hardware trigger 1<br>1 FTM2 channel match drives FTM0 hardware trigger 1 |
| 28<br>FTM0TRG0SRC | FlexTimer 0 Hardware Trigger 0 Source Select<br><br>Selects the source of FTM0 hardware trigger 0.<br><br>0 HSCMP0 output drives FTM0 hardware trigger 0<br>1 FTM1 channel match drives FTM0 hardware trigger 0 |
| 27<br>FTM3CLKSEL | FlexTimer 3 External Clock Pin Select<br><br>Selects the external pin used to drive the clock to the FTM3 module.<br><br>**NOTE:** The selected pin must also be configured for the FTM3 module external clock function through the appropriate pin control register in the port control module.<br><br>0 FTM3 external clock driven by FTM_CLK0 pin.<br>1 FTM3 external clock driven by FTM_CLK1 pin. |
| 26<br>FTM2CLKSEL | FlexTimer 2 External Clock Pin Select<br><br>Selects the external pin used to drive the clock to the FTM2 module.<br><br>**NOTE:** The selected pin must also be configured for the FTM2 module external clock function through the appropriate pin control register in the port control module.<br><br>0 FTM2 external clock driven by FTM_CLK0 pin.<br>1 FTM2 external clock driven by FTM_CLK1 pin. |
| 25<br>FTM1CLKSEL | FTM1 External Clock Pin Select<br><br>Selects the external pin used to drive the clock to the FTM1 module.<br><br>**NOTE:** The selected pin must also be configured for the FTM external clock function through the appropriate pin control register in the port control module.<br><br>0 FTM_CLK0 pin<br>1 FTM_CLK1 pin |
| 24<br>FTM0CLKSEL | FlexTimer 0 External Clock Pin Select<br><br>Selects the external pin used to drive the clock to the FTM0 module.<br><br>**NOTE:** The selected pin must also be configured for the FTM external clock function through the appropriate pin control register in the port control module. |

*Table continues on the next page...*

## SIM_SOPT4 field descriptions (continued)

| Field | Description |
|---|---|
| | 0 FTM_CLK0 pin<br>1 FTM_CLK1 pin |
| 23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 21–20<br>FTM2CH0SRC | FTM2 channel 0 input capture source select<br><br>Selects the source for FTM2 channel 0 input capture.<br><br>**NOTE:** When the FTM is not in input capture mode, clear this field.<br><br>00 FTM2_CH0 signal<br>01 CMP0 output<br>10 CMP1 output<br>11 Reserved |
| 19–18<br>FTM1CH0SRC | FTM1 channel 0 input capture source select<br><br>Selects the source for FTM1 channel 0 input capture.<br><br>**NOTE:** When the FTM is not in input capture mode, clear this field.<br><br>00 FTM1_CH0 signal<br>01 CMP0 output<br>10 CMP1 output<br>11 USB start of frame pulse |
| 17–13<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 12<br>FTM3FLT0 | FTM3 Fault 0 Select<br><br>Selects the source of FTM3 fault 0.<br><br>**NOTE:** The pin source for fault 0 must be configured for the FTM module fault function through the appropriate PORTx pin control register.<br><br>0 FTM3_FLT0 pin<br>1 CMP0 out |
| 11–9<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 8<br>FTM2FLT0 | FTM2 Fault 0 Select<br><br>Selects the source of FTM2 fault 0.<br><br>**NOTE:** The pin source for fault 0 must be configured for the FTM module fault function through the appropriate PORTx pin control register.<br><br>0 FTM2_FLT0 pin<br>1 CMP0 out |
| 7–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>FTM1FLT0 | FTM1 Fault 0 Select |

*Table continues on the next page...*

**SIM_SOPT4 field descriptions (continued)**

| Field | Description |
|---|---|
| | Selects the source of FTM1 fault 0.<br><br>**NOTE:** The pin source for fault 0 must be configured for the FTM module fault function through the appropriate pin control register in the port control module.<br><br>0 FTM1_FLT0 pin<br>1 CMP0 out |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>FTM0FLT2 | FTM0 Fault 2 Select<br><br>Selects the source of FTM0 fault 2.<br><br>**NOTE:** The pin source for fault 2 must be configured for the FTM module fault function through the appropriate pin control register in the port control module.<br><br>0 FTM0_FLT2 pin<br>1 CMP2 out |
| 1<br>FTM0FLT1 | FTM0 Fault 1 Select<br><br>Selects the source of FTM0 fault 1.<br><br>**NOTE:** The pin source for fault 1 must be configured for the FTM module fault function through the appropriate pin control register in the port control module.<br><br>0 FTM0_FLT1 pin<br>1 CMP1 out |
| 0<br>FTM0FLT0 | FTM0 Fault 0 Select<br><br>Selects the source of FTM0 fault 0.<br><br>**NOTE:** The pin source for fault 0 must be configured for the FTM module fault function through the appropriate pin control register in the port control module.<br><br>0 FTM0_FLT0 pin<br>1 CMP0 out |

## 12.2.5 System Options Register 5 (SIM_SOPT5)

Address: 4004_7000h base + 1010h offset = 4004_8010h

| Bit | 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 |

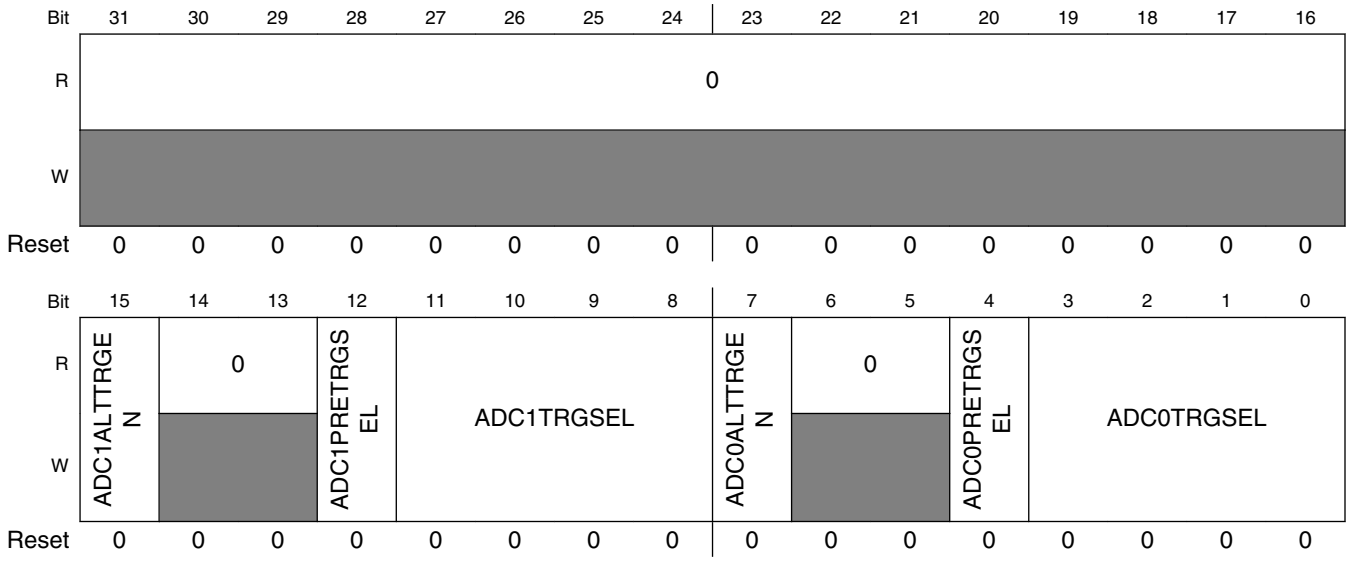
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | UART1RXSRC | | UART1TXSRC | | UART0RXSRC | | UART0TXSRC | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## SIM_SOPT5 field descriptions

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7–6<br>UART1RXSRC | UART 1 receive data source select<br><br>Selects the source for the UART 1 receive data.<br><br>00 UART1_RX pin<br>01 CMP0<br>10 CMP1<br>11 Reserved |
| 5–4<br>UART1TXSRC | UART 1 transmit data source select<br><br>Selects the source for the UART 1 transmit data.<br><br>00 UART1_TX pin<br>01 UART1_TX pin modulated with FTM1 channel 0 output<br>10 UART1_TX pin modulated with FTM2 channel 0 output<br>11 Reserved |
| 3–2<br>UART0RXSRC | UART 0 receive data source select<br><br>Selects the source for the UART 0 receive data.<br><br>00 UART0_RX pin<br>01 CMP0<br>10 CMP1<br>11 Reserved |
| 1–0<br>UART0TXSRC | UART 0 transmit data source select<br><br>Selects the source for the UART 0 transmit data.<br><br>00 UART0_TX pin<br>01 UART0_TX pin modulated with FTM1 channel 0 output<br>10 UART0_TX pin modulated with FTM2 channel 0 output<br>11 Reserved |

## 12.2.6 System Options Register 7 (SIM_SOPT7)

Address: 4004_7000h base + 1018h offset = 4004_8018h




### SIM_SOPT7 field descriptions

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15<br>ADC1ALTTRGEN | ADC1 alternate trigger enable<br><br>Enable alternative conversion triggers for ADC1.<br><br>0 PDB trigger selected for ADC1<br>1 Alternate trigger selected for ADC1 as defined by ADC1TRGSEL. |
| 14–13<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 12<br>ADC1PRETRGSEL | ADC1 pre-trigger select<br><br>Selects the ADC1 pre-trigger source when alternative triggers are enabled through ADC1ALTTRGEN.<br><br>0 Pre-trigger A selected for ADC1.<br>1 Pre-trigger B selected for ADC1. |
| 11–8<br>ADC1TRGSEL | ADC1 trigger select<br><br>Selects the ADC1 trigger source when alternative triggers are functional in stop and VLPS modes.<br><br>0000 PDB external trigger pin input (PDB0_EXTRG)<br>0001 High speed comparator 0 output<br>0010 High speed comparator 1 output<br>0011 High speed comparator 2 output<br>0100 PIT trigger 0<br>0101 PIT trigger 1 |

*Table continues on the next page...*

## SIM_SOPT7 field descriptions (continued)

| Field | Description |
|---|---|
| | 0110 PIT trigger 2<br>0111 PIT trigger 3<br>1000 FTM0 trigger<br>1001 FTM1 trigger<br>1010 FTM2 trigger<br>1011 FTM3 trigger<br>1100 RTC alarm<br>1101 RTC seconds<br>1110 Low-power timer trigger<br>1111 Reserved |
| 7<br>ADC0ALTTRGEN | ADC0 alternate trigger enable<br><br>Enable alternative conversion triggers for ADC0.<br><br>0 PDB trigger selected for ADC0.<br>1 Alternate trigger selected for ADC0. |
| 6–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>ADC0PRETRGSEL | ADC0 pretrigger select<br><br>Selects the ADC0 pre-trigger source when alternative triggers are enabled through ADC0ALTTRGEN.<br><br>0 Pre-trigger A<br>1 Pre-trigger B |
| 3–0<br>ADC0TRGSEL | ADC0 trigger select<br><br>Selects the ADC0 trigger source when alternative triggers are functional in stop and VLPS modes. .<br><br>0000 PDB external trigger pin input (PDB0_EXTRG)<br>0001 High speed comparator 0 output<br>0010 High speed comparator 1 output<br>0011 High speed comparator 2 output<br>0100 PIT trigger 0<br>0101 PIT trigger 1<br>0110 PIT trigger 2<br>0111 PIT trigger 3<br>1000 FTM0 trigger<br>1001 FTM1 trigger<br>1010 FTM2 trigger<br>1011 FTM3 trigger<br>1100 RTC alarm<br>1101 RTC seconds<br>1110 Low-power timer trigger<br>1111 Reserved |

## 12.2.7 System Device Identification Register (SIM_SDID)

Address: 4004_7000h base + 1024h offset = 4004_8024h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | REVID | | | | DIEID | | | | | FAMID | | | PINID | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x* | x* | x* | x* | 0 | 0 | 1 | 1 | 0 | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**SIM_SDID field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–12<br>REVID | Device revision number<br><br>Specifies the silicon implementation number for the device. |
| 11–7<br>DIEID | Device die number<br><br>Specifies the silicon implementation number for the device. |
| 6–4<br>FAMID | Kinetis family identification<br><br>Specifies the Kinetis family of the device.<br><br>000 K10 or K12 Family<br>001 K20 or K22 Family<br>010 K30 Family or K11 Family or K61 Family<br>011 K40 Family or K21 Family<br>100 K60 or K62 Family<br>101 K70 Family<br>110 Reserved<br>111 Reserved |
| 3–0<br>PINID | Pincount identification<br><br>Specifies the pincount of the device.<br><br>0000 Reserved<br>0001 Reserved<br>0010 32-pin<br>0011 Reserved<br>0100 48-pin<br>0101 64-pin<br>0110 80-pin<br>0111 81-pin or 121-pin<br>1000 100-pin<br>1001 121-pin<br>1010 144-pin |

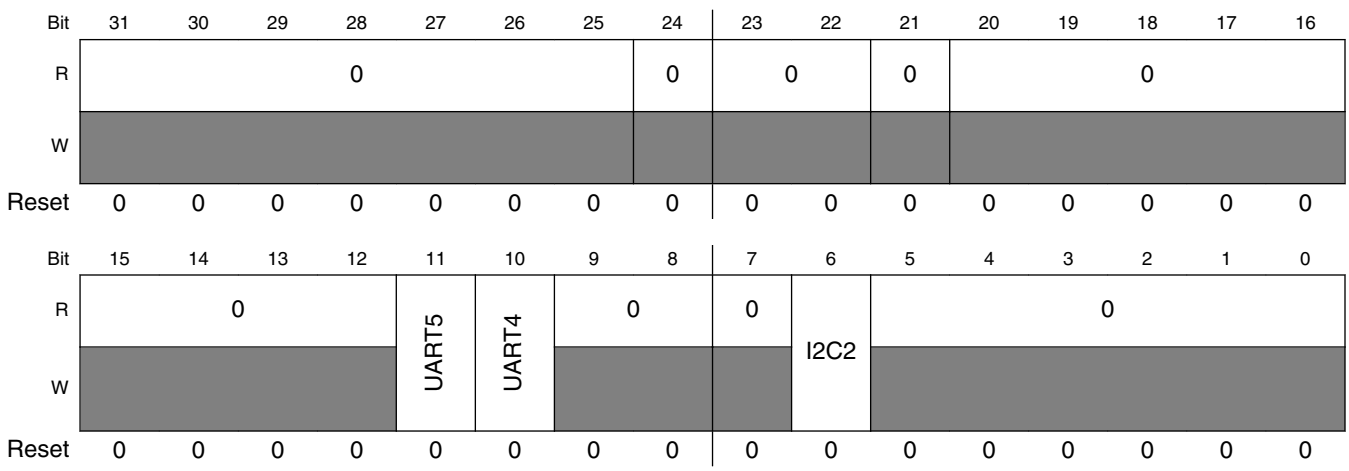
*Table continues on the next page...*

**SIM_SDID field descriptions (continued)**

| Field | Description |
|---|---|
| | 1011 Custom pinout (WLCSP)<br>1100 Reserved<br>1101 Reserved<br>1110 256-pin<br>1111 Reserved |

## 12.2.8 System Clock Gating Control Register 1 (SIM_SCGC1)

Address: 4004_7000h base + 1028h offset = 4004_8028h




**SIM_SCGC1 field descriptions**

| Field | Description |
|---|---|
| 31–25<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 24<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 23–22<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 21<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 20–12<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 11<br>UART5 | UART5 Clock Gate Control<br><br>This bit controls the clock gate to the UART5 module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 10<br>UART4 | UART4 Clock Gate Control |

*Table continues on the next page...*

**SIM_SCGC1 field descriptions (continued)**

| Field | Description |
|---|---|
| | This bit controls the clock gate to the UART4 module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 9–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6<br>I2C2 | I2C2 Clock Gate Control<br><br>This bit controls the clock gate to the I2C2 module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 5–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 12.2.9 System Clock Gating Control Register 2 (SIM_SCGC2)

DAC0 can be accessed through both AIPS0 and AIPS1. When accessing through AIPS1, define the clock gate control bits in the SCGC2. When accessing through AIPS0, define the clock gate control bits in SCGC6. See the Chip Configuration chapter for the base addresses of RNGA, FTM2, and DAC0 accessed via AIPS0 and AIPS1

Address: 4004_7000h base + 102Ch offset = 4004_802Ch

| Bit | 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | 0 | | | | 0 | | | | 0 | | | 0 |
| W | | | DAC1 | DAC0 | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

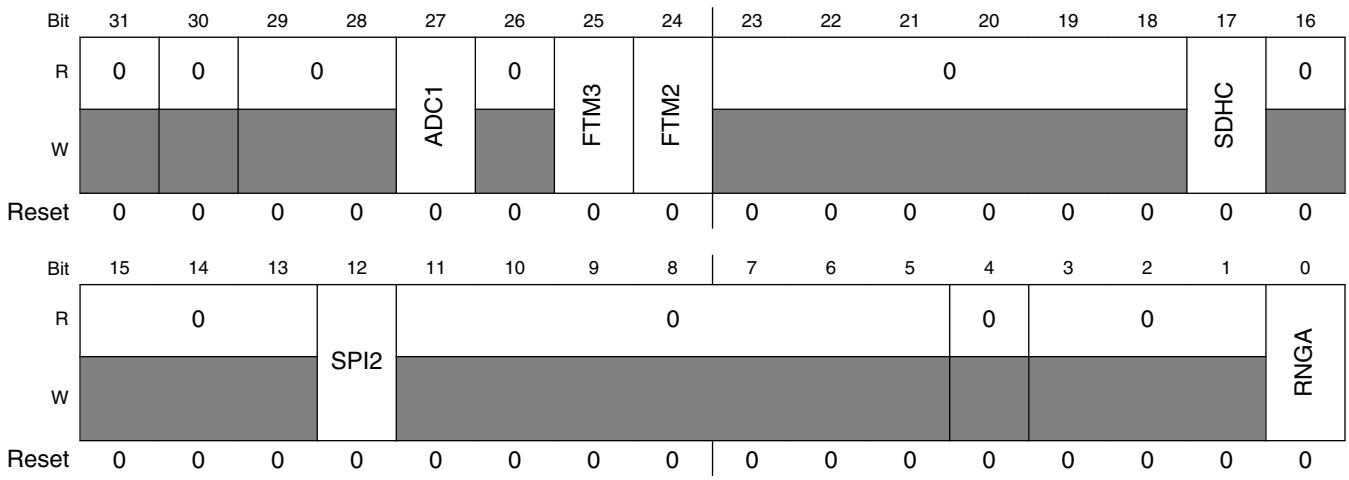
## SIM_SCGC2 field descriptions

| Field | Description |
|---|---|
| 31–14<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 13<br>DAC1 | DAC1 Clock Gate Control<br><br>This bit controls the clock gate to the DAC1 module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 12<br>DAC0 | DAC0 Clock Gate Control<br><br>This bit controls the clock gate to the DAC0 module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 11–9<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 8–4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 12.2.10 System Clock Gating Control Register 3 (SIM_SCGC3)

FTM2 and RNGA can be accessed through both AIPS0 and AIPS1. When accessing through AIPS1, define the clock gate control bits in the SCGC3. When accessing through AIPS0, define the clock gate control bits in SCGC6. See the Chip Configuration chapter for the base addresses of RNGA, FTM2, and DAC0 accessed via AIPS0 and AIPS1.

Address: 4004_7000h base + 1030h offset = 4004_8030h




**SIM_SCGC3 field descriptions**

| Field | Description |
|---|---|
| 31<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 30<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 29–28<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 27<br>ADC1 | ADC1 Clock Gate Control<br><br>This bit controls the clock gate to the ADC1 module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 26<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 25<br>FTM3 | FTM3 Clock Gate Control<br><br>This bit controls the clock gate to the FTM3 module.<br><br>0 Clock disabled<br>1 Clock enabled |

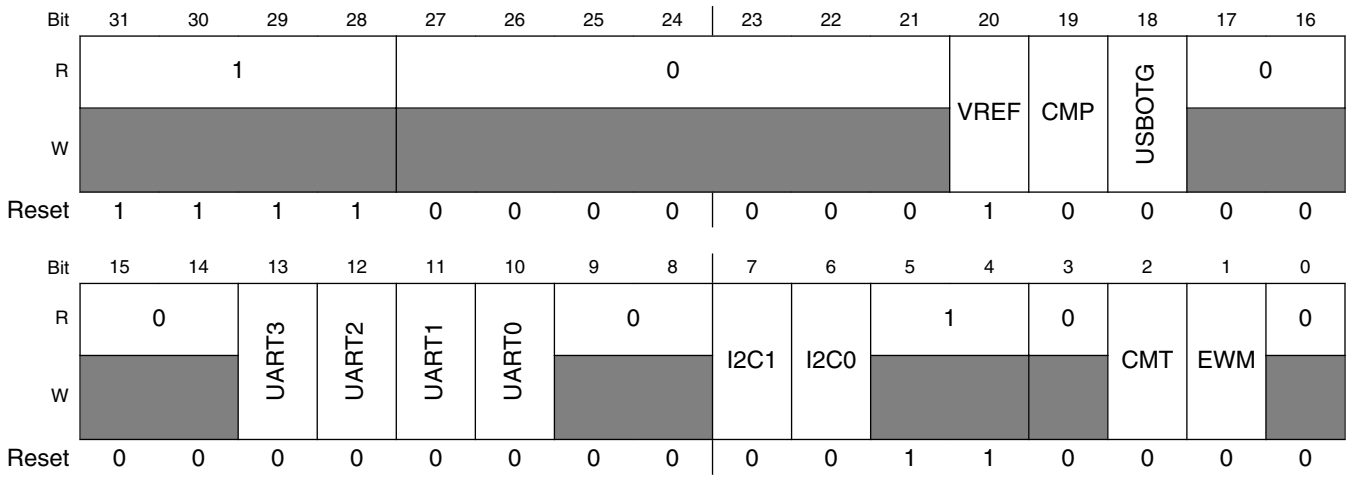
*Table continues on the next page...*

## SIM_SCGC3 field descriptions (continued)

| Field | Description |
|---|---|
| 24<br>FTM2 | FTM2 Clock Gate Control<br><br>This bit controls the clock gate to the FTM2 module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 23–18<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 17<br>SDHC | SDHC Clock Gate Control<br><br>This bit controls the clock gate to the SDHC module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 16–13<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 12<br>SPI2 | SPI2 Clock Gate Control<br><br>This bit controls the clock gate to the SPI2 module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 11–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>RNGA | RNGA Clock Gate Control<br><br>This bit controls the clock gate to the RNGA module.<br><br>0 Clock disabled<br>1 Clock enabled |

## 12.2.11 System Clock Gating Control Register 4 (SIM_SCGC4)

Address: 4004_7000h base + 1034h offset = 4004_8034h




### SIM_SCGC4 field descriptions

| Field | Description |
|---|---|
| 31–28<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 1. |
| 27–21<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 20<br>VREF | VREF Clock Gate Control<br><br>This bit controls the clock gate to the VREF module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 19<br>CMP | Comparator Clock Gate Control<br><br>This bit controls the clock gate to the comparator module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 18<br>USBOTG | USB Clock Gate Control<br><br>This bit controls the clock gate to the USB module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 17–14<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 13<br>UART3 | UART3 Clock Gate Control<br><br>This bit controls the clock gate to the UART3 module. |

*Table continues on the next page...*

## SIM_SCGC4 field descriptions (continued)

| Field | Description |
|---|---|
| | 0 Clock disabled<br>1 Clock enabled |
| 12<br>UART2 | UART2 Clock Gate Control<br><br>This bit controls the clock gate to the UART2 module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 11<br>UART1 | UART1 Clock Gate Control<br><br>This bit controls the clock gate to the UART1 module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 10<br>UART0 | UART0 Clock Gate Control<br><br>This bit controls the clock gate to the UART0 module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 9–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>I2C1 | I2C1 Clock Gate Control<br><br>This bit controls the clock gate to the $I^2C1$ module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 6<br>I2C0 | I2C0 Clock Gate Control<br><br>This bit controls the clock gate to the $I^2C0$ module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 5–4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 1. |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>CMT | CMT Clock Gate Control<br><br>This bit controls the clock gate to the CMT module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 1<br>EWM | EWM Clock Gate Control<br><br>This bit controls the clock gate to the EWM module.<br><br>0 Clock disabled<br>1 Clock enabled |

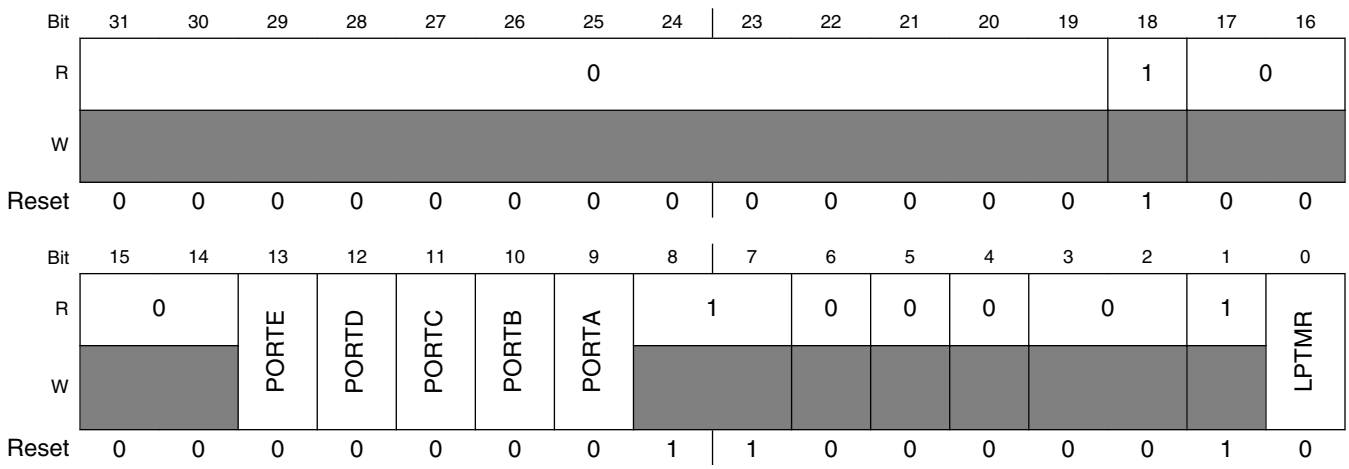
*Table continues on the next page...*

**SIM_SCGC4 field descriptions (continued)**

| Field | Description |
|---|---|
| 0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 12.2.12 System Clock Gating Control Register 5 (SIM_SCGC5)

Address: 4004_7000h base + 1038h offset = 4004_8038h




**SIM_SCGC5 field descriptions**

| Field | Description |
|---|---|
| 31–19<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 18<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 1. |
| 17–14<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 13<br>PORTE | Port E Clock Gate Control<br><br>This bit controls the clock gate to the Port E module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 12<br>PORTD | Port D Clock Gate Control<br><br>This bit controls the clock gate to the Port D module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 11<br>PORTC | Port C Clock Gate Control<br><br>This bit controls the clock gate to the Port C module. |

*Table continues on the next page...*

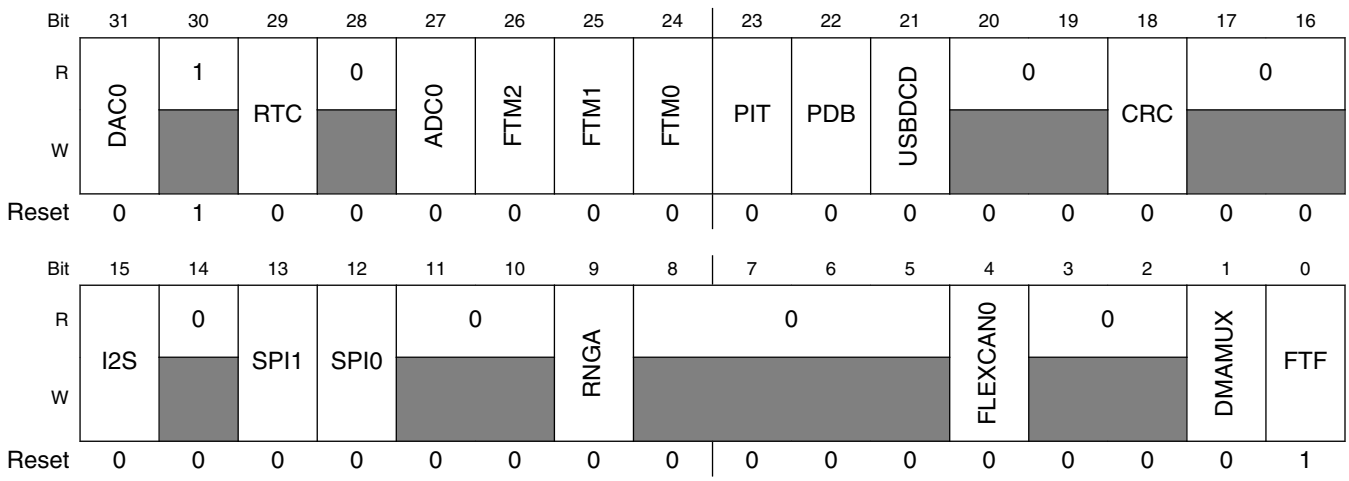
## SIM_SCGC5 field descriptions (continued)

| Field | Description |
|---|---|
| | 0 Clock disabled<br>1 Clock enabled |
| 10<br>PORTB | Port B Clock Gate Control<br><br>This bit controls the clock gate to the Port B module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 9<br>PORTA | Port A Clock Gate Control<br><br>This bit controls the clock gate to the Port A module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 8–7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 1. |
| 6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3–2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 1. |
| 0<br>LPTMR | Low Power Timer Access Control<br><br>This bit controls software access to the Low Power Timer module.<br><br>0 Access disabled<br>1 Access enabled |

## 12.2.13 System Clock Gating Control Register 6 (SIM_SCGC6)

DAC0, FTM2, and RNGA can be accessed through both AIPS0 and AIPS1. When accessing through AIPS1, define the clock gate control bits in the SCGC2 and SCGC3. When accessing through AIPS0, define the clock gate control bits in SCGC6. See the Chip Configuration chapter for the base addresses of RNGA, FTM2, and DAC0 accessed via AIPS0 and AIPS1.

Address: 4004_7000h base + 103Ch offset = 4004_803Ch




**SIM_SCGC6 field descriptions**

| Field | Description |
|---|---|
| 31<br>DAC0 | DAC0 Clock Gate Control<br><br>This bit controls the clock gate to the DAC0 module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 30<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 1. |
| 29<br>RTC | RTC Access Control<br><br>This bit controls software access and interrupts to the RTC module.<br><br>0 Access and interrupts disabled<br>1 Access and interrupts enabled |
| 28<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 27<br>ADC0 | ADC0 Clock Gate Control<br><br>This bit controls the clock gate to the ADC0 module.<br><br>0 Clock disabled<br>1 Clock enabled |

*Table continues on the next page...*

## SIM_SCGC6 field descriptions (continued)

| Field | Description |
|---|---|
| 26<br>FTM2 | FTM2 Clock Gate Control<br><br>This bit controls the clock gate to the FTM2 module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 25<br>FTM1 | FTM1 Clock Gate Control<br><br>This bit controls the clock gate to the FTM1 module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 24<br>FTM0 | FTM0 Clock Gate Control<br><br>This bit controls the clock gate to the FTM0 module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 23<br>PIT | PIT Clock Gate Control<br><br>This bit controls the clock gate to the PIT module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 22<br>PDB | PDB Clock Gate Control<br><br>This bit controls the clock gate to the PDB module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 21<br>USBDCD | USB DCD Clock Gate Control<br><br>This bit controls the clock gate to the USB DCD module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 20–19<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 18<br>CRC | CRC Clock Gate Control<br><br>This bit controls the clock gate to the CRC module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 17–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15<br>I2S | I2S Clock Gate Control<br><br>This bit controls the clock gate to the $I^2S$ module. |

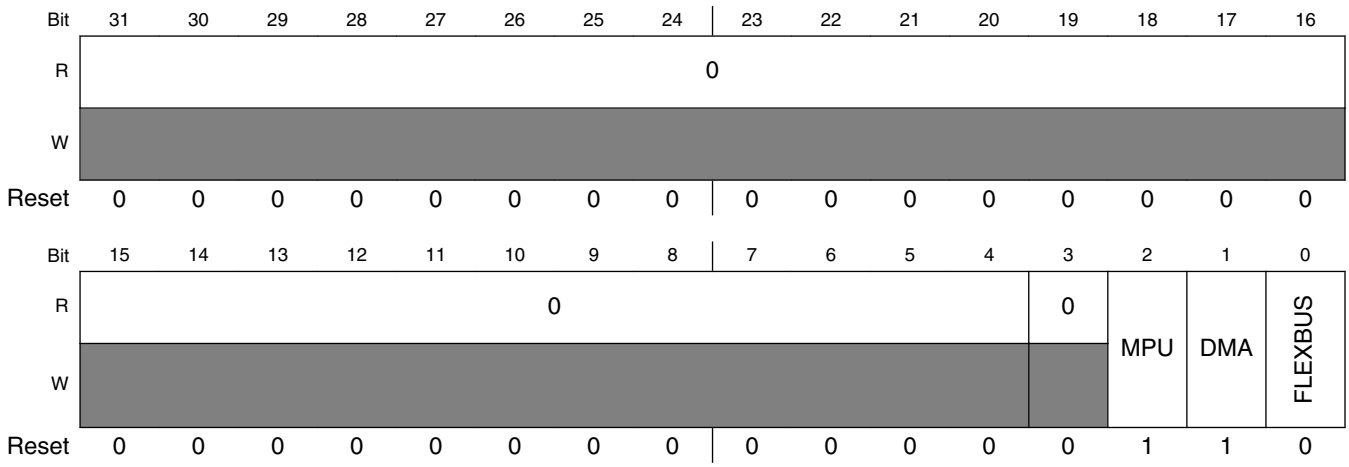
*Table continues on the next page...*

## SIM_SCGC6 field descriptions (continued)

| Field | Description |
|---|---|
| | 0 Clock disabled<br>1 Clock enabled |
| 14<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 13<br>SPI1 | SPI1 Clock Gate Control<br><br>This bit controls the clock gate to the SPI1 module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 12<br>SPI0 | SPI0 Clock Gate Control<br><br>This bit controls the clock gate to the SPI0 module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 11–10<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 9<br>RNGA | RNGA Clock Gate Control<br><br>This bit controls the clock gate to the RNGA module. |
| 8–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>FLEXCAN0 | FlexCAN0 Clock Gate Control<br><br>This bit controls the clock gate to the FlexCAN0 module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 3–2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 1<br>DMAMUX | DMA Mux Clock Gate Control<br><br>This bit controls the clock gate to the DMA Mux module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 0<br>FTF | Flash Memory Clock Gate Control<br><br>This bit controls the clock gate to the flash memory. Flash reads are still supported while the flash memory is clock gated, but entry into low power modes is blocked.<br><br>0 Clock disabled<br>1 Clock enabled |

## 12.2.14 System Clock Gating Control Register 7 (SIM_SCGC7)

Address: 4004_7000h base + 1040h offset = 4004_8040h




**SIM_SCGC7 field descriptions**

| Field | Description |
|---|---|
| 31–4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>MPU | MPU Clock Gate Control<br><br>This bit controls the clock gate to the MPU module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 1<br>DMA | DMA Clock Gate Control<br><br>This bit controls the clock gate to the DMA module.<br><br>0 Clock disabled<br>1 Clock enabled |
| 0<br>FLEXBUS | FlexBus Clock Gate Control<br><br>This bit controls the clock gate to the FlexBus module.<br><br>0 Clock disabled<br>1 Clock enabled |

## 12.2.15 System Clock Divider Register 1 (SIM_CLKDIV1)

When updating CLKDIV1, update all fields using the same write command.

## NOTE

The CLKDIV1 register cannot be written to when the device is in VLPR mode.

Address: 4004_7000h base + 1044h offset = 4004_8044h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OUTDIV1 | | | | OUTDIV2 | | | | OUTDIV3 | | | | OUTDIV4 | | | | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 1* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* |

* Notes:
- Reset value loaded during Syetem Reset from FTF_FOPT[LPBOOT].

### SIM_CLKDIV1 field descriptions

| Field | Description |
|---|---|
| 31–28<br>OUTDIV1 | Clock 1 output divider value<br><br>This field sets the divide value for the core/system clock from MCGOUTCLK. At the end of reset, it is loaded with either 0000 or 0111 depending on FTF_FOPT[LPBOOT].<br><br>0000 Divide-by-1.<br>0001 Divide-by-2.<br>0010 Divide-by-3.<br>0011 Divide-by-4.<br>0100 Divide-by-5.<br>0101 Divide-by-6.<br>0110 Divide-by-7.<br>0111 Divide-by-8.<br>1000 Divide-by-9.<br>1001 Divide-by-10.<br>1010 Divide-by-11.<br>1011 Divide-by-12.<br>1100 Divide-by-13.<br>1101 Divide-by-14.<br>1110 Divide-by-15.<br>1111 Divide-by-16. |
| 27–24<br>OUTDIV2 | Clock 2 output divider value<br><br>This field sets the divide value for the bus clock from MCGOUTCLK. At the end of reset, it is loaded with either 0000 or 0111 depending on FTF_FOPT[LPBOOT]. The bus clock frequency must be an integer divide of the core/system clock frequency.<br><br>0000 Divide-by-1.<br>0001 Divide-by-2.<br>0010 Divide-by-3.<br>0011 Divide-by-4.<br>0100 Divide-by-5.<br>0101 Divide-by-6.<br>0110 Divide-by-7.<br>0111 Divide-by-8. |

*Table continues on the next page...*

## SIM_CLKDIV1 field descriptions (continued)

| Field | Description |
|---|---|
| | 1000 Divide-by-9.<br>1001 Divide-by-10.<br>1010 Divide-by-11.<br>1011 Divide-by-12.<br>1100 Divide-by-13.<br>1101 Divide-by-14.<br>1110 Divide-by-15.<br>1111 Divide-by-16. |
| 23–20<br>OUTDIV3 | Clock 3 output divider value<br><br>This field sets the divide value for the FlexBus clock (external pin FB_CLK) from MCGOUTCLK. At the end of reset, it is loaded with either 0001 or 1111 depending on FTF_FOPT[LPBOOT]. The FlexBus clock frequency must be an integer divide of the system clock frequency.<br><br>0000 Divide-by-1.<br>0001 Divide-by-2.<br>0010 Divide-by-3.<br>0011 Divide-by-4.<br>0100 Divide-by-5.<br>0101 Divide-by-6.<br>0110 Divide-by-7.<br>0111 Divide-by-8.<br>1000 Divide-by-9.<br>1001 Divide-by-10.<br>1010 Divide-by-11.<br>1011 Divide-by-12.<br>1100 Divide-by-13.<br>1101 Divide-by-14.<br>1110 Divide-by-15.<br>1111 Divide-by-16. |
| 19–16<br>OUTDIV4 | Clock 4 output divider value<br><br>This field sets the divide value for the flash clock from MCGOUTCLK. At the end of reset, it is loaded with either 0001 or 1111 depending on FTF_FOPT[LPBOOT]. The flash clock frequency must be an integer divide of the system clock frequency.<br><br>0000 Divide-by-1.<br>0001 Divide-by-2.<br>0010 Divide-by-3.<br>0011 Divide-by-4.<br>0100 Divide-by-5.<br>0101 Divide-by-6.<br>0110 Divide-by-7.<br>0111 Divide-by-8.<br>1000 Divide-by-9.<br>1001 Divide-by-10.<br>1010 Divide-by-11.<br>1011 Divide-by-12. |

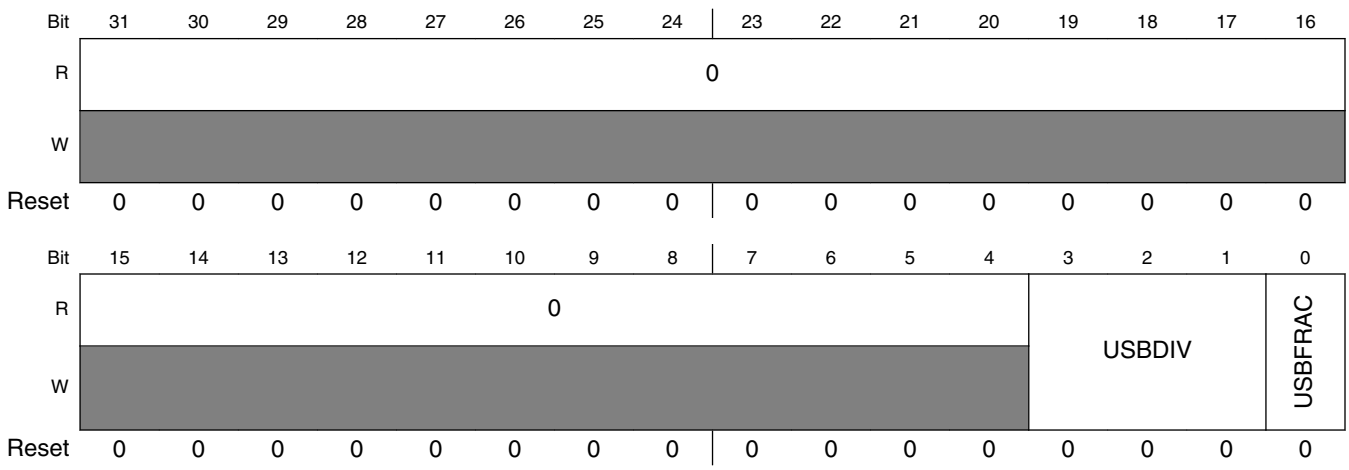
*Table continues on the next page...*

**SIM_CLKDIV1 field descriptions (continued)**

| Field | Description |
|---|---|
| | 1100 Divide-by-13.<br>1101 Divide-by-14.<br>1110 Divide-by-15.<br>1111 Divide-by-16. |
| 15–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 12.2.16 System Clock Divider Register 2 (SIM_CLKDIV2)

Address: 4004_7000h base + 1048h offset = 4004_8048h




**SIM_CLKDIV2 field descriptions**

| Field | Description |
|---|---|
| 31–4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3–1<br>USBDIV | USB clock divider divisor<br>This field sets the divide value for the fractional clock divider when the MCGFLLCLK/MCGPLLCLK clock is the USB clock source (SOPT2[USBSRC] = 1).<br>Divider output clock = Divider input clock × [ (USBFRAC+1) / (USBDIV+1) ] |
| 0<br>USBFRAC | USB clock divider fraction<br>This field sets the fraction multiply value for the fractional clock divider when the MCGFLLCLK/ MCGPLLCLK clock is the USB clock source (SOPT2[USBSRC] = 1).<br>Divider output clock = Divider input clock × [ (USBFRAC+1) / (USBDIV+1) ] |

## 12.2.17 Flash Configuration Register 1 (SIM_FCFG1)

For devices with FlexNVM: The reset value of EESIZE and DEPART are based on user programming in user IFR via the PGMPART flash command.

For devices with program flash only: The EESIZE and DEPART filelds are not applicable.

Address: 4004_7000h base + 104Ch offset = 4004_804Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | NVMSIZE | | | | PFSIZE | | | | 0 | | | | EESIZE | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 0* | 0* | 0* | 0* | 1* | 1* | 1* | 1* |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DEPART | | | | 0 | | | | | | FLASHDOZE | FLASHDIS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0* | 0* | 0* | 0* | 1* | 1* | 1* | 1* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* |

* Notes:
- Reset value loaded during System Reset from Flash IFR.

**SIM_FCFG1 field descriptions**

| Field | Description |
|---|---|
| 31–28<br>NVMSIZE | FlexNVM size<br><br>This field specifies the amount of FlexNVM memory available on the device . Undefined values are reserved.<br><br>0000 0 KB of FlexNVM<br>0011 32 KB of FlexNVM<br>0101 64 KB of FlexNVM |

*Table continues on the next page...*

## SIM_FCFG1 field descriptions (continued)

| Field | Description |
|---|---|
| | 0111 128 KB of FlexNVM<br>1001 256 KB of FlexNVM<br>1011 512 KB of FlexNVM<br>1111 512 KB of FlexNVM |
| 27–24<br>PFSIZE | Program flash size<br><br>This field specifies the amount of program flash memory available on the device . Undefined values are reserved.<br><br>0011 32 KB of program flash memory<br>0101 64 KB of program flash memory<br>0111 128 KB of program flash memory<br>1001 256 KB of program flash memory<br>1011 512 KB of program flash memory<br>1101 1024 KB of program flash memory<br>1111 1024 KB of program flash memory |
| 23–20<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 19–16<br>EESIZE | EEPROM size<br><br>EEPROM data size .<br><br>0000 16 KB<br>0001 8 KB<br>0010 4 KB<br>0011 2 KB<br>0100 1 KB<br>0101 512 Bytes<br>0110 256 Bytes<br>0111 128 Bytes<br>1000 64 Bytes<br>1001 32 Bytes<br>1010-1110 Reserved<br>1111 0 Bytes |
| 15–12<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 11–8<br>DEPART | FlexNVM partition<br><br>For devices with FlexNVM: Data flash / EEPROM backup split . See DEPART bit description in FTFE chapter.<br><br>For devices without FlexNVM: Reserved |
| 7–2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 1<br>FLASHDOZE | Flash Doze<br><br>When set, Flash memory is disabled for the duration of Wait mode. An attempt by the DMA or other bus master to access the Flash when the Flash is disabled will result in a bus error. This bit should be clear during VLP modes. The Flash will be automatically enabled again at the end of Wait mode so interrupt |

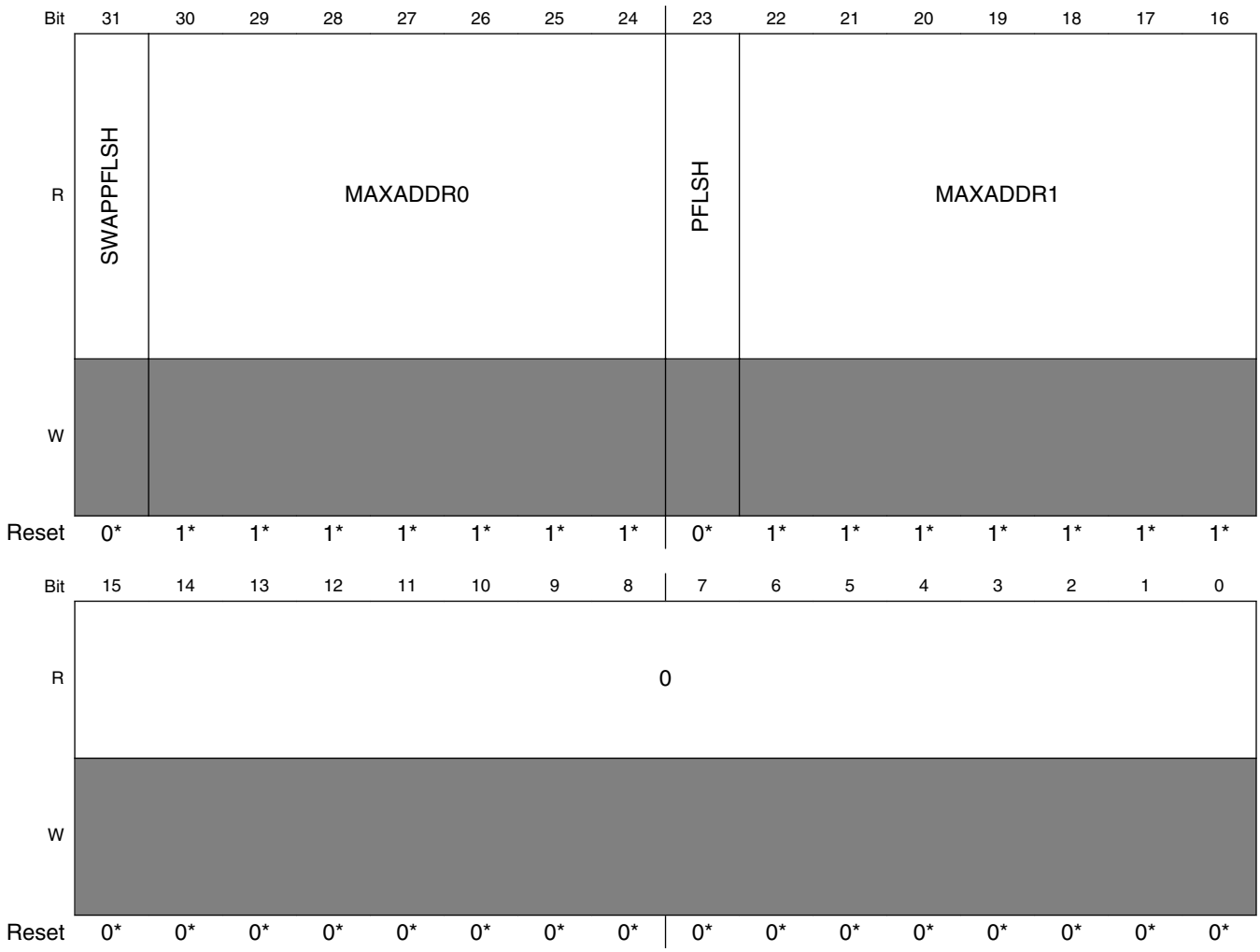
*Table continues on the next page...*

## SIM_FCFG1 field descriptions (continued)

| Field | Description |
|---|---|
| | vectors do not need to be relocated out of Flash memory. The wakeup time from Wait mode is extended when this bit is set.<br><br>0 Flash remains enabled during Wait mode<br>1 Flash is disabled for the duration of Wait mode |
| 0<br>FLASHDIS | Flash Disable<br><br>Flash accesses are disabled (and generate a bus error) and the Flash memory is placed in a low power state. This bit should not be changed during VLP modes. Relocate the interrupt vectors out of Flash memory before disabling the Flash.<br><br>0 Flash is enabled<br>1 Flash is disabled |

## 12.2.18 Flash Configuration Register 2 (SIM_FCFG2)

Address: 4004_7000h base + 1050h offset = 4004_8050h




* Notes:
- Reset value loaded during System Reset from Flash IFR.

### SIM_FCFG2 field descriptions

| Field | Description |
|---|---|
| 31<br>SWAPPFLSH | Swap program flash<br><br>For devices without FlexNVM: Indicates that swap is active .<br><br>0 Swap is not active.<br>1 Swap is active. |
| 30–24<br>MAXADDR0 | Max address block 0 |

*Table continues on the next page...*

**SIM_FCFG2 field descriptions (continued)**

| Field | Description |
|---|---|
| | This field concatenated with 13 trailing zeros indicates the first invalid address of flash block 0 (program flash 0).<br>For example, if MAXADDR0 = 0x20 the first invalid address of flash block 0 is 0x0004_0000. This would be the MAXADDR0 value for a device with 256 KB program flash in flash block 0. |
| 23<br>PFLSH | Program flash<br>For devices with FlexNVM, this bit is always clear.<br>For devices without FlexNVM, this bit is always set.<br>0 Physical flash block 1 is used as FlexNVM<br>Reserved for devices without FlexNVM<br>1 Physical flash block 1 is used as program flash |
| 22–16<br>MAXADDR1 | Max address block 1<br>For devices with FlexNVM: This field concatenated with 13 trailing zeros plus the FlexNVM base address indicates the first invalid address of the FlexNVM (flash block 1).<br>For example, if MAXADDR1 = 0x20 the first invalid address of flash block 1 is 0x4_0000 + 0x1000_0000 . This would be the MAXADDR1 value for a device with 256 KB FlexNVM.<br>For devices with program flash only: This field concatenated with 13 trailing zeros plus the value of the MAXADDR1 field indicates the first invalid address of the second program flash block (flash block 1).<br>For example, if MAXADDR0 = MAXADDR1 = 0x20 the first invalid address of flash block 1 is 0x4_0000 + 0x4_0000. This would be the MAXADDR1 value for a device with 512 KB program flash memory and no FlexNVM. |
| 15–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 12.2.19 Unique Identification Register High (SIM_UIDH)

Address: 4004_7000h base + 1054h offset = 4004_8054h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | UID | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* |

* Notes:
- Reset value loaded during System Reset from Flash IFR.

**SIM_UIDH field descriptions**

| Field | Description |
|---|---|
| 31–0<br>UID | Unique Identification<br>Unique identification for the device. |

## 12.2.20 Unique Identification Register Mid-High (SIM_UIDMH)

Address: 4004_7000h base + 1058h offset = 4004_8058h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | UID | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* |

* Notes:
- Reset value loaded during System Reset from Flash IFR.

### SIM_UIDMH field descriptions

| Field | Description |
|---|---|
| 31–0<br>UID | Unique Identification<br><br>Unique identification for the device. |

## 12.2.21 Unique Identification Register Mid Low (SIM_UIDML)

Address: 4004_7000h base + 105Ch offset = 4004_805Ch

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | UID | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* |

* Notes:
- Reset value loaded during System Reset from Flash IFR.

### SIM_UIDML field descriptions

| Field | Description |
|---|---|
| 31–0<br>UID | Unique Identification<br><br>Unique identification for the device. |

## 12.2.22 Unique Identification Register Low (SIM_UIDL)

Address: 4004_7000h base + 1060h offset = 4004_8060h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | UID | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* |

* Notes:
- Reset value loaded during System Reset from Flash IFR.

**SIM_UIDL field descriptions**

| Field | Description |
|---|---|
| 31–0<br>UID | Unique Identification<br><br>Unique identification for the device. |

# 12.3 Functional description

For more information about the functions of SIM, see the Introduction section.

# Chapter 13
# Reset Control Module (RCM)

## 13.1 Introduction

This chapter describes the registers of the Reset Control Module (RCM). The RCM implements many of the reset functions for the chip. See the chip's reset chapter for more information.

## 13.2 Reset memory map and register descriptions

The Reset Control Module (RCM) registers provide reset status information and reset filter control.

**NOTE**

The RCM registers can be written only in supervisor mode. Write accesses in user mode are blocked and will result in a bus error.

**RCM memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4007_F000 | System Reset Status Register 0 (RCM_SRS0) | 8 | R | 82h | 13.2.1/308 |
| 4007_F001 | System Reset Status Register 1 (RCM_SRS1) | 8 | R | 00h | 13.2.2/309 |
| 4007_F004 | Reset Pin Filter Control register (RCM_RPFC) | 8 | R/W | 00h | 13.2.3/311 |
| 4007_F005 | Reset Pin Filter Width register (RCM_RPFW) | 8 | R/W | 00h | 13.2.4/312 |
| 4007_F007 | Mode Register (RCM_MR) | 8 | R | 00h | 13.2.5/313 |

### 13.2.1 System Reset Status Register 0 (RCM_SRS0)

This register includes read-only status flags to indicate the source of the most recent reset. The reset state of these bits depends on what caused the MCU to reset.

**NOTE**

The reset value of this register depends on the reset source:

- POR (including LVD) — 0x82
- LVD (without POR) — 0x02
- VLLS mode wakeup due to $\overline{\text{RESET}}$ pin assertion — 0x41
- VLLS mode wakeup due to other wakeup sources — 0x01
- Other reset — a bit is set if its corresponding reset source caused the reset

Address: 4007_F000h base + 0h offset = 4007_F000h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | POR | PIN | WDOG | 0 | LOL | LOC | LVD | WAKEUP |
| Write | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

**RCM_SRS0 field descriptions**

| Field | Description |
|---|---|
| 7<br>POR | Power-On Reset<br><br>Indicates a reset has been caused by the power-on detection logic. Because the internal supply voltage was ramping up at the time, the low-voltage reset (LVD) status bit is also set to indicate that the reset occurred while the internal supply was below the LVD threshold.<br><br>0 Reset not caused by POR<br>1 Reset caused by POR |
| 6<br>PIN | External Reset Pin<br><br>Indicates a reset has been caused by an active-low level on the external $\overline{\text{RESET}}$ pin.<br><br>0 Reset not caused by external reset pin<br>1 Reset caused by external reset pin |
| 5<br>WDOG | Watchdog<br><br>Indicates a reset has been caused by the watchdog timer timing out.<br><br>0 Reset not caused by watchdog timeout<br>1 Reset caused by watchdog timeout |
| 4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3<br>LOL | Loss-of-Lock Reset |

*Table continues on the next page...*

**RCM_SRS0 field descriptions (continued)**

| Field | Description |
|---|---|
| | Indicates a reset has been caused by a loss of lock in the MCG PLL. See the MCG description for information on the loss-of-clock event.<br><br>0 Reset not caused by a loss of lock in the PLL<br>1 Reset caused by a loss of lock in the PLL |
| 2<br>LOC | Loss-of-Clock Reset<br><br>Indicates a reset has been caused by a loss of external clock. The MCG clock monitor must be enabled for a loss of clock to be detected. Refer to the detailed MCG description for information on enabling the clock monitor.<br><br>0 Reset not caused by a loss of external clock.<br>1 Reset caused by a loss of external clock. |
| 1<br>LVD | Low-Voltage Detect Reset<br><br>If the LVDRE bit is set and the supply drops below the LVD trip voltage, an LVD reset occurs. This bit is also set by POR.<br><br>0 Reset not caused by LVD trip or POR<br>1 Reset caused by LVD trip or POR |
| 0<br>WAKEUP | Low Leakage Wakeup Reset<br><br>Indicates a reset has been caused by an enabled LLWU module wakeup source while the chip was in a low leakage mode. In LLS mode, the $\overline{\text{RESET}}$ pin is the only wakeup source that can cause this reset. Any enabled wakeup source in a VLLSx mode causes a reset. This bit is cleared by any reset except WAKEUP.<br><br>0 Reset not caused by LLWU module wakeup source<br>1 Reset caused by LLWU module wakeup source |

## 13.2.2 System Reset Status Register 1 (RCM_SRS1)

This register includes read-only status flags to indicate the source of the most recent reset. The reset state of these bits depends on what caused the MCU to reset.

**NOTE**

The reset value of this register depends on the reset source:

- POR (including LVD) — 0x00
- LVD (without POR) — 0x00
- VLLS mode wakeup — 0x00
- Other reset — a bit is set if its corresponding reset source caused the reset

Address: 4007_F000h base + 1h offset = 4007_F001h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | TAMPER | 0 | SACKERR | EZPT | MDM_AP | SW | LOCKUP | JTAG |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## RCM_SRS1 field descriptions

| Field | Description |
|---|---|
| 7<br>TAMPER | Tamper detect<br><br>Indicates a reset was caused by tamper detect.<br><br>0 Reset not caused by tamper detect<br>1 Reset caused by tamper detect. |
| 6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5<br>SACKERR | Stop Mode Acknowledge Error Reset<br><br>Indicates that after an attempt to enter Stop mode, a reset has been caused by a failure of one or more peripherals to acknowledge within approximately one second to enter stop mode.<br><br>0 Reset not caused by peripheral failure to acknowledge attempt to enter stop mode<br>1 Reset caused by peripheral failure to acknowledge attempt to enter stop mode |
| 4<br>EZPT | EzPort Reset<br><br>Indicates a reset has been caused by EzPort receiving the RESET command while the device is in EzPort mode.<br><br>0 Reset not caused by EzPort receiving the RESET command while the device is in EzPort mode<br>1 Reset caused by EzPort receiving the RESET command while the device is in EzPort mode |
| 3<br>MDM_AP | MDM-AP System Reset Request<br><br>Indicates a reset has been caused by the host debugger system setting of the System Reset Request bit in the MDM-AP Control Register.<br><br>0 Reset not caused by host debugger system setting of the System Reset Request bit<br>1 Reset caused by host debugger system setting of the System Reset Request bit |
| 2<br>SW | Software<br><br>Indicates a reset has been caused by software setting of SYSRESETREQ bit in Application Interrupt and Reset Control Register in the ARM core.<br><br>0 Reset not caused by software setting of SYSRESETREQ bit<br>1 Reset caused by software setting of SYSRESETREQ bit |
| 1<br>LOCKUP | Core Lockup<br><br>Indicates a reset has been caused by the ARM core indication of a LOCKUP event.<br><br>0 Reset not caused by core LOCKUP event<br>1 Reset caused by core LOCKUP event |
| 0<br>JTAG | JTAG Generated Reset |

*Table continues on the next page...*

**RCM_SRS1 field descriptions (continued)**

| Field | Description |
|---|---|
| | Indicates a reset has been caused by JTAG selection of certain IR codes: EZPORT, EXTEST, HIGHZ, and CLAMP.<br><br>0 Reset not caused by JTAG<br>1 Reset caused by JTAG |

## 13.2.3 Reset Pin Filter Control register (RCM_RPFC)

**NOTE**

The reset values of bits 2-0 are for Chip POR only. They are unaffected by other reset types.

**NOTE**

The bus clock filter is reset when disabled or when entering stop mode. The LPO filter is reset when disabled or when entering any low leakage stop mode .

Address: 4007_F000h base + 4h offset = 4007_F004h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | | | | | RSTFLTSS | RSTFLTSRW | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RCM_RPFC field descriptions**

| Field | Description |
|---|---|
| 7–3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>RSTFLTSS | Reset Pin Filter Select in Stop Mode<br><br>Selects how the reset pin filter is enabled in Stop and VLPS modes<br><br>0 All filtering disabled<br>1 LPO clock filter enabled |
| 1–0<br>RSTFLTSRW | Reset Pin Filter Select in Run and Wait Modes<br><br>Selects how the reset pin filter is enabled in run and wait modes.<br><br>00 All filtering disabled<br>01 Bus clock filter enabled for normal operation<br>10 LPO clock filter enabled for normal operation<br>11 Reserved |

## 13.2.4 Reset Pin Filter Width register (RCM_RPFW)

**NOTE**

The reset values of the bits in the RSTFLTSEL field are for Chip POR only. They are unaffected by other reset types.

Address: 4007_F000h base + 5h offset = 4007_F005h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | | | | RSTFLTSEL | | | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RCM_RPFW field descriptions**

| Field | Description |
|---|---|
| 7–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4–0<br>RSTFLTSEL | Reset Pin Filter Bus Clock Select<br><br>Selects the reset pin bus clock filter width.<br><br>00000 Bus clock filter count is 1<br>00001 Bus clock filter count is 2<br>00010 Bus clock filter count is 3<br>00011 Bus clock filter count is 4<br>00100 Bus clock filter count is 5<br>00101 Bus clock filter count is 6<br>00110 Bus clock filter count is 7<br>00111 Bus clock filter count is 8<br>01000 Bus clock filter count is 9<br>01001 Bus clock filter count is 10<br>01010 Bus clock filter count is 11<br>01011 Bus clock filter count is 12<br>01100 Bus clock filter count is 13<br>01101 Bus clock filter count is 14<br>01110 Bus clock filter count is 15<br>01111 Bus clock filter count is 16<br>10000 Bus clock filter count is 17<br>10001 Bus clock filter count is 18<br>10010 Bus clock filter count is 19<br>10011 Bus clock filter count is 20<br>10100 Bus clock filter count is 21<br>10101 Bus clock filter count is 22<br>10110 Bus clock filter count is 23<br>10111 Bus clock filter count is 24<br>11000 Bus clock filter count is 25 |

*Table continues on the next page...*

**RCM_RPFW field descriptions (continued)**

| Field | Description |
|---|---|
| | 11001 Bus clock filter count is 26<br>11010 Bus clock filter count is 27<br>11011 Bus clock filter count is 28<br>11100 Bus clock filter count is 29<br>11101 Bus clock filter count is 30<br>11110 Bus clock filter count is 31<br>11111 Bus clock filter count is 32 |

## 13.2.5 Mode Register (RCM_MR)

This register includes read-only status flags to indicate the state of the mode pins during the last Chip Reset.

Address: 4007_F000h base + 7h offset = 4007_F007h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | | | | | | EZP_MS | 0 |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RCM_MR field descriptions**

| Field | Description |
|---|---|
| 7–2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 1<br>EZP_MS | EZP_MS_B pin state<br><br>Reflects the state of the $\overline{\text{EZP\_MS}}$ pin during the last Chip Reset<br><br>0 Pin deasserted (logic 1)<br>1 Pin asserted (logic 0) |
| 0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

# Chapter 14
# System Mode Controller (SMC)

## 14.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The system mode controller (SMC) is responsible for sequencing the system into and out of all low power stop and run modes. Specifically, it monitors events to trigger transitions between power modes while controlling the power, clocks, and memories of the system to achieve the power consumption and functionality of that mode.

This chapter describes all the available low power modes, the sequence followed to enter/exit each mode, and the functionality available while in each of the modes.

The SMC is able to function during even the deepest low power modes.

## 14.2 Modes of operation

The ARM CPU has three primary modes of operation:

- Run
- Sleep
- Deep Sleep

The WFI or WFE instruction is used to invoke Sleep and Deep Sleep modes. Run, wait and stop are the common terms used for the primary operating modes of Freescale microcontrollers. The following table shows the translation between the ARM CPU modes and the Freescale MCU power modes.

| ARM CPU mode | MCU mode |
|---|---|
| Sleep | Wait |
| Deep Sleep | Stop |

Accordingly, the ARM CPU documentation refers to sleep and deep sleep, while the Freescale MCU documentation normally uses wait and stop.

In addition, Freescale MCUs also augment stop, wait, and run modes in a number of ways. The power management controller (PMC) contains a run and a stop mode regulator. Run regulation is used in normal run, wait and stop modes. Stop mode regulation is used during all very low power and low leakage modes. During stop mode regulation, the bus frequencies are limited in the very low power modes.

The SMC provides the user with multiple power options. The Very Low Power Run (VLPR) mode can drastically reduce run time power when maximum bus frequency is not required to handle the application needs. From Normal Run mode, the Run Mode (RUNM) field can be modified to change the MCU into VLPR mode when limited frequency is sufficient for the application. From VLPR mode, a corresponding wait (VLPW) and stop (VLPS) mode can be entered.

Depending on the needs of the user application, a variety of stop modes are available that allow the state retention, partial power down or full power down of certain logic and/or memory. I/O states are held in all modes of operation. Several registers are used to configure the various modes of operation for the device.

The following table describes the power modes available for the device.

**Table 14-1. Power modes**

| Mode | Description |
|---|---|
| RUN | The MCU can be run at full speed and the internal supply is fully regulated, that is, in run regulation. This mode is also referred to as Normal Run mode. |
| WAIT | The core clock is gated off. The system clock continues to operate. Bus clocks, if enabled, continue to operate. Run regulation is maintained. |
| STOP | The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid. |
| VLPR | The core, system, bus, and flash clock maximum frequencies are restricted in this mode. See the Power Management chapter for details about the maximum allowable frequencies. |
| VLPW | The core clock is gated off. The system, bus, and flash clocks continue to operate, although their maximum frequency is restricted. See the Power Management chapter for details on the maximum allowable frequencies. |
| VLPS | The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid. |

*Table continues on the next page...*

**Table 14-1. Power modes (continued)**

| Mode | Description |
|---|---|
| LLS | The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid. The MCU is placed in a low leakage mode by reducing the voltage to internal logic. All system RAM contents, internal logic and I/O states are retained. |
| VLLS3 | The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid. The MCU is placed in a low leakage mode by powering down the internal logic. All system RAM contents are retained and I/O states are held. Internal logic states are not retained. |
| VLLS2 | The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid.The MCU is placed in a low leakage mode by powering down the internal logic and the system RAM3 partition. The system RAM2 partition can be optionally retained using VLLSCTRL[RAM2PO]. The system RAM1 partition contents are retained in this mode. Internal logic states are not retained. [1] |
| VLLS1 | The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid. The MCU is placed in a low leakage mode by powering down the internal logic and all system RAM. I/O states are held. Internal logic states are not retained. |
| VLLS0 | The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid. The MCU is placed in a low leakage mode by powering down the internal logic and all system RAM. I/O states are held. Internal logic states are not retained. The 1kHz LPO clock is disabled and the power on reset (POR) circuit can be optionally enabled using VLLSCTRL[PORPO]. |

1. See the devices' chip configuration details for the size and location of the system RAM partitions.

## 14.3 Memory map and register descriptions

Details follow about the registers related to the system mode controller.

Different SMC registers reset on different reset types. Each register's description provides details. For more information about the types of reset on this chip, refer to the Reset section details.

**NOTE**

The SMC registers can be written only in supervisor mode. Write accesses in user mode are blocked and will result in a bus error.

**SMC memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4007_E000 | Power Mode Protection register (SMC_PMPROT) | 8 | R/W | 00h | 14.3.1/318 |
| 4007_E001 | Power Mode Control register (SMC_PMCTRL) | 8 | R/W | 00h | 14.3.2/319 |

*Table continues on the next page...*

**SMC memory map (continued)**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4007_E002 | VLLS Control register (SMC_VLLSCTRL) | 8 | R/W | 03h | 14.3.3/321 |
| 4007_E003 | Power Mode Status register (SMC_PMSTAT) | 8 | R | 01h | 14.3.4/322 |

## 14.3.1 Power Mode Protection register (SMC_PMPROT)

This register provides protection for entry into any low-power run or stop mode. The enabling of the low-power run or stop mode occurs by configuring the Power Mode Control register (PMCTRL).

The PMPROT register can be written only once after any system reset.

If the MCU is configured for a disallowed or reserved power mode, the MCU remains in its current power mode. For example, if the MCU is in normal RUN mode and AVLP is 0, an attempt to enter VLPR mode using PMCTRL[RUNM] is blocked and the RUNM bits remain 00b, indicating the MCU is still in Normal Run mode.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Reset section details for more information.

Address: 4007_E000h base + 0h offset = 4007_E000h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | 0 | AVLP | 0 | ALLS | 0 | AVLLS | 0 |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SMC_PMPROT field descriptions**

| Field | Description |
|---|---|
| 7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5<br>AVLP | Allow Very-Low-Power Modes<br><br>Provided the appropriate control bits are set up in PMCTRL, this write-once bit allows the MCU to enter any very-low-power mode (VLPR, VLPW, and VLPS).<br><br>0 VLPR, VLPW and VLPS are not allowed<br>1 VLPR, VLPW and VLPS are allowed |

*Table continues on the next page...*

**SMC_PMPROT field descriptions (continued)**

| Field | Description |
|---|---|
| 4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3<br>ALLS | Allow Low-Leakage Stop Mode<br><br>Provided the appropriate control bits are set up in PMCTRL, this write-once bit allows the MCU to enter any low-leakage stop mode (LLS).<br><br>0 LLS is not allowed<br>1 LLS is allowed |
| 2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 1<br>AVLLS | Allow Very-Low-Leakage Stop Mode<br><br>Provided the appropriate control bits are set up in PMCTRL, this write once bit allows the MCU to enter any very-low-leakage stop mode (VLLSx).<br><br>0 Any VLLSx mode is not allowed<br>1 Any VLLSx mode is allowed |
| 0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 14.3.2 Power Mode Control register (SMC_PMCTRL)

The PMCTRL register controls entry into low-power run and stop modes, provided that the selected power mode is allowed via an appropriate setting of the protection (PMPROT) register.

**NOTE**

This register is reset on Chip POR not VLLS and by reset types that trigger Chip POR not VLLS. It is unaffected by reset types that do not trigger Chip POR not VLLS. See the Reset section details for more information.

Address: 4007_E000h base + 1h offset = 4007_E001h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | LPWUI | RUNM | | 0 | STOPA | STOPM | | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SMC_PMCTRL field descriptions**

| Field | Description |
|---|---|
| 7<br>LPWUI | Low-Power Wake Up On Interrupt |

*Table continues on the next page...*

## SMC_PMCTRL field descriptions (continued)

| Field | Description |
|---|---|
| | Causes the SMC to exit to normal RUN mode when any active MCU interrupt occurs while in a VLP mode (VLPR, VLPW or VLPS).<br><br>**NOTE:** If VLPS mode was entered directly from RUN mode, the SMC will always exit back to normal RUN mode regardless of the LPWUI setting.<br><br>**NOTE:** LPWUI must be modified only while the system is in RUN mode, that is, when PMSTAT=RUN.<br><br>0 The system remains in a VLP mode on an interrupt<br>1 The system exits to Normal RUN mode on an interrupt |
| 6–5<br>RUNM | Run Mode Control<br><br>When written, causes entry into the selected run mode. Writes to this field are blocked if the protection level has not been enabled using the PMPROT register. This field is cleared by hardware on any exit to normal RUN mode.<br><br>**NOTE:** RUNM may be set to VLPR only when PMSTAT=RUN. After being written to VLPR, RUNM should not be written back to RUN until PMSTAT=VLPR.<br><br>00 Normal Run mode (RUN)<br>01 Reserved<br>10 Very-Low-Power Run mode (VLPR)<br>11 Reserved |
| 4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3<br>STOPA | Stop Aborted<br><br>When set, this read-only status bit indicates an interrupt or reset occured during the previous stop mode entry sequence, preventing the system from entering that mode. This bit is cleared by hardware at the beginning of any stop mode entry sequence and is set if the sequence was aborted.<br><br>0 The previous stop mode entry was successful.<br>1 The previous stop mode entry was aborted. |
| 2–0<br>STOPM | Stop Mode Control<br><br>When written, controls entry into the selected stop mode when Sleep-Now or Sleep-On-Exit mode is entered with SLEEPDEEP=1 . Writes to this field are blocked if the protection level has not been enabled using the PMPROT register. After any system reset, this field is cleared by hardware on any successful write to the PMPROT register.<br><br>**NOTE:** When set to VLLSx, the VLLSM bits in the VLLSCTRL register is used to further select the particular VLLS submode which will be entered.<br><br>**NOTE:**<br><br>000 Normal Stop (STOP)<br>001 Reserved<br>010 Very-Low-Power Stop (VLPS)<br>011 Low-Leakage Stop (LLS)<br>100 Very-Low-Leakage Stop (VLLSx)<br>101 Reserved<br>110 Reseved<br>111 Reserved |

### 14.3.3 VLLS Control register (SMC_VLLSCTRL)

The VLLSCTRL register controls features related to VLLS modes.

**NOTE**

This register is reset on Chip POR not VLLS and by reset types that trigger Chip POR not VLLS. It is unaffected by reset types that do not trigger Chip POR not VLLS. See the Reset section details for more information.

Address: 4007_E000h base + 2h offset = 4007_E002h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | | PORPO | RAM2PO | 0 | VLLSM | | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**SMC_VLLSCTRL field descriptions**

| Field | Description |
|---|---|
| 7–6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5<br>PORPO | POR Power Option<br><br>Controls whether the POR detect circuit (for brown-out detection) is enabled in VLLS0 mode.<br><br>0 POR detect circuit is enabled in VLLS0<br>1 POR detect circuit is disabled in VLLS0 |
| 4<br>RAM2PO | RAM2 Power Option<br><br>Controls powering of RAM partition 2 in VLLS2 mode.<br><br>**NOTE:** See the device's chip configuration details for the size and location of RAM parition 2<br><br>0 RAM2 not powered in VLLS2<br>1 RAM2 powered in VLLS2 |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2–0<br>VLLSM | VLLS Mode Control<br><br>Controls which VLLS sub-mode to enter if STOPM=VLLS.<br><br>000 VLLS0<br>001 VLLS1<br>010 VLLS2<br>011 VLLS3<br>100 Reserved<br>101 Reserved<br>110 Reserved<br>111 Reserved |

### 14.3.4 Power Mode Status register (SMC_PMSTAT)

PMSTAT is a read-only, one-hot register which indicates the current power mode of the system.

**NOTE**

This register is reset on Chip POR not VLLS and by reset types that trigger Chip POR not VLLS. It is unaffected by reset types that do not trigger Chip POR not VLLS. See the Reset section details for more information.

Address: 4007_E000h base + 3h offset = 4007_E003h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | PMSTAT | | | | | | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**SMC_PMSTAT field descriptions**

| Field | Description |
|---|---|
| 7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6–0<br>PMSTAT | **NOTE:** When debug is enabled, the PMSTAT will not update to STOP or VLPS<br><br>000_0001 Current power mode is RUN<br>000_0010 Current power mode is STOP<br>000_0100 Current power mode is VLPR<br>000_1000 Current power mode is VLPW<br>001_0000 Current power mode is VLPS<br>010_0000 Current power mode is LLS<br>100_0000 Current power mode is VLLS |

## 14.4 Functional description

### 14.4.1 Power mode transitions

The following figure shows the power mode state transitions available on the chip. Any reset always brings the MCU back to the normal run state.

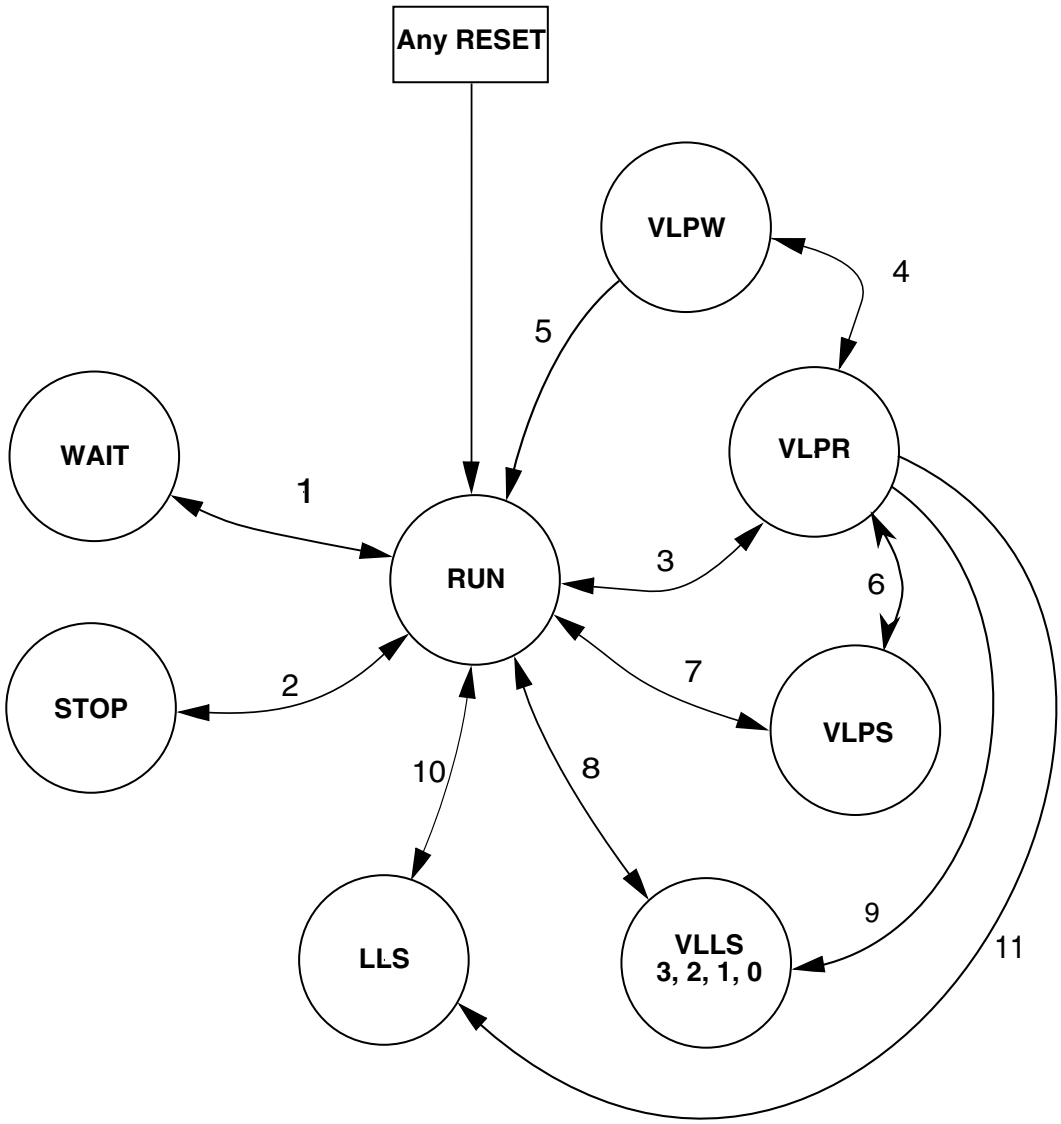



**Figure 14-5. Power mode state diagram**

The following table defines triggers for the various state transitions shown in the previous figure.

**Table 14-7. Power mode transition triggers**

| Transition # | From | To | Trigger conditions |
|---|---|---|---|
| 1 | RUN | WAIT | Sleep-now or sleep-on-exit modes entered with SLEEPDEEP clear, controlled in System Control Register in ARM core.<br>See note.[1] |
| | WAIT | RUN | Interrupt or Reset |

*Table continues on the next page...*

## Table 14-7. Power mode transition triggers (continued)

| Transition # | From | To | Trigger conditions |
|---|---|---|---|
| 2 | RUN | STOP | PMCTRL[RUNM]=00, PMCTRL[STOPM]=000<br>Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core.<br>See note.[1] |
| | STOP | RUN | Interrupt or Reset |
| 3 | RUN | VLPR | The core, system, bus and flash clock frequencies and MCG clocking mode are restricted in this mode. See the Power Management chapter for the maximum allowable frequencies and MCG modes supported.<br>Set PMPROT[AVLP]=1, PMCTRL[RUNM]=10. |
| | VLPR | RUN | Set PMCTRL[RUNM]=00 or<br>Interrupt with PMCTRL[LPWUI] =1 or<br>Reset. |
| 4 | VLPR | VLPW | Sleep-now or sleep-on-exit modes entered with SLEEPDEEP clear, which is controlled in System Control Register in ARM core.<br>See note.[1] |
| | VLPW | VLPR | Interrupt with PMCTRL[LPWUI]=0 |
| 5 | VLPW | RUN | Interrupt with PMCTRL[LPWUI]=1 or<br>Reset |
| 6 | VLPR | VLPS | PMCTRL[STOPM]=000 or 010,<br>Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core.<br>See note.[1] |
| | VLPS | VLPR | Interrupt with PMCTRL[LPWUI]=0<br>**NOTE:** If VLPS was entered directly from RUN, hardware will not allow this transition and will force exit back to RUN |
| 7 | RUN | VLPS | PMPROT[AVLP]=1, PMCTRL[STOPM]=010,<br>Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core.<br>See note.[1] |
| | VLPS | RUN | Interrupt with PMCTRL[LPWUI]=1 or<br>Interrupt with PMCTRL[LPWUI]=0 and VLPS mode was entered directly from RUN or<br>Reset |

*Table continues on the next page...*

**Table 14-7. Power mode transition triggers (continued)**

| Transition # | From | To | Trigger conditions |
|---|---|---|---|
| 8 | RUN | VLLSx | PMPROT[AVLLS]=1, PMCTRL[STOPM]=100, VLLSCTRL[VLLSM]=x (VLLSx), Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core. |
| | VLLSx | RUN | Wakeup from enabled LLWU input source or RESET pin |
| 9 | VLPR | VLLSx | PMPROT[AVLLS]=1, PMCTRL[STOPM]=100, VLLSCTRL[VLLSM]=x (VLLSx), Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core. |
| 10 | RUN | LLS | PMPROT[ALLS]=1, PMCTRL[STOPM]=011, Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core. |
| | LLS | RUN | Wakeup from enabled LLWU input source and LLS mode was entered directly from RUN or<br>RESET pin. |
| 11 | VLPR | LLS | PMPROT[ALLS]=1, PMCTRL[STOPM]=011, Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core. |
| | LLS | VLPR | Wakeup from enabled LLWU input source and LLS mode was entered directly from VLPR<br>**NOTE:** If LLS was entered directly from RUN, hardware will not allow this transition and will force exit back to RUN |

1. If debug is enabled, the core clock remains to support debug.

## 14.4.2 Power mode entry/exit sequencing

When entering or exiting low-power modes, the system must conform to an orderly sequence to manage transitions safely. The SMC manages the system's entry into and exit from all power modes. The following diagram illustrates the connections of the SMC with other system components in the chip that are necessary to sequence the system through all power modes.

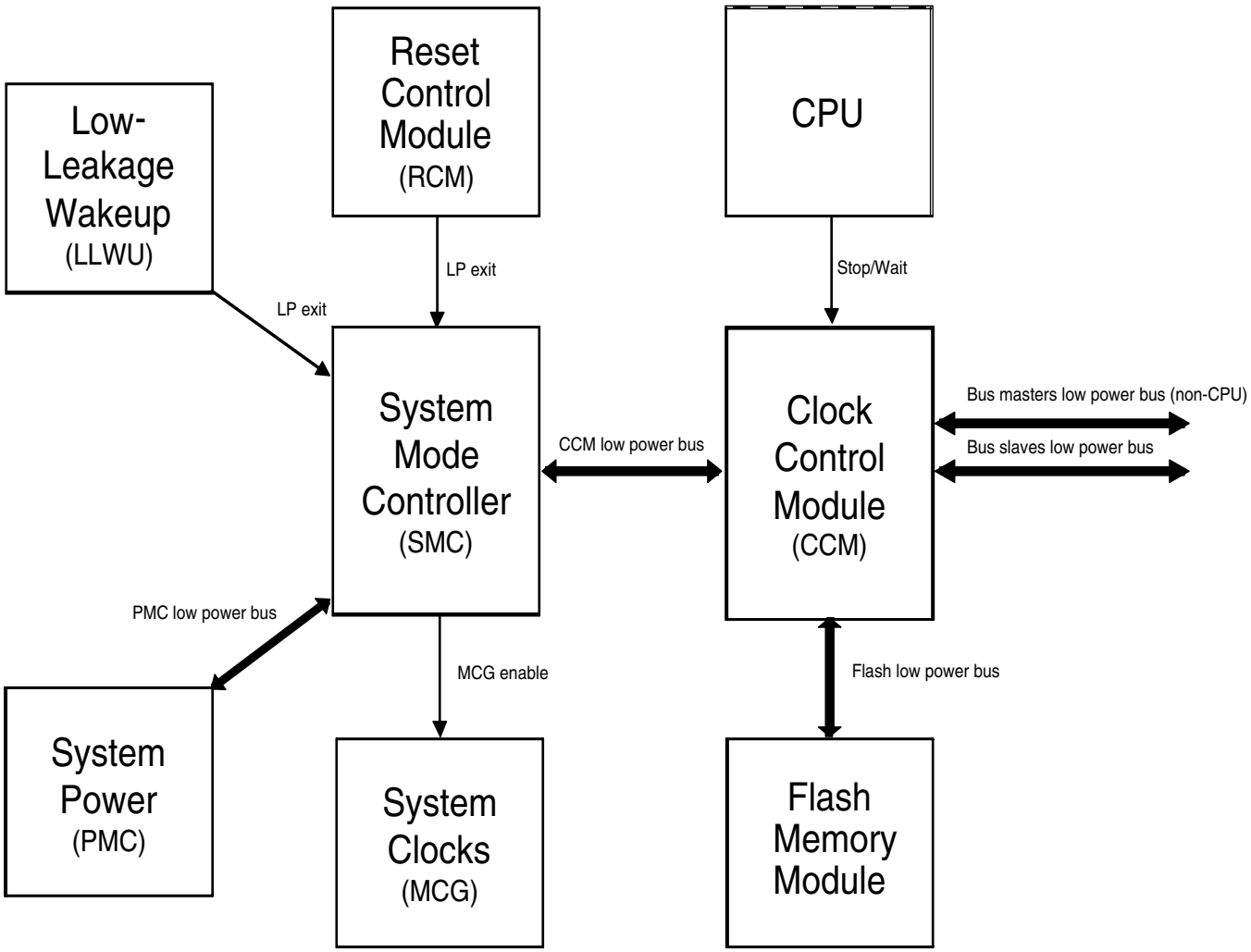



**Figure 14-6. Low-power system components and connections**

### 14.4.2.1 Stop mode entry sequence

Entry into a low-power stop mode (Stop, VLPS, LLS, VLLSx) is initiated by CPU execution of the WFI instruction. After the instruction is executed, the following sequence occurs:

1. The CPU clock is gated off immediately.
2. Requests are made to all non-CPU bus masters to enter Stop mode.
3. After all masters have acknowledged they are ready to enter Stop mode, requests are made to all bus slaves to enter Stop mode.
4. After all slaves have acknowledged they are ready to enter Stop mode, all system and bus clocks are gated off.
5. Clock generators are disabled in the MCG.
6. The on-chip regulator in the PMC and internal power switches are configured to meet the power consumption goals for the targeted low-power mode.

#### 14.4.2.2 Stop mode exit sequence

Exit from a low-power stop mode is initiated either by a reset or an interrupt event. The following sequence then executes to restore the system to a run mode (RUN or VLPR):

1. The on-chip regulator in the PMC and internal power switches are restored.
2. Clock generators are enabled in the MCG.
3. System and bus clocks are enabled to all masters and slaves.
4. The CPU clock is enabled and the CPU begins servicing the reset or interrupt that initiated the exit from the low-power stop mode.

#### 14.4.2.3 Aborted stop mode entry

If an interrupt or a reset occurs during a stop entry sequence, the SMC can abort the transition early and return to RUN mode without completely entering the stop mode. An aborted entry is possible only if the reset or interrupt occurs before the PMC begins the transition to stop mode regulation. After this point, the interrupt or reset is ignored until the PMC has completed its transition to stop mode regulation. When an aborted stop mode entry sequence occurs, the SMC's PMCTRL[STOPA] is set to 1.

#### 14.4.2.4 Transition to wait modes

For wait modes (WAIT and VLPW), the CPU clock is gated off while all other clocking continues, as in RUN and VLPR mode operation. Some modules that support stop-in-wait functionality have their clocks disabled in these configurations.

#### 14.4.2.5 Transition from stop modes to Debug mode

The debugger module supports a transition from STOP, WAIT, VLPS, and VLPW back to a Halted state when the debugger has been enabled, that is, ENBDM is 1. As part of this transition, system clocking is re-established and is equivalent to the normal RUN and VLPR mode clocking configuration.

### 14.4.3 Run modes

The device supports the following run modes:

- Run (RUN)
- Very Low-Power Run (VLPR)

#### 14.4.3.1 RUN mode

This is the normal operating mode for the device.

This mode is selected after any reset. When the ARM processor exits reset, it sets up the stack, program counter (PC), and link register (LR):

- The processor reads the start SP (SP_main) from vector-table offset 0x000
- The processor reads the start PC from vector-table offset 0x004
- LR is set to 0xFFFF_FFFF.

To reduce power in this mode, disable the clocks to unused modules using their corresponding clock gating control bits in the SIM's registers.

#### 14.4.3.2 Very-Low Power Run (VLPR) mode

In VLPR mode, the on-chip voltage regulator is put into a stop mode regulation state. In this state, the regulator is designed to supply enough current to the MCU over a reduced frequency. To further reduce power in this mode, disable the clocks to unused modules using their corresponding clock gating control bits in the SIM's registers.

Before entering this mode, the following conditions must be met:

- The MCG must be configured in a mode which is supported during VLPR. See the Power Management details for information about these MCG modes.
- All clock monitors in the MCG must be disabled.
- The maximum frequencies of the system, bus, flash, and core are restricted. See the Power Management details about which frequencies are supported.
- Mode protection must be set to allow VLP modes, that is, PMPROT[AVLP] is 1.
- PMCTRL[RUNM] is set to 10b to enter VLPR.
- Flash programming/erasing is not allowed.

**NOTE**

Do not change the clock frequency while in VLPR mode, because the regulator is slow responding and cannot manage fast load transitions. In addition, do not modify the clock source in the MCG module, the module clock enables in the SIM, or any clock divider registers.

To reenter Normal Run mode, clear RUNM. The PMSTAT register is a read-only status register that can be used to determine when the system has completed an exit to RUN mode. When PMSTAT=RUN, the system is in run regulation and the MCU can run at full speed in any clock mode. If a higher execution frequency is desired, poll the PMSTAT register until it is set to RUN when returning from VLPR mode.

VLPR mode also provides the option to return to run regulation if any interrupt occurs. Implement this option by setting Low-Power Wakeup On Interrupt (LPWUI) in the PMCTRL register. Any reset always causes an exit from VLPR and returns the device to RUN mode after the MCU exits its reset flow. The RUNM bits are cleared by hardware on any interrupt when LPWUI is set or on any reset.

### 14.4.4 Wait modes

This device contains two different wait modes:

- Wait
- Very-Low Power Wait (VLPW)

#### 14.4.4.1 WAIT mode

WAIT mode is entered when the ARM core enters the Sleep-Now or Sleep-On-Exit modes while SLEEDEEP is cleared. The ARM CPU enters a low-power state in which it is not clocked, but peripherals continue to be clocked provided they are enabled. Clock gating to the peripheral is enabled via the SIM..

When an interrupt request occurs, the CPU exits WAIT mode and resumes processing in RUN mode, beginning with the stacking operations leading to the interrupt service routine.

A system reset will cause an exit from WAIT mode, returning the device to normal RUN mode.

#### 14.4.4.2 Very-Low-Power Wait (VLPW) mode

VLPW is entered by the entering the Sleep-Now or Sleep-On-Exit mode while SLEEPDEEP is cleared and the MCU is in VLPR mode.

In VLPW, the on-chip voltage regulator remains in its stop regulation state. In this state, the regulator is designed to supply enough current to the MCU over a reduced frequency. To further reduce power in this mode, disable the clocks to unused modules by clearing the peripherals' corresponding clock gating control bits in the SIM.

VLPR mode restrictions also apply to VLPW.

VLPW mode provides the option to return to fully-regulated normal RUN mode if any enabled interrupt occurs. This is done by setting PMCTRL[LPWUI]. Wait for the PMSTAT register to set to RUN before increasing the frequency.

If the LPWUI bit is clear, when an interrupt from VLPW occurs, the device returns to VLPR mode to execute the interrupt service routine.

A system reset will cause an exit from VLPW mode, returning the device to normal RUN mode.

### 14.4.5 Stop modes

This device contains a variety of stop modes to meet your application needs. The stop modes range from:

- a stopped CPU, with all I/O, logic, and memory states retained, and certain asynchronous mode peripherals operating

to:

- a powered down CPU, with only I/O and a small register file retained, very few asynchronous mode peripherals operating, while the remainder of the MCU is powered down.

The choice of stop mode depends upon the user's application, and how power usage and state retention versus functional needs and recovery time may be traded off.

**NOTE**

All clock monitors must be disabled before entering these low-power modes: Stop, VLPS, VLPR, VLPW, LLS, and VLLSx.

The various stop modes are selected by setting the appropriate fields in PMPROT and PMCTRL. The selected stop mode is entered during the sleep-now or sleep-on-exit entry with the SLEEPDEEP bit set in the System Control Register in the ARM core.

The available stop modes are:

- Normal Stop (STOP)
- Very-Low Power Stop (VLPS)

- Low-Leakage Stop (LLS)
- Very-Low-Leakage Stop (VLLSx)

#### 14.4.5.1 STOP mode

STOP mode is entered via the sleep-now or sleep-on-exit with the SLEEPDEEP bit set in the System Control Register in the ARM core.

The MCG module can be configured to leave the reference clocks running.

A module capable of providing an asynchronous interrupt to the device takes the device out of STOP mode and returns the device to normal RUN mode. Refer to the device's Power Management chapter for peripheral, I/O, and memory operation in STOP mode. When an interrupt request occurs, the CPU exits STOP mode and resumes processing, beginning with the stacking operations leading to the interrupt service routine.

A system reset will cause an exit from STOP mode, returning the device to normal RUN mode via an MCU reset.

#### 14.4.5.2 Very-Low-Power Stop (VLPS) mode

VLPS mode can be entered in one of two ways:

- Entry into stop via the sleep-now or sleep-on-exit with the SLEEPDEEP bit set in the System Control Register in the ARM core while the MCU is in VLPR mode and STOPM=010 or 000 in the PMCTRL register.
- Entry into stop via the sleep-now or sleep-on-exit with the SLEEPDEEP bit set in the System Control Register in the ARM core while the MCU is in normal RUN mode and STOPM=010 in the PMCTRL register. When VLPS is entered directly from RUN mode, exit to VLPR is disabled by hardware and the system will always exit back to RUN.

In VLPS, the on-chip voltage regulator remains in its stop regulation state as in VLPR.

A module capable of providing an asynchronous interrupt to the device takes the device out of VLPS and returns the device to VLPR mode, provided LPWUI is clear.

If LPWUI is set, the device returns to normal RUN mode upon an interrupt request. PMSTAT must be set to RUN before allowing the system to return to a frequency higher than that allowed in VLPR mode.

A system reset will also cause a VLPS exit, returning the device to normal RUN mode.

### 14.4.5.3 Low-Leakage Stop (LLS) mode

Low-Leakage Stop (LLS) mode can be entered from normal RUN or VLPR modes.

The MCU enters LLS mode if:

- In Sleep-Now or Sleep-On-Exit mode, SLEEPDEEP is set in the System Control Register in the ARM core, and
- The device is configured as shown in Table 14-7.

In LLS, the on-chip voltage regulator is in stop regulation. Most of the peripherals are put in a state-retention mode that does not allow them to operate while in LLS.

Before entering LLS mode, the user should configure the low-leakage wakeup (LLWU) module to enable the desired wakeup sources. The available wakeup sources in LLS are detailed in the chip configuration details for this device.

After wakeup from LLS, the device returns to the run mode from which LLS was entered (either normal RUN or VLPR) with a pending LLWU module interrupt. In the LLWU interrupt service routine (ISR), the user can poll the LLWU module wakeup flags to determine the source of the wakeup.

**NOTE**

The LLWU interrupt must not be masked by the interrupt controller to avoid a scenario where the system does not fully exit stop mode on an LLS recovery.

An asserted $\overline{\text{RESET}}$ pin will cause an exit from LLS mode, returning the device to normal RUN mode. When LLS is exiting via the $\overline{\text{RESET}}$ pin, the PIN and WAKEUP bits are set in the SRS0 register of the reset control module (RCM).

### 14.4.5.4 Very-Low-Leakage Stop (VLLSx) modes

This device contains these very low leakage modes:

- VLLS3
- VLLS2
- VLLS1
- VLLS0

VLLSx is often used in this document to refer to all of these modes.

All VLLSx modes can be entered from normal RUN or VLPR modes.

The MCU enters the configured VLLS mode if:

- In Sleep-Now or Sleep-On-Exit mode, the SLEEPDEEP bit is set in the System Control Register in the ARM core, and
- The device is configured as shown in Table 14-7.

In VLLS, the on-chip voltage regulator is in its stop-regulation state while most digital logic is powered off.

Before entering VLLS mode, the user should configure the low-leakage wakeup (LLWU) module to enable the desired wakeup sources. The available wakeup sources in VLLS are detailed in the chip configuration details for this device.

After wakeup from VLLS, the device returns to normal RUN mode with a pending LLWU interrupt. In the LLWU interrupt service routine (ISR), the user can poll the LLWU module wakeup flags to determine the source of the wakeup.

When entering VLLS, each I/O pin is latched as configured before executing VLLS. Because all digital logic in the MCU is powered off, all port and peripheral data is lost during VLLS. This information must be restored before the ACKISO bit in the PMC is set.

An asserted $\overline{\text{RESET}}$ pin will cause an exit from any VLLS mode, returning the device to normal RUN mode. When exiting VLLS via the $\overline{\text{RESET}}$ pin, the PIN and WAKEUP bits are set in the SRS0 register of the reset control module (RCM).

### 14.4.6 Debug in low power modes

When the MCU is secure, the device disables/limits debugger operation. When the MCU is unsecure, the ARM debugger can assert two power-up request signals:

- System power up, via SYSPWR in the Debug Port Control/Stat register
- Debug power up, via CDBGPWRUPREQ in the Debug Port Control/Stat register

When asserted while in RUN, WAIT, VLPR, or VLPW, the mode controller drives a corresponding acknowledge for each signal, that is, both CDBGPWRUPACK and CSYSPWRUPACK. When both requests are asserted, the mode controller handles attempts to enter STOP and VLPS by entering an emulated stop state. In this emulated stop state:

- the regulator is in run regulation,
- the MCG-generated clock source is enabled,
- all system clocks, except the core clock, are disabled,
- the debug module has access to core registers, and
- access to the on-chip peripherals is blocked.

No debug is available while the MCU is in LLS or VLLS modes. LLS is a state-retention mode and all debug operation can continue after waking from LLS, even in cases where system wakeup is due to a system reset event.

Entering into a VLLS mode causes all of the debug controls and settings to be powered off. To give time to the debugger to sync with the MCU, the MDM AP Control Register includes a Very-Low-Leakage Debug Request (VLLDBGREQ) bit that is set to configure the Reset Controller logic to hold the system in reset after the next recovery from a VLLS mode. This bit allows the debugger time to reinitialize the debug module before the debug session continues.

The MDM AP Control Register also includes a Very Low Leakage Debug Acknowledge (VLLDBGACK) bit that is set to release the ARM core being held in reset following a VLLS recovery. The debugger reinitializes all debug IP, and then asserts the VLLDBGACK control bit to allow the RCM to release the ARM core from reset and allow CPU operation to begin.

The VLLDBGACK bit is cleared by the debugger (or can be left set as is) or clears automatically due to the reset generated as part of the next VLLS recovery.

# Chapter 15
# Power Management Controller (PMC)

## 15.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The power management controller (PMC) contains the internal voltage regulator, power on reset (POR), and low voltage detect system.

## 15.2 Features

The PMC features include:

- Internal voltage regulator
- Active POR providing brown-out detect
- Low-voltage detect supporting two low-voltage trip points with four warning levels per trip point

## 15.3 Low-voltage detect (LVD) system

This device includes a system to guard against low-voltage conditions. This protects memory contents and controls MCU system states during supply voltage variations. The system is comprised of a power-on reset (POR) circuit and a LVD circuit with a user-selectable trip voltage: high ($V_{LVDH}$) or low ($V_{LVDL}$). The trip voltage is selected by the LVDSC1[LVDV] bits. The LVD is disabled upon entering VLPx, LLS, and VLLSx modes.

Two flags are available to indicate the status of the low-voltage detect system:

- The low voltage detect flag (LVDF) operates in a level sensitive manner. The LVDF bit is set when the supply voltage falls below the selected trip point (VLVD). The LVDF bit is cleared by writing one to the LVDACK bit, but only if the internal supply has returned above the trip point; otherwise, the LVDF bit remains set.
- The low voltage warning flag (LVWF) operates in a level sensitive manner. The LVWF bit is set when the supply voltage falls below the selected monitor trip point (VLVW). The LVWF bit is cleared by writing one to the LVWACK bit, but only if the internal supply has returned above the trip point; otherwise, the LVWF bit remains set.

### 15.3.1 LVD reset operation

By setting the LVDRE bit, the LVD generates a reset upon detection of a low voltage condition. The low voltage detection threshold is determined by the LVDV bits. After an LVD reset occurs, the LVD system holds the MCU in reset until the supply voltage rises above this threshold. The LVD bit in the SRS register is set following an LVD or power-on reset.

### 15.3.2 LVD interrupt operation

By configuring the LVD circuit for interrupt operation (LVDIE set and LVDRE clear), LVDSC1[LVDF] is set and an LVD interrupt request occurs upon detection of a low voltage condition. The LVDF bit is cleared by writing one to the LVDSC1[LVDACK] bit.

### 15.3.3 Low-voltage warning (LVW) interrupt operation

The LVD system contains a low-voltage warning flag (LVWF) to indicate that the supply voltage is approaching, but is above, the LVD voltage. The LVW also has an interrupt, which is enabled by setting the LVDSC2[LVWIE] bit. If enabled, an LVW interrupt request occurs when the LVWF is set. LVWF is cleared by writing one to the LVDSC2[LVWACK] bit.

The LVDSC2[LVWV] bits select one of four trip voltages:

- Highest: $V_{LVW4}$
- Two mid-levels: $V_{LVW3}$ and $V_{LVW2}$
- Lowest: $V_{LVW1}$

## 15.4 I/O retention

When in LLS mode, the I/O pins are held in their input or output state. Upon wakeup, the PMC is re-enabled, goes through a power up sequence to full regulation, and releases the logic from state retention mode. The I/O are released immediately after a wakeup or reset event. In the case of LLS exit via a RESET pin, the I/O default to their reset state.

When in VLLS modes, the I/O states are held on a wakeup event (with the exception of wakeup by reset event) until the wakeup has been acknowledged via a write to the ACKISO bit. In the case of VLLS exit via a RESET pin, the I/O are released and default to their reset state. In this case, no write to the ACKISO is needed.

## 15.5 Memory map and register descriptions

PMC register details follow.

**NOTE**

Different portions of PMC registers are reset only by particular reset types. Each register's description provides details. For more information about the types of reset on this chip, refer to the Reset section details.

The PMC registers can be written only in supervisor mode. Write accesses in user mode are blocked and will result in a bus error.

**PMC memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4007_D000 | Low Voltage Detect Status And Control 1 register (PMC_LVDSC1) | 8 | R/W | 10h | 15.5.1/338 |
| 4007_D001 | Low Voltage Detect Status And Control 2 register (PMC_LVDSC2) | 8 | R/W | 00h | 15.5.2/339 |
| 4007_D002 | Regulator Status And Control register (PMC_REGSC) | 8 | R/W | 04h | 15.5.3/340 |

### 15.5.1 Low Voltage Detect Status And Control 1 register (PMC_LVDSC1)

This register contains status and control bits to support the low voltage detect function. This register should be written during the reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

While the device is in the very low power or low leakage modes, the LVD system is disabled regardless of LVDSC1 settings. To protect systems that must have LVD always on, configure the SMC's power mode protection register (PMPROT) to disallow any very low power or low leakage modes from being enabled.

See the device's data sheet for the exact LVD trip voltages.

**NOTE**

The LVDV bits are reset solely on a POR Only event. The register's other bits are reset on Chip Reset Not VLLS. For more information about these reset types, refer to the Reset section details.

Address: 4007_D000h base + 0h offset = 4007_D000h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | LVDF | 0 | LVDIE | LVDRE | 0 | | LVDV | |
| Write | | LVDACK | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**PMC_LVDSC1 field descriptions**

| Field | Description |
|---|---|
| 7<br>LVDF | Low-Voltage Detect Flag<br><br>This read-only status bit indicates a low-voltage detect event.<br><br>0 Low-voltage event not detected<br>1 Low-voltage event detected |
| 6<br>LVDACK | Low-Voltage Detect Acknowledge<br><br>This write-only bit is used to acknowledge low voltage detection errors. Write 1 to clear LVDF. Reads always return 0. |
| 5<br>LVDIE | Low-Voltage Detect Interrupt Enable<br><br>Enables hardware interrupt requests for LVDF.<br><br>0 Hardware interrupt disabled (use polling)<br>1 Request a hardware interrupt when LVDF = 1 |

*Table continues on the next page...*

**PMC_LVDSC1 field descriptions (continued)**

| Field | Description |
|---|---|
| 4<br>LVDRE | Low-Voltage Detect Reset Enable<br><br>This write-once bit enables LVDF events to generate a hardware reset. Additional writes are ignored.<br><br>0 LVDF does not generate hardware resets<br>1 Force an MCU reset when LVDF = 1 |
| 3–2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 1–0<br>LVDV | Low-Voltage Detect Voltage Select<br><br>Selects the LVD trip point voltage ($V_{LVD}$).<br><br>00 Low trip point selected ($V_{LVD} = V_{LVDL}$)<br>01 High trip point selected ($V_{LVD} = V_{LVDH}$)<br>10 Reserved<br>11 Reserved |

## 15.5.2 Low Voltage Detect Status And Control 2 register (PMC_LVDSC2)

This register contains status and control bits to support the low voltage warning function.

While the device is in the very low power or low leakage modes, the LVD system is disabled regardless of LVDSC2 settings.

See the device's data sheet for the exact LVD trip voltages.

**NOTE**

The LVW trip voltages depend on LVWV and LVDV bits.

**NOTE**

The LVWV bits are reset solely on a POR Only event. The register's other bits are reset on Chip Reset Not VLLS. For more information about these reset types, refer to the Reset section details.

Address: 4007_D000h base + 1h offset = 4007_D001h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | LVWF | 0 | LVWIE | 0 | | | LVWV | |
| Write | | LVWACK | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PMC_LVDSC2 field descriptions**

| Field | Description |
|---|---|
| 7<br>LVWF | Low-Voltage Warning Flag<br><br>This read-only status bit indicates a low-voltage warning event. LVWF is set when $V_{Supply}$ transitions below the trip point, or after reset and $V_{Supply}$ is already below $V_{LVW}$. LVWF bit may be 1 after power on reset, therefore, to use LVW interrupt function, before enabling LVWIE, LVWF must be cleared by writing LVWACK first.<br><br>0 Low-voltage warning event not detected<br>1 Low-voltage warning event detected |
| 6<br>LVWACK | Low-Voltage Warning Acknowledge<br><br>This write-only bit is used to acknowledge low voltage warning errors. Write 1 to clear LVWF. Reads always return 0. |
| 5<br>LVWIE | Low-Voltage Warning Interrupt Enable<br><br>Enables hardware interrupt requests for LVWF.<br><br>0 Hardware interrupt disabled (use polling)<br>1 Request a hardware interrupt when LVWF = 1 |
| 4–2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 1–0<br>LVWV | Low-Voltage Warning Voltage Select<br><br>Selects the LVW trip point voltage ($V_{LVW}$). The actual voltage for the warning depends on LVDSC1[LVDV].<br><br>00 Low trip point selected ($V_{LVW} = V_{LVW1}$)<br>01 Mid 1 trip point selected ($V_{LVW} = V_{LVW2}$)<br>10 Mid 2 trip point selected ($V_{LVW} = V_{LVW3}$)<br>11 High trip point selected ($V_{LVW} = V_{LVW4}$) |

## 15.5.3 Regulator Status And Control register (PMC_REGSC)

The PMC contains an internal voltage regulator. The voltage regulator design uses a bandgap reference that is also available through a buffer as input to certain internal peripherals, such as the CMP and ADC. The internal regulator provides a status bit (REGONS) indicating the regulator is in run regulation.

**NOTE**

This register is reset on Chip Reset Not VLLS and by reset types that trigger Chip Reset not VLLS. See the Reset section details for more information.

Address: 4007_D000h base + 2h offset = 4007_D002h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | | Reserved | BGEN | ACKISO | REGONS | Reserved | BGBE |
| Write | | | | | w1c | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

## PMC_REGSC field descriptions

| Field | Description |
|---|---|
| 7–6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5<br>Reserved | This field is reserved. |
| 4<br>BGEN | Bandgap Enable In VLPx Operation<br><br>BGEN controls whether the bandgap is enabled in lower power modes of operation (VLPx, LLS, and VLLSx). When on-chip peripherals require the bandgap voltage reference in low power modes of operation, set BGEN to continue to enable the bandgap operation.<br><br>**NOTE:** When the bandgap voltage reference is not needed in low power modes, clear BGEN to avoid excess power consumption.<br><br>0 Bandgap voltage reference is disabled in VLPx , LLS , and VLLSx modes<br>1 Bandgap voltage reference is enabled in VLPx , LLS , and VLLSx modes |
| 3<br>ACKISO | Acknowledge Isolation<br><br>Reading this bit indicates whether certain peripherals and the I/O pads are in a latched state as a result of having been in a VLLS mode. Writing one to this bit when it is set releases the I/O pads and certain peripherals to their normal run mode state.<br><br>**NOTE:** After recovering from a VLLS mode, user should restore chip configuration before clearing ACKISO. In particular, pin configuration for enabled LLWU wakeup pins should be restored to avoid any LLWU flag from being falsely set when ACKISO is cleared.<br><br>0 Peripherals and I/O pads are in normal run state<br>1 Certain peripherals and I/O pads are in an isolated and latched state |
| 2<br>REGONS | Regulator In Run Regulation Status<br><br>This read-only bit provides the current status of the internal voltage regulator.<br><br>0 Regulator is in stop regulation or in transition to/from it<br>1 Regulator is in run regulation |
| 1<br>Reserved | This field is reserved.<br><br>**NOTE:** This reserved bit must remain cleared (set to 0). |
| 0<br>BGBE | Bandgap Buffer Enable<br><br>Enables the bandgap buffer.<br><br>0 Bandgap buffer not enabled<br>1 Bandgap buffer enabled |

# Chapter 16
# Low-Leakage Wakeup Unit (LLWU)

## 16.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The LLWU module allows the user to select up to 16 external pin sources and up to 8 internal modules as a wakeup source from low-leakage power modes. The input sources are described in the device's chip configuration details. Each of the available wakeup sources can be individually enabled.

The $\overline{\text{RESET}}$ pin is an additional source for triggering an exit from low-leakage power modes, and causes the MCU to exit both LLS and VLLS through a reset flow. The LLWU_RST[LLRSTE] bit must be set to allow an exit from low-leakage modes via the $\overline{\text{RESET}}$ pin. On a device where the $\overline{\text{RESET}}$ pin is shared with other functions, the explicit port mux control register must be set for the $\overline{\text{RESET}}$ pin before the $\overline{\text{RESET}}$ pin can be used as a low-leakage reset source.

The LLWU module also includes optional digital pin filters: two for the external wakeup pins and one for the $\overline{\text{RESET}}$ pin.

### 16.1.1 Features

The LLWU module features include:

- Support for up to 16 external input pins and up to 8 internal modules with individual enable bits
- Input sources may be external pins or from internal peripherals capable of running in LLS or VLLS. See the chip configuration information for wakeup input sources for this device.

- External pin wakeup inputs, each of which is programmable as falling-edge, rising-edge, or any change
- Wakeup inputs that are activated if enabled after MCU enters a low-leakage power mode
- Optional digital filters provided to qualify an external pin detect and $\overline{\text{RESET}}$ pin detect. Note that when the LPO clock is disabled, the filters are disabled and bypassed.

## 16.1.2 Modes of operation

The LLWU module becomes functional on entry into a low-leakage power mode. After recovery from LLS, the LLWU is immediately disabled. After recovery from VLLS, the LLWU continues to detect wakeup events until the user has acknowledged the wakeup via a write to the PMC_REGSC[ACKISO] bit.

### 16.1.2.1 LLS mode

Wakeup events due to external wakeup inputs and internal module wakeup inputs result in an interrupt flow when exiting LLS. A reset event due to $\overline{\text{RESET}}$ pin assertion results in a reset flow when exiting LLS.

**NOTE**

The LLWU interrupt must not be masked by the interrupt controller to avoid a scenario where the system does not fully exit Stop mode on an LLS recovery.

### 16.1.2.2 VLLS modes

All wakeup and reset events result in VLLS exit via a reset flow.

### 16.1.2.3 Non-low leakage modes

The LLWU is not active in all non-low leakage modes where detection and control logic are in a static state. The LLWU registers are accessible in non-low leakage modes and are available for configuring and reading status when bus transactions are possible.

When the $\overline{\text{RESET}}$ pin filter or wakeup pin filters are enabled, filter operation begins immediately. If a low leakage mode is entered within 5 LPO clock cycles of an active edge, the edge event will be detected by the LLWU. For $\overline{\text{RESET}}$ pin filtering, this means that there is no restart to the minimum LPO cycle duration as the filtering transitions from a non-low leakage filter, which is implemented in the RCM, to the LLWU filter.

#### 16.1.2.4 Debug mode

When the chip is in Debug mode and then enters LLS or a VLLSx mode, no debug logic works in the fully-functional low-leakage mode. Upon an exit from the LLS or VLLSx mode, the LLWU becomes inactive.

### 16.1.3 Block diagram

The following figure is the block diagram for the LLWU module.

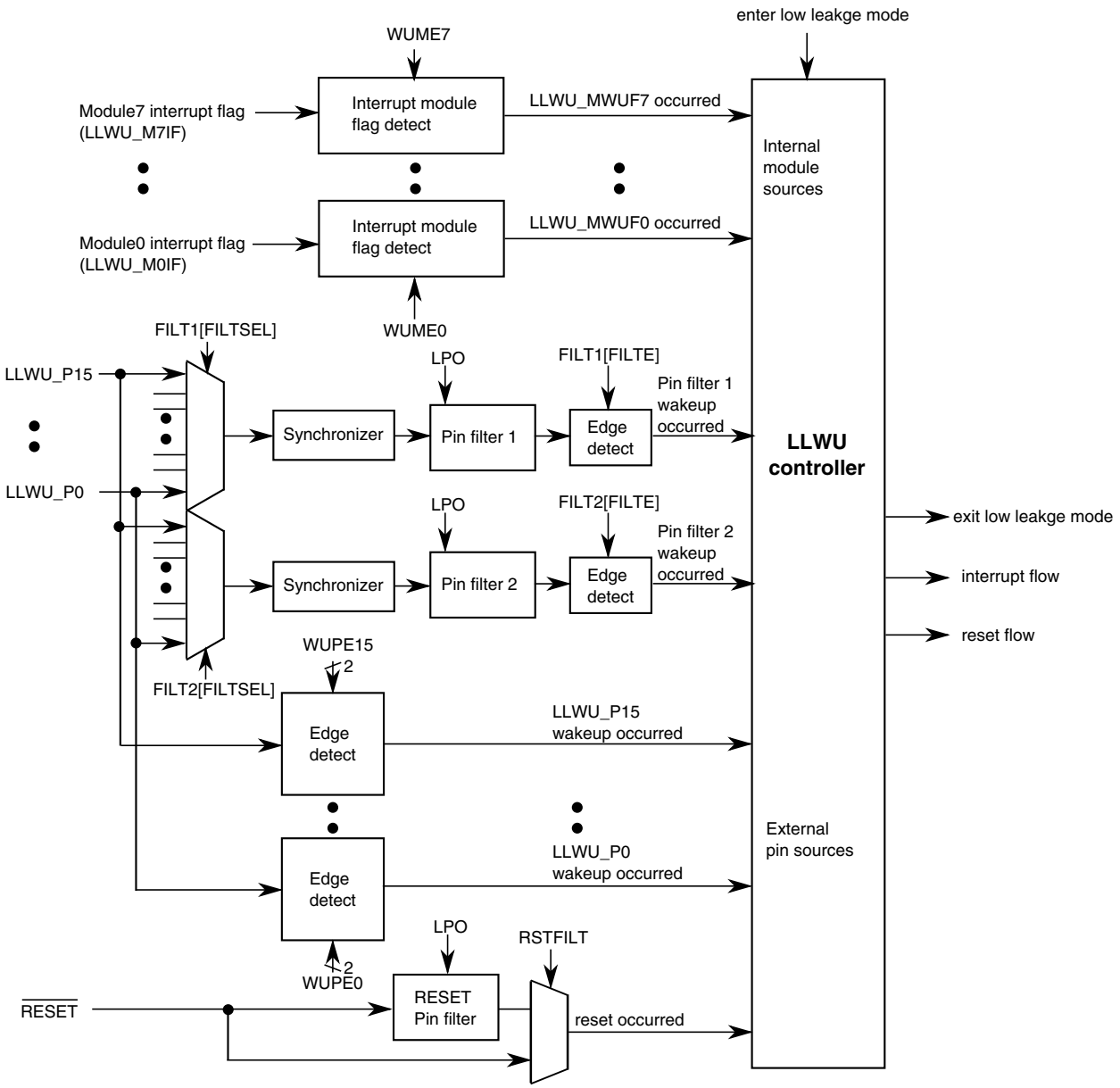



**Figure 16-1. LLWU block diagram**

## 16.2 LLWU signal descriptions

The signal properties of LLWU are shown in the following table. The external wakeup input pins can be enabled to detect either rising-edge, falling-edge, or on any change.

**Table 16-1. LLWU signal descriptions**

| Signal | Description | I/O |
|---|---|---|
| LLWU_Pn | Wakeup inputs (n = 0-15) | I |

## 16.3 Memory map/register definition

The LLWU includes the following registers:

- Wakeup source enable registers
  - Enable external pin input sources
  - Enable internal peripheral sources
- Wakeup flag registers
  - Indication of wakeup source that caused exit from a low-leakage power mode includes external pin or internal module interrupt
- Wakeup pin filter enable registers
- RESET pin filter enable register

**NOTE**

The LLWU registers can be written only in supervisor mode. Write accesses in user mode are blocked and will result in a bus error.

All LLWU registers are reset by Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. Each register's displayed reset value represents this subset of reset types. LLWU registers are unaffected by reset types that do not trigger Chip Reset not VLLS. For more information about the types of reset on this chip, refer to the Introduction details.

**LLWU memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4007_C000 | LLWU Pin Enable 1 register (LLWU_PE1) | 8 | R/W | 00h | 16.3.1/348 |
| 4007_C001 | LLWU Pin Enable 2 register (LLWU_PE2) | 8 | R/W | 00h | 16.3.2/349 |
| 4007_C002 | LLWU Pin Enable 3 register (LLWU_PE3) | 8 | R/W | 00h | 16.3.3/350 |
| 4007_C003 | LLWU Pin Enable 4 register (LLWU_PE4) | 8 | R/W | 00h | 16.3.4/351 |
| 4007_C004 | LLWU Module Enable register (LLWU_ME) | 8 | R/W | 00h | 16.3.5/352 |
| 4007_C005 | LLWU Flag 1 register (LLWU_F1) | 8 | R/W | 00h | 16.3.6/354 |
| 4007_C006 | LLWU Flag 2 register (LLWU_F2) | 8 | R/W | 00h | 16.3.7/355 |
| 4007_C007 | LLWU Flag 3 register (LLWU_F3) | 8 | R | 00h | 16.3.8/357 |
| 4007_C008 | LLWU Pin Filter 1 register (LLWU_FILT1) | 8 | R/W | 00h | 16.3.9/359 |
| 4007_C009 | LLWU Pin Filter 2 register (LLWU_FILT2) | 8 | R/W | 00h | 16.3.10/360 |
| 4007_C00A | LLWU Reset Enable register (LLWU_RST) | 8 | R/W | 02h | 16.3.11/361 |

### 16.3.1 LLWU Pin Enable 1 register (LLWU_PE1)

LLWU_PE1 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P3-LLWU_P0.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 0h offset = 4007_C000h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | WUPE3 | | WUPE2 | | WUPE1 | | WUPE0 | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**LLWU_PE1 field descriptions**

| Field | Description |
|---|---|
| 7–6<br>WUPE3 | Wakeup Pin Enable For LLWU_P3<br><br>Enables and configures the edge detection for the wakeup pin.<br><br>00 External input pin disabled as wakeup input<br>01 External input pin enabled with rising edge detection<br>10 External input pin enabled with falling edge detection<br>11 External input pin enabled with any change detection |
| 5–4<br>WUPE2 | Wakeup Pin Enable For LLWU_P2<br><br>Enables and configures the edge detection for the wakeup pin.<br><br>00 External input pin disabled as wakeup input<br>01 External input pin enabled with rising edge detection<br>10 External input pin enabled with falling edge detection<br>11 External input pin enabled with any change detection |
| 3–2<br>WUPE1 | Wakeup Pin Enable For LLWU_P1<br><br>Enables and configures the edge detection for the wakeup pin.<br><br>00 External input pin disabled as wakeup input<br>01 External input pin enabled with rising edge detection<br>10 External input pin enabled with falling edge detection<br>11 External input pin enabled with any change detection |
| 1–0<br>WUPE0 | Wakeup Pin Enable For LLWU_P0<br><br>Enables and configures the edge detection for the wakeup pin.<br><br>00 External input pin disabled as wakeup input<br>01 External input pin enabled with rising edge detection |

*Table continues on the next page...*

**LLWU_PE1 field descriptions (continued)**

| Field | Description |
|---|---|
| | 10 External input pin enabled with falling edge detection<br>11 External input pin enabled with any change detection |

## 16.3.2 LLWU Pin Enable 2 register (LLWU_PE2)

LLWU_PE2 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P7-LLWU_P4.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 1h offset = 4007_C001h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | WUPE7 | | WUPE6 | | WUPE5 | | WUPE4 | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**LLWU_PE2 field descriptions**

| Field | Description |
|---|---|
| 7–6<br>WUPE7 | Wakeup Pin Enable For LLWU_P7<br><br>Enables and configures the edge detection for the wakeup pin.<br><br>00 External input pin disabled as wakeup input<br>01 External input pin enabled with rising edge detection<br>10 External input pin enabled with falling edge detection<br>11 External input pin enabled with any change detection |
| 5–4<br>WUPE6 | Wakeup Pin Enable For LLWU_P6<br><br>Enables and configures the edge detection for the wakeup pin.<br><br>00 External input pin disabled as wakeup input<br>01 External input pin enabled with rising edge detection<br>10 External input pin enabled with falling edge detection<br>11 External input pin enabled with any change detection |
| 3–2<br>WUPE5 | Wakeup Pin Enable For LLWU_P5<br><br>Enables and configures the edge detection for the wakeup pin.<br><br>00 External input pin disabled as wakeup input<br>01 External input pin enabled with rising edge detection |

*Table continues on the next page...*

**LLWU_PE2 field descriptions (continued)**

| Field | Description |
|---|---|
| | 10 External input pin enabled with falling edge detection<br>11 External input pin enabled with any change detection |
| 1–0<br>WUPE4 | Wakeup Pin Enable For LLWU_P4<br><br>Enables and configures the edge detection for the wakeup pin.<br><br>00 External input pin disabled as wakeup input<br>01 External input pin enabled with rising edge detection<br>10 External input pin enabled with falling edge detection<br>11 External input pin enabled with any change detection |

## 16.3.3 LLWU Pin Enable 3 register (LLWU_PE3)

LLWU_PE3 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P11-LLWU_P8.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 2h offset = 4007_C002h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | WUPE11 | | WUPE10 | | WUPE9 | | WUPE8 | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**LLWU_PE3 field descriptions**

| Field | Description |
|---|---|
| 7–6<br>WUPE11 | Wakeup Pin Enable For LLWU_P11<br><br>Enables and configures the edge detection for the wakeup pin.<br><br>00 External input pin disabled as wakeup input<br>01 External input pin enabled with rising edge detection<br>10 External input pin enabled with falling edge detection<br>11 External input pin enabled with any change detection |
| 5–4<br>WUPE10 | Wakeup Pin Enable For LLWU_P10<br><br>Enables and configures the edge detection for the wakeup pin.<br><br>00 External input pin disabled as wakeup input<br>01 External input pin enabled with rising edge detection |

*Table continues on the next page...*

**LLWU_PE3 field descriptions (continued)**

| Field | Description |
|---|---|
| | 10 External input pin enabled with falling edge detection<br>11 External input pin enabled with any change detection |
| 3–2<br>WUPE9 | Wakeup Pin Enable For LLWU_P9<br><br>Enables and configures the edge detection for the wakeup pin.<br><br>00 External input pin disabled as wakeup input<br>01 External input pin enabled with rising edge detection<br>10 External input pin enabled with falling edge detection<br>11 External input pin enabled with any change detection |
| 1–0<br>WUPE8 | Wakeup Pin Enable For LLWU_P8<br><br>Enables and configures the edge detection for the wakeup pin.<br><br>00 External input pin disabled as wakeup input<br>01 External input pin enabled with rising edge detection<br>10 External input pin enabled with falling edge detection<br>11 External input pin enabled with any change detection |

## 16.3.4 LLWU Pin Enable 4 register (LLWU_PE4)

LLWU_PE4 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P15-LLWU_P12.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 3h offset = 4007_C003h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | WUPE15 | | WUPE14 | | WUPE13 | | WUPE12 | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**LLWU_PE4 field descriptions**

| Field | Description |
|---|---|
| 7–6<br>WUPE15 | Wakeup Pin Enable For LLWU_P15<br><br>Enables and configures the edge detection for the wakeup pin.<br><br>00 External input pin disabled as wakeup input<br>01 External input pin enabled with rising edge detection |

*Table continues on the next page...*

**LLWU_PE4 field descriptions (continued)**

| Field | Description |
|---|---|
| | 10 External input pin enabled with falling edge detection<br>11 External input pin enabled with any change detection |
| 5–4<br>WUPE14 | Wakeup Pin Enable For LLWU_P14<br><br>Enables and configures the edge detection for the wakeup pin.<br><br>00 External input pin disabled as wakeup input<br>01 External input pin enabled with rising edge detection<br>10 External input pin enabled with falling edge detection<br>11 External input pin enabled with any change detection |
| 3–2<br>WUPE13 | Wakeup Pin Enable For LLWU_P13<br><br>Enables and configures the edge detection for the wakeup pin.<br><br>00 External input pin disabled as wakeup input<br>01 External input pin enabled with rising edge detection<br>10 External input pin enabled with falling edge detection<br>11 External input pin enabled with any change detection |
| 1–0<br>WUPE12 | Wakeup Pin Enable For LLWU_P12<br><br>Enables and configures the edge detection for the wakeup pin.<br><br>00 External input pin disabled as wakeup input<br>01 External input pin enabled with rising edge detection<br>10 External input pin enabled with falling edge detection<br>11 External input pin enabled with any change detection |

## 16.3.5 LLWU Module Enable register (LLWU_ME)

LLWU_ME contains the bits to enable the internal module flag as a wakeup input source for inputs MWUF7-MWUF0.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 4h offset = 4007_C004h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | WUME7 | WUME6 | WUME5 | WUME4 | WUME3 | WUME2 | WUME1 | WUME0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## LLWU_ME field descriptions

| Field | Description |
|---|---|
| 7<br>WUME7 | Wakeup Module Enable For Module 7<br><br>Enables an internal module as a wakeup source input.<br><br>0 Internal module flag not used as wakeup source<br>1 Internal module flag used as wakeup source |
| 6<br>WUME6 | Wakeup Module Enable For Module 6<br><br>Enables an internal module as a wakeup source input.<br><br>0 Internal module flag not used as wakeup source<br>1 Internal module flag used as wakeup source |
| 5<br>WUME5 | Wakeup Module Enable For Module 5<br><br>Enables an internal module as a wakeup source input.<br><br>0 Internal module flag not used as wakeup source<br>1 Internal module flag used as wakeup source |
| 4<br>WUME4 | Wakeup Module Enable For Module 4<br><br>Enables an internal module as a wakeup source input.<br><br>0 Internal module flag not used as wakeup source<br>1 Internal module flag used as wakeup source |
| 3<br>WUME3 | Wakeup Module Enable For Module 3<br><br>Enables an internal module as a wakeup source input.<br><br>0 Internal module flag not used as wakeup source<br>1 Internal module flag used as wakeup source |
| 2<br>WUME2 | Wakeup Module Enable For Module 2<br><br>Enables an internal module as a wakeup source input.<br><br>0 Internal module flag not used as wakeup source<br>1 Internal module flag used as wakeup source |
| 1<br>WUME1 | Wakeup Module Enable for Module 1<br><br>Enables an internal module as a wakeup source input.<br><br>0 Internal module flag not used as wakeup source<br>1 Internal module flag used as wakeup source |
| 0<br>WUME0 | Wakeup Module Enable For Module 0<br><br>Enables an internal module as a wakeup source input.<br><br>0 Internal module flag not used as wakeup source<br>1 Internal module flag used as wakeup source |

### 16.3.6 LLWU Flag 1 register (LLWU_F1)

LLWU_F1 contains the wakeup flags indicating which wakeup source caused the MCU to exit LLS or VLLS mode. For LLS, this is the source causing the CPU interrupt flow. For VLLS, this is the source causing the MCU reset flow.

The external wakeup flags are read-only and clearing a flag is accomplished by a write of a 1 to the corresponding WUFx bit. The wakeup flag (WUFx), if set, will remain set if the associated WUPEx bit is cleared.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 5h offset = 4007_C005h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | WUF7 | WUF6 | WUF5 | WUF4 | WUF3 | WUF2 | WUF1 | WUF0 |
| Write | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**LLWU_F1 field descriptions**

| Field | Description |
|---|---|
| 7<br>WUF7 | Wakeup Flag For LLWU_P7<br><br>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF7.<br><br>0 LLWU_P7 input was not a wakeup source<br>1 LLWU_P7 input was a wakeup source |
| 6<br>WUF6 | Wakeup Flag For LLWU_P6<br><br>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF6.<br><br>0 LLWU_P6 input was not a wakeup source<br>1 LLWU_P6 input was a wakeup source |
| 5<br>WUF5 | Wakeup Flag For LLWU_P5<br><br>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF5.<br><br>0 LLWU_P5 input was not a wakeup source<br>1 LLWU_P5 input was a wakeup source |

*Table continues on the next page...*

**LLWU_F1 field descriptions (continued)**

| Field | Description |
|---|---|
| 4<br>WUF4 | Wakeup Flag For LLWU_P4<br><br>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF4.<br><br>0 LLWU_P4 input was not a wakeup source<br>1 LLWU_P4 input was a wakeup source |
| 3<br>WUF3 | Wakeup Flag For LLWU_P3<br><br>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF3.<br><br>0 LLWU_P3 input was not a wakeup source<br>1 LLWU_P3 input was a wakeup source |
| 2<br>WUF2 | Wakeup Flag For LLWU_P2<br><br>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF2.<br><br>0 LLWU_P2 input was not a wakeup source<br>1 LLWU_P2 input was a wakeup source |
| 1<br>WUF1 | Wakeup Flag For LLWU_P1<br><br>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF1.<br><br>0 LLWU_P1 input was not a wakeup source<br>1 LLWU_P1 input was a wakeup source |
| 0<br>WUF0 | Wakeup Flag For LLWU_P0<br><br>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF0.<br><br>0 LLWU_P0 input was not a wakeup source<br>1 LLWU_P0 input was a wakeup source |

## 16.3.7 LLWU Flag 2 register (LLWU_F2)

LLWU_F2 contains the wakeup flags indicating which wakeup source caused the MCU to exit LLS or VLLS mode. For LLS, this is the source causing the CPU interrupt flow. For VLLS, this is the source causing the MCU reset flow.

The external wakeup flags are read-only and clearing a flag is accomplished by a write of a 1 to the corresponding WUFx bit. The wakeup flag (WUFx), if set, will remain set if the associated WUPEx bit is cleared.

## NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 6h offset = 4007_C006h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | WUF15 | WUF14 | WUF13 | WUF12 | WUF11 | WUF10 | WUF9 | WUF8 |
| Write | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### LLWU_F2 field descriptions

| Field | Description |
|---|---|
| 7<br>WUF15 | Wakeup Flag For LLWU_P15<br><br>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF15.<br><br>0 LLWU_P15 input was not a wakeup source<br>1 LLWU_P15 input was a wakeup source |
| 6<br>WUF14 | Wakeup Flag For LLWU_P14<br><br>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF14.<br><br>0 LLWU_P14 input was not a wakeup source<br>1 LLWU_P14 input was a wakeup source |
| 5<br>WUF13 | Wakeup Flag For LLWU_P13<br><br>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF13.<br><br>0 LLWU_P13 input was not a wakeup source<br>1 LLWU_P13 input was a wakeup source |
| 4<br>WUF12 | Wakeup Flag For LLWU_P12<br><br>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF12.<br><br>0 LLWU_P12 input was not a wakeup source<br>1 LLWU_P12 input was a wakeup source |
| 3<br>WUF11 | Wakeup Flag For LLWU_P11<br><br>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF11.<br><br>0 LLWU_P11 input was not a wakeup source<br>1 LLWU_P11 input was a wakeup source |
| 2<br>WUF10 | Wakeup Flag For LLWU_P10 |

*Table continues on the next page...*

**LLWU_F2 field descriptions (continued)**

| Field | Description |
|---|---|
| | Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF10.<br><br>0 LLWU_P10 input was not a wakeup source<br>1 LLWU_P10 input was a wakeup source |
| 1<br>WUF9 | Wakeup Flag For LLWU_P9<br><br>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF9.<br><br>0 LLWU_P9 input was not a wakeup source<br>1 LLWU_P9 input was a wakeup source |
| 0<br>WUF8 | Wakeup Flag For LLWU_P8<br><br>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF8.<br><br>0 LLWU_P8 input was not a wakeup source<br>1 LLWU_P8 input was a wakeup source |

## 16.3.8 LLWU Flag 3 register (LLWU_F3)

LLWU_F3 contains the wakeup flags indicating which internal wakeup source caused the MCU to exit LLS or VLLS mode. For LLS, this is the source causing the CPU interrupt flow. For VLLS, this is the source causing the MCU reset flow.

For internal peripherals that are capable of running in a low-leakage power mode, such as a real time clock module or CMP module, the flag from the associated peripheral is accessible as the MWUFx bit. The flag will need to be cleared in the peripheral instead of writing a 1 to the MWUFx bit.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 7h offset = 4007_C007h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | MWUF7 | MWUF6 | MWUF5 | MWUF4 | MWUF3 | MWUF2 | MWUF1 | MWUF0 |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## LLWU_F3 field descriptions

| Field | Description |
|---|---|
| 7<br>MWUF7 | Wakeup flag For module 7<br><br>Indicates that an enabled internal peripheral was a source of exiting a low-leakage power mode. To clear the flag, follow the internal peripheral flag clearing mechanism.<br><br>0 Module 7 input was not a wakeup source<br>1 Module 7 input was a wakeup source |
| 6<br>MWUF6 | Wakeup flag For module 6<br><br>Indicates that an enabled internal peripheral was a source of exiting a low-leakage power mode. To clear the flag, follow the internal peripheral flag clearing mechanism.<br><br>0 Module 6 input was not a wakeup source<br>1 Module 6 input was a wakeup source |
| 5<br>MWUF5 | Wakeup flag For module 5<br><br>Indicates that an enabled internal peripheral was a source of exiting a low-leakage power mode. To clear the flag, follow the internal peripheral flag clearing mechanism.<br><br>0 Module 5 input was not a wakeup source<br>1 Module 5 input was a wakeup source |
| 4<br>MWUF4 | Wakeup flag For module 4<br><br>Indicates that an enabled internal peripheral was a source of exiting a low-leakage power mode. To clear the flag, follow the internal peripheral flag clearing mechanism.<br><br>0 Module 4 input was not a wakeup source<br>1 Module 4 input was a wakeup source |
| 3<br>MWUF3 | Wakeup flag For module 3<br><br>Indicates that an enabled internal peripheral was a source of exiting a low-leakage power mode. To clear the flag, follow the internal peripheral flag clearing mechanism.<br><br>0 Module 3 input was not a wakeup source<br>1 Module 3 input was a wakeup source |
| 2<br>MWUF2 | Wakeup flag For module 2<br><br>Indicates that an enabled internal peripheral was a source of exiting a low-leakage power mode. To clear the flag, follow the internal peripheral flag clearing mechanism.<br><br>0 Module 2 input was not a wakeup source<br>1 Module 2 input was a wakeup source |
| 1<br>MWUF1 | Wakeup flag For module 1<br><br>Indicates that an enabled internal peripheral was a source of exiting a low-leakage power mode. To clear the flag, follow the internal peripheral flag clearing mechanism.<br><br>0 Module 1 input was not a wakeup source<br>1 Module 1 input was a wakeup source |
| 0<br>MWUF0 | Wakeup flag For module 0 |

*Table continues on the next page...*

**LLWU_F3 field descriptions (continued)**

| Field | Description |
|---|---|
| | Indicates that an enabled internal peripheral was a source of exiting a low-leakage power mode. To clear the flag, follow the internal peripheral flag clearing mechanism.<br><br>0 Module 0 input was not a wakeup source<br>1 Module 0 input was a wakeup source |

## 16.3.9 LLWU Pin Filter 1 register (LLWU_FILT1)

LLWU_FILT1 is a control and status register that is used to enable/disable the digital filter 1 features for an external pin.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 8h offset = 4007_C008h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | FILTF | FILTE | | 0 | FILTSEL | | | |
| Write | w1c | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**LLWU_FILT1 field descriptions**

| Field | Description |
|---|---|
| 7<br>FILTF | Filter Detect Flag<br><br>Indicates that the filtered external wakeup pin, selected by FILTSEL, was a source of exiting a low-leakage power mode. To clear the flag write a one to FILTF.<br><br>0 Pin Filter 1 was not a wakeup source<br>1 Pin Filter 1 was a wakeup source |
| 6–5<br>FILTE | Digital Filter On External Pin<br><br>Controls the digital filter options for the external pin detect.<br><br>00 Filter disabled<br>01 Filter posedge detect enabled<br>10 Filter negedge detect enabled<br>11 Filter any edge detect enabled |
| 4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

*Table continues on the next page...*

**LLWU_FILT1 field descriptions (continued)**

| Field | Description |
|---|---|
| 3–0<br>FILTSEL | Filter Pin Select<br><br>Selects 1 out of the 16 wakeup pins to be muxed into the filter.<br><br>0000 Select LLWU_P0 for filter<br>... ...<br>1111 Select LLWU_P15 for filter |

## 16.3.10 LLWU Pin Filter 2 register (LLWU_FILT2)

LLWU_FILT2 is a control and status register that is used to enable/disable the digital filter 2 features for an external pin.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 9h offset = 4007_C009h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | FILTF | FILTE | | 0 | FILTSEL | | | |
| Write | w1c | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**LLWU_FILT2 field descriptions**

| Field | Description |
|---|---|
| 7<br>FILTF | Filter Detect Flag<br><br>Indicates that the filtered external wakeup pin, selected by FILTSEL, was a source of exiting a low-leakage power mode. To clear the flag write a one to FILTF.<br><br>0 Pin Filter 2 was not a wakeup source<br>1 Pin Filter 2 was a wakeup source |
| 6–5<br>FILTE | Digital Filter On External Pin<br><br>Controls the digital filter options for the external pin detect.<br><br>00 Filter disabled<br>01 Filter posedge detect enabled<br>10 Filter negedge detect enabled<br>11 Filter any edge detect enabled |
| 4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

*Table continues on the next page...*

**LLWU_FILT2 field descriptions (continued)**

| Field | Description |
|---|---|
| 3–0<br>FILTSEL | Filter Pin Select<br><br>Selects 1 out of the 16 wakeup pins to be muxed into the filter.<br><br>0000 Select LLWU_P0 for filter<br>... ...<br>1111 Select LLWU_P15 for filter |

## 16.3.11 LLWU Reset Enable register (LLWU_RST)

LLWU_RST is a control register that is used to enable/disable the digital filter for the external pin detect and RESET pin.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + Ah offset = 4007_C00Ah

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | | | | | | LLRSTE | RSTFILT |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

**LLWU_RST field descriptions**

| Field | Description |
|---|---|
| 7–2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 1<br>LLRSTE | Low-Leakage Mode RESET Enable<br><br>This bit must be set to allow the device to be reset while in a low-leakage power mode. On devices where Reset is not a dedicated pin, the RESET pin must also be enabled in the explicit port mux control.<br><br>0 RESET pin not enabled as a leakage mode exit source<br>1 RESET pin enabled as a low leakage mode exit source |
| 0<br>RSTFILT | Digital Filter On RESET Pin<br><br>Enables the digital filter for the RESET pin during LLS, VLLS3, VLLS2, or VLLS1 modes.<br><br>0 Filter not enabled<br>1 Filter enabled |

## 16.4 Functional description

This on-chip peripheral module is called a low-leakage wakeup unit (LLWU) module because it allows internal peripherals and external input pins as a source of wakeup from low-leakage modes. It is operational only in LLS and VLLSx modes.

The LLWU module contains pin enables for each external pin and internal module. For each external pin, the user can disable or select the edge type for the wakeup. Type options are:

- Falling-edge
- Rising-edge
- Either-edge

When an external pin is enabled as a wakeup source, the pin must be configured as an input pin.

The LLWU implements optional 3-cycle glitch filters, based on the LPO clock. A detected external pin, either wakeup or $\overline{\text{RESET}}$, is required to remain asserted until the enabled glitch filter times out. Additional latency of up to 2 cycles is due to synchronization, which results in a total of up to 5 cycles of delay before the detect circuit alerts the system to the wakeup or reset event when the filter function is enabled. Two wakeup detect filters are available for selected external pins. A separate reset filter is on the $\overline{\text{RESET}}$ pin. Glitch filtering is not provided on the internal modules.

For internal module wakeup operation, the WUMEx bit enables the associated module as a wakeup source.

### 16.4.1 LLS mode

Wakeup events triggered from either an external pin input or an internal module input result in a CPU interrupt flow to begin user code execution.

An LLS reset event due to RESET pin assertion causes an exit via a system reset. State retention data is lost, and the I/O states return to their reset state. The RCM_SRS[WAKEUP] and RCM_SRS[PIN] bits are set and the system executes a reset flow before CPU operation begins with a reset vector fetch.

### 16.4.2 VLLS modes

In the case of a wakeup due to external pin or internal module wakeup, recovery is always via a reset flow and the RCM_SRS[WAKEUP] is set indicating the low-leakage mode was active. State retention data is lost and I/O will be restored after PMC_REGSC[ACKISO] has been written.

A VLLS exit event due to $\overline{\text{RESET}}$ pin assertion causes an exit via a system reset. State retention data is lost and the I/O states immediately return to their reset state. The RCM_SRS[WAKEUP] and RCM_SRS[PIN] bits are set and the system executes a reset flow before CPU operation begins with a reset vector fetch.

### 16.4.3 Initialization

For an enabled peripheral wakeup input, the peripheral flag must be cleared by software before entering LLS or VLLSx mode to avoid an immediate exit from the mode.

Flags associated with external input pins, filtered and unfiltered, must also be cleared by software prior to entry to LLS or VLLSx mode.

After enabling an external pin filter or changing the source pin, wait at least 5 LPO clock cycles before entering LLS or VLLSx mode to allow the filter to initialize.

**NOTE**

After recovering from a VLLS mode, user must restore chip configuration before clearing ACKISO. In particular, pin configuration for enabled LLWU wakeup pins must be restored to avoid any LLWU flag from being falsely set when ACKISO is cleared.

The signal selected as a wakeup source pin must be a digital pin, as selected in the pin mux control.

# Chapter 17
# Miscellaneous Control Module (MCM)

## 17.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The Miscellaneous Control Module (MCM) provides a myriad of miscellaneous control functions.

### 17.1.1 Features

The MCM includes the following features:

- Program-visible information on the platform configuration and revision
- Control and counting logic for embedded trace buffer (ETB) almost full

## 17.2 Memory map/register descriptions

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

**MCM memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| E008_0008 | Crossbar Switch (AXBS) Slave Configuration (MCM_PLASC) | 16 | R | 001Fh | 17.2.1/366 |
| E008_000A | Crossbar Switch (AXBS) Master Configuration (MCM_PLAMC) | 16 | R | 0037h | 17.2.2/367 |

*Table continues on the next page...*

**MCM memory map (continued)**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| E008_000C | Control Register (MCM_CR) | 32 | R/W | 0000_0000h | 17.2.3/368 |
| E008_0010 | Interrupt Status Register (MCM_ISCR) | 32 | R | 0000_0000h | 17.2.4/370 |
| E008_0014 | ETB Counter Control register (MCM_ETBCC) | 32 | R/W | 0000_0000h | 17.2.5/373 |
| E008_0018 | ETB Reload register (MCM_ETBRL) | 32 | R/W | 0000_0000h | 17.2.6/374 |
| E008_001C | ETB Counter Value register (MCM_ETBCNT) | 32 | R | 0000_0000h | 17.2.7/374 |
| E008_0030 | Process ID register (MCM_PID) | 32 | R/W | 0000_0000h | 17.2.8/375 |

## 17.2.1 Crossbar Switch (AXBS) Slave Configuration (MCM_PLASC)

PLASC is a 16-bit read-only register identifying the presence/absence of bus slave connections to the device's crossbar switch.

Address: E008_0000h base + 8h offset = E008_0008h

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read | 0 | | | | | | | | ASC | | | | | | | |
| Write | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

**MCM_PLASC field descriptions**

| Field | Description |
|---|---|
| 15–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7–0<br>ASC | Each bit in the ASC field indicates whether there is a corresponding connection to the crossbar switch's slave input port.<br><br>0 A bus slave connection to AXBS input port *n* is absent<br>1 A bus slave connection to AXBS input port *n* is present |

## 17.2.2 Crossbar Switch (AXBS) Master Configuration (MCM_PLAMC)

PLAMC is a 16-bit read-only register identifying the presence/absence of bus master connections to the device's crossbar switch.

Address: E008_0000h base + Ah offset = E008_000Ah

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read | 0 | | | | | | | | AMC | | | | | | | |
| Write | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |

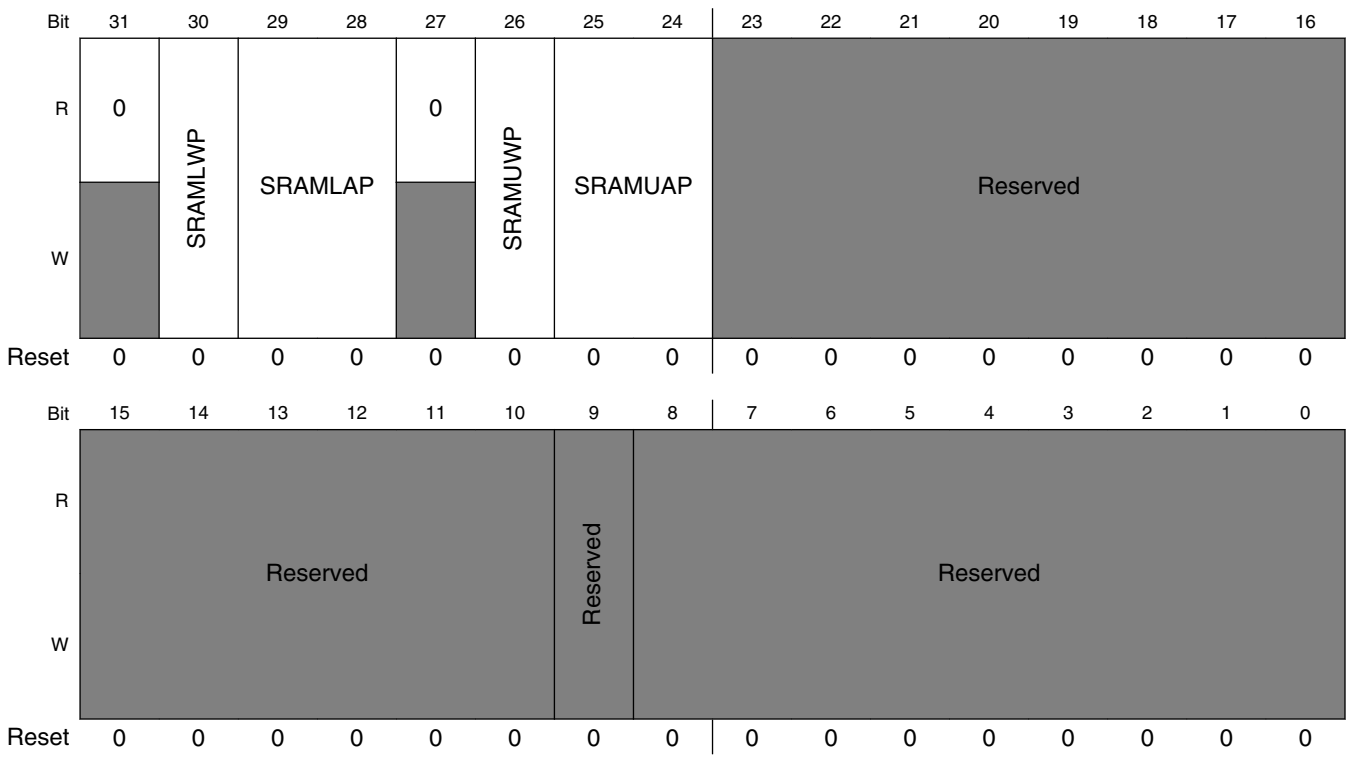
**MCM_PLAMC field descriptions**

| Field | Description |
|---|---|
| 15–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7–0<br>AMC | Each bit in the AMC field indicates whether there is a corresponding connection to the AXBS master input port.<br><br>0 A bus master connection to AXBS input port *n* is absent<br>1 A bus master connection to AXBS input port *n* is present |

## 17.2.3 Control Register (MCM_CR)

CR defines the arbitration and protection schemes for the two system RAM arrays.

Address: E008_0000h base + Ch offset = E008_000Ch




**MCM_CR field descriptions**

| Field | Description |
|---|---|
| 31<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 30<br>SRAMLWP | SRAM_L Write Protect<br><br>When this bit is set, writes to SRAM_L array generates a bus error. |
| 29–28<br>SRAMLAP | SRAM_L arbitration priority<br><br>Defines the arbitration scheme and priority for the processor and SRAM backdoor accesses to the SRAM_L array.<br><br>00 Round robin<br>01 Special round robin (favors SRAM backoor accesses over the processor)<br>10 Fixed priority. Processor has highest, backdoor has lowest<br>11 Fixed priority. Backdoor has highest, processor has lowest |
| 27<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

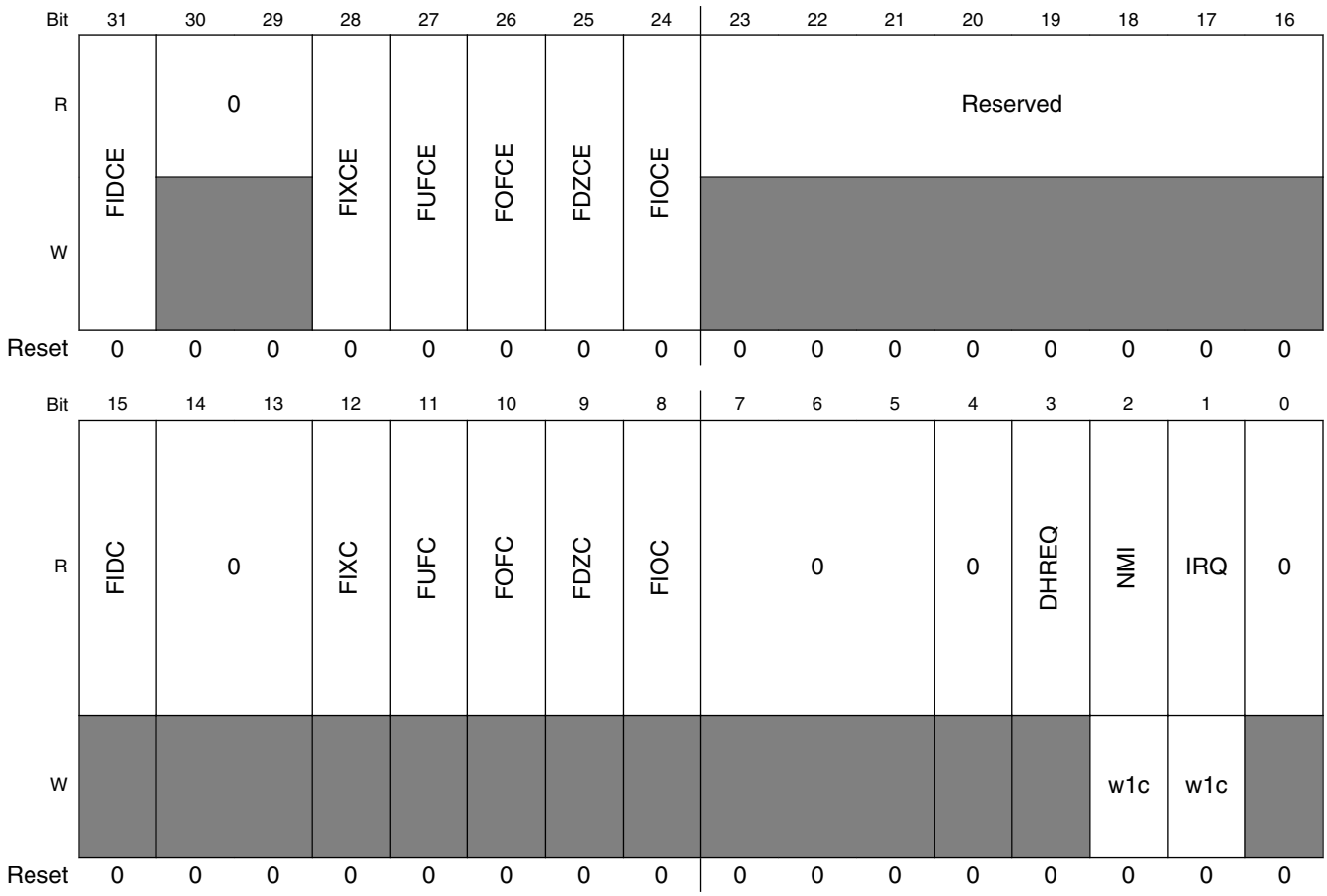
*Table continues on the next page...*

## MCM_CR field descriptions (continued)

| Field | Description |
|---|---|
| 26<br>SRAMUWP | SRAM_U write protect<br><br>When this bit is set, writes to SRAM_U array generates a bus error. |
| 25–24<br>SRAMUAP | SRAM_U arbitration priority<br><br>Defines the arbitration scheme and priority for the processor and SRAM backdoor accesses to the SRAM_U array.<br><br>00 Round robin<br>01 Special round robin (favors SRAM backoor accesses over the processor)<br>10 Fixed priority. Processor has highest, backdoor has lowest<br>11 Fixed priority. Backdoor has highest, processor has lowest |
| 23–10<br>Reserved | This field is reserved. |
| 9<br>Reserved | This field is reserved. |
| 8–0<br>Reserved | This field is reserved. |

## 17.2.4 Interrupt Status Register (MCM_ISCR)

Address: E008_0000h base + 10h offset = E008_0010h




### MCM_ISCR field descriptions

| Field | Description |
|---|---|
| 31<br>FIDCE | FPU input denormal interrupt enable<br><br>0 Disable interrupt<br>1 Enable interrupt |
| 30–29<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 28<br>FIXCE | FPU inexact interrupt enable<br><br>0 Disable interrupt<br>1 Enable interrupt |
| 27<br>FUFCE | FPU underflow interrupt enable<br><br>0 Disable interrupt<br>1 Enable interrupt |

*Table continues on the next page...*

## MCM_ISCR field descriptions (continued)

| Field | Description |
|---|---|
| 26<br>FOFCE | FPU overflow interrupt enable<br><br>0 Disable interrupt<br>1 Enable interrupt |
| 25<br>FDZCE | FPU divide-by-zero interrupt enable<br><br>0 Disable interrupt<br>1 Enable interrupt |
| 24<br>FIOCE | FPU invalid operation interrupt enable<br><br>0 Disable interrupt<br>1 Enable interrupt |
| 23–16<br>Reserved | This field is reserved. |
| 15<br>FIDC | FPU input denormal interrupt status<br><br>This read-only bit is a copy of the core's FPSCR[IDC] bit and signals input denormalized number has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[IDC] bit.<br><br>0 No interrupt<br>1 Interrupt occurred |
| 14–13<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 12<br>FIXC | FPU inexact interrupt status<br><br>This read-only bit is a copy of the core's FPSCR[IXC] bit and signals an inexact number has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[IXC] bit.<br><br>0 No interrupt<br>1 Interrupt occurred |
| 11<br>FUFC | FPU underflow interrupt status<br><br>This read-only bit is a copy of the core's FPSCR[UFC] bit and signals an underflow has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[UFC] bit.<br><br>0 No interrupt<br>1 Interrupt occurred |
| 10<br>FOFC | FPU overflow interrupt status<br><br>This read-only bit is a copy of the core's FPSCR[OFC] bit and signals an overflow has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[OFC] bit.<br><br>0 No interrupt<br>1 Interrupt occurred |
| 9<br>FDZC | FPU divide-by-zero interrupt status<br><br>This read-only bit is a copy of the core's FPSCR[DZC] bit and signals a divide by zero has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[DZC] bit.<br><br>0 No interrupt<br>1 Interrupt occurred |

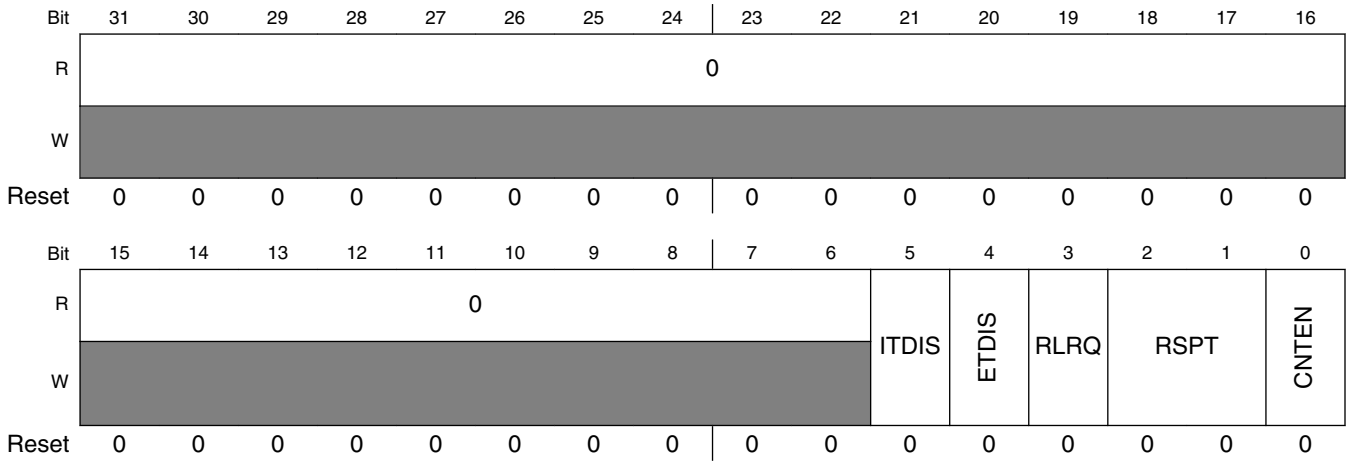
*Table continues on the next page...*

## MCM_ISCR field descriptions (continued)

| Field | Description |
|---|---|
| 8<br>FIOC | FPU invalid operation interrupt status<br><br>This read-only bit is a copy of the core's FPSCR[IOC] bit and signals an illegal operation has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[IOC] bit.<br><br>0 No interrupt<br>1 Interrupt occurred |
| 7–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3<br>DHREQ | Debug Halt Request Indicator<br><br>Indicates that a debug halt request is initiated due to a ETB counter expiration, ETBCC[2:0] = 3b111 & ETBCV[10:0] = 11h0. This bit is cleared when the counter is disabled or when the ETB counter is reloaded.<br><br>0 No debug halt request<br>1 Debug halt request initiated |
| 2<br>NMI | Non-maskable Interrupt Pending<br><br>If ETBCC[RSPT] is set to 10b, this bit is set when the ETB counter expires.<br><br>0 No pending NMI<br>1 Due to the ETB counter expiring, an NMI is pending |
| 1<br>IRQ | Normal Interrupt Pending<br><br>If ETBCC[RSPT] is set to 01b, this bit is set when the ETB counter expires.<br><br>0 No pending interrupt<br>1 Due to the ETB counter expiring, a normal interrupt is pending |
| 0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 17.2.5 ETB Counter Control register (MCM_ETBCC)

Address: E008_0000h base + 14h offset = E008_0014h




### MCM_ETBCC field descriptions

| Field | Description |
|---|---|
| 31–6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5<br>ITDIS | ITM-To-TPIU Disable<br><br>Disables the trace path from ITM to TPIU.<br><br>0 ITM-to-TPIU trace path enabled<br>1 ITM-to-TPIU trace path disabled |
| 4<br>ETDIS | ETM-To-TPIU Disable<br><br>Disables the trace path from ETM to TPIU.<br><br>0 ETM-to-TPIU trace path enabled<br>1 ETM-to-TPIU trace path disabled |
| 3<br>RLRQ | Reload Request<br><br>Reloads the ETB packet counter with the MCM_ETBRL RELOAD value.<br><br>If IRQ or NMI interrupts were enabled and an NMI or IRQ interrupt was generated on counter expiration, setting this bit clears the pending NMI or IRQ interrupt request.<br><br>If debug halt was enabled and a debug halt request was asserted on counter expiration, setting this bit clears the debug halt request.<br><br>0 No effect<br>1 Clears pending debug halt, NMI, or IRQ interrupt requests |
| 2–1<br>RSPT | Response Type<br><br>00 No response when the ETB count expires<br>01 Generate a normal interrupt when the ETB count expires |

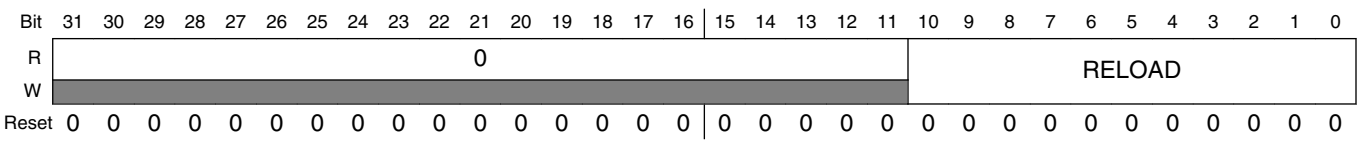
*Table continues on the next page...*

**MCM_ETBCC field descriptions (continued)**

| Field | Description |
|---|---|
| | 10 Generate an NMI when the ETB count expires<br>11 Generate a debug halt when the ETB count expires |
| 0<br>CNTEN | Counter Enable<br><br>Enables the ETB counter.<br><br>0 ETB counter disabled<br>1 ETB counter enabled |

## 17.2.6 ETB Reload register (MCM_ETBRL)

Address: E008_0000h base + 18h offset = E008_0018h

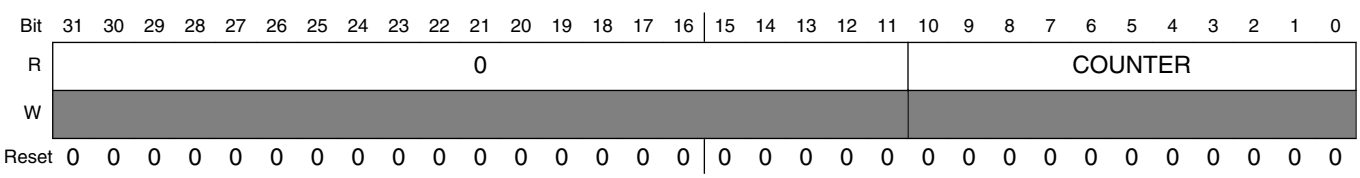



**MCM_ETBRL field descriptions**

| Field | Description |
|---|---|
| 31–11<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 10–0<br>RELOAD | Byte Count Reload Value<br><br>Indicates the 0-mod-4 value the counter reloads to. Writing a non-0-mod-4 value to this field results in a bus error. |

## 17.2.7 ETB Counter Value register (MCM_ETBCNT)

Address: E008_0000h base + 1Ch offset = E008_001Ch



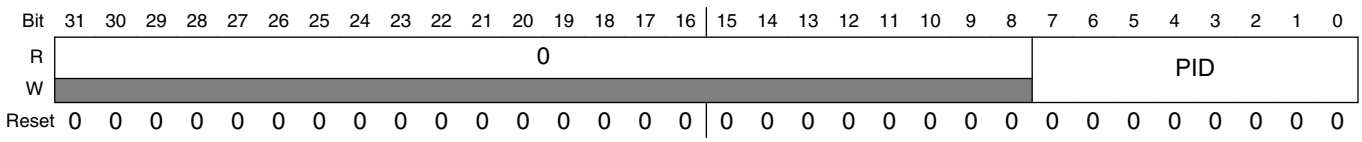

**MCM_ETBCNT field descriptions**

| Field | Description |
|---|---|
| 31–11<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 10–0<br>COUNTER | Byte Count Counter Value<br><br>Indicates the current 0-mod-4 value of the counter. |

### 17.2.8 Process ID register (MCM_PID)

This register drives the M0_PID and M1_PID values in the Memory Protection Unit(MPU). System software loads this register before passing control to a given user mode process. If the PID of the process does not match the value in this register, a bus error occurs. See the MPU chapter for more details.

Address: E008_0000h base + 30h offset = E008_0030h




**MCM_PID field descriptions**

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7–0<br>PID | M0_PID And M1_PID For MPU<br><br>Drives the M0_PID and M1_PID values in the MPU. |

## 17.3 Functional description

This section describes the functional description of MCM module.

### 17.3.1 Interrupts

The MCM generates two interrupt requests:

- Non-maskable interrupt
- Normal interrupt

#### 17.3.1.1 Non-maskable interrupt

The MCM's NMI is generated if:

- ISCR[ETBN] is set, when
  - The ETB counter is enabled, ETBCC[CNTEN] = 1
  - The ETB count expires
  - The response to counter expiration is an NMI, MCM_ETBCC[RSPT] = 10

#### 17.3.1.2 Normal interrupt

The MCM's normal interrupt is generated if any of the following is true:

- ISCR[ETBI] is set, when
  - The ETB counter is enabled, ETBCC[CNTEN] = 1
  - The ETB count expires
  - The response to counter expiration is a normal interrupt, ETBCC[RSPT] = 01
- FPU input denormal interrupt is enabled (FIDCE) and an input is denormalized (FIDC)
- FPU inexact interrupt is enabled (FIXCE) and a number is inexact (FIXC)
- FPU underflow interrupt is enabled (FUFCE) and an underflow occurs (FUFC)
- FPU overflow interrupt is enabled (FOFCE) and an overflow occurs (FOFC)
- FPU divide-by-zero interrupt is enabled (FDZCE) and a divide-by-zero occurs (FDZC)
- FPU invalid operation interrupt is enabled (FDZCE) and an invalid occurs (FDZC)

#### 17.3.1.3 Determining source of the normal interrupt

To determine the exact source of the normal interrupt qualify the interrupt status flags with the corresponding interrupt enable bits.

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

# Chapter 18
# Crossbar Switch (AXBS)

## 18.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

This chapter provides information on the layout, configuration, and programming of the crossbar switch. The crossbar switch connects bus masters and bus slaves using a crossbar switch structure. This structure allows all bus masters to access different bus slaves simultaneously, while providing arbitration among the bus masters when they access the same slave. A variety of bus arbitration methods and attributes may be programmed on a slave-by-slave basis.

### 18.1.1 Features

The crossbar switch includes these features:

- Symmetric crossbar bus switch implementation
  - Allows concurrent accesses from different masters to different slaves
  - Slave arbitration attributes configured on a slave-by-slave basis
- 32-bit data bus
- Support for byte, 2-byte, 4-byte, and 16-byte burst transfers
- Operation at a 1-to-1 clock frequency with the bus masters
- Low-Power Park mode support

## 18.2 Memory Map / Register Definition

Each slave port of the crossbar switch contains configuration registers. Read- and write-transfers require two bus clock cycles. The registers can be read from and written to only in supervisor mode. Additionally, these registers can be read from or written to only by 32-bit accesses.

A bus error response is returned if an unimplemented location is accessed within the crossbar switch.

The slave registers also feature a bit that, when set, prevents the registers from being written. The registers remain readable, but future write attempts have no effect on the registers and are terminated with a bus error response to the master initiating the write. The core, for example, takes a bus error interrupt.

**NOTE**

This section shows the registers for all eight master and slave ports. If a master or slave is not used on this particular device, then unexpected results occur when writing to its registers. See the chip configuration details for the exact master/slave assignments for your device.

**AXBS memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4000_4000 | Priority Registers Slave (AXBS_PRS0) | 32 | R/W | See section | 18.2.1/379 |
| 4000_4010 | Control Register (AXBS_CRS0) | 32 | R/W | 0000_0000h | 18.2.2/382 |
| 4000_4100 | Priority Registers Slave (AXBS_PRS1) | 32 | R/W | See section | 18.2.1/379 |
| 4000_4110 | Control Register (AXBS_CRS1) | 32 | R/W | 0000_0000h | 18.2.2/382 |
| 4000_4200 | Priority Registers Slave (AXBS_PRS2) | 32 | R/W | See section | 18.2.1/379 |
| 4000_4210 | Control Register (AXBS_CRS2) | 32 | R/W | 0000_0000h | 18.2.2/382 |
| 4000_4300 | Priority Registers Slave (AXBS_PRS3) | 32 | R/W | See section | 18.2.1/379 |
| 4000_4310 | Control Register (AXBS_CRS3) | 32 | R/W | 0000_0000h | 18.2.2/382 |
| 4000_4400 | Priority Registers Slave (AXBS_PRS4) | 32 | R/W | See section | 18.2.1/379 |
| 4000_4410 | Control Register (AXBS_CRS4) | 32 | R/W | 0000_0000h | 18.2.2/382 |
| 4000_4500 | Priority Registers Slave (AXBS_PRS5) | 32 | R/W | See section | 18.2.1/379 |
| 4000_4510 | Control Register (AXBS_CRS5) | 32 | R/W | 0000_0000h | 18.2.2/382 |
| 4000_4600 | Priority Registers Slave (AXBS_PRS6) | 32 | R/W | See section | 18.2.1/379 |
| 4000_4610 | Control Register (AXBS_CRS6) | 32 | R/W | 0000_0000h | 18.2.2/382 |
| 4000_4700 | Priority Registers Slave (AXBS_PRS7) | 32 | R/W | See section | 18.2.1/379 |

*Table continues on the next page...*

**AXBS memory map (continued)**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4000_4710 | Control Register (AXBS_CRS7) | 32 | R/W | 0000_0000h | 18.2.2/382 |
| 4000_4800 | Master General Purpose Control Register (AXBS_MGPCR0) | 32 | R/W | 0000_0000h | 18.2.3/383 |
| 4000_4900 | Master General Purpose Control Register (AXBS_MGPCR1) | 32 | R/W | 0000_0000h | 18.2.3/383 |
| 4000_4A00 | Master General Purpose Control Register (AXBS_MGPCR2) | 32 | R/W | 0000_0000h | 18.2.3/383 |
| 4000_4B00 | Master General Purpose Control Register (AXBS_MGPCR3) | 32 | R/W | 0000_0000h | 18.2.3/383 |
| 4000_4C00 | Master General Purpose Control Register (AXBS_MGPCR4) | 32 | R/W | 0000_0000h | 18.2.3/383 |
| 4000_4D00 | Master General Purpose Control Register (AXBS_MGPCR5) | 32 | R/W | 0000_0000h | 18.2.3/383 |
| 4000_4E00 | Master General Purpose Control Register (AXBS_MGPCR6) | 32 | R/W | 0000_0000h | 18.2.3/383 |
| 4000_4F00 | Master General Purpose Control Register (AXBS_MGPCR7) | 32 | R/W | 0000_0000h | 18.2.3/383 |

## 18.2.1 Priority Registers Slave (AXBS_PRS*n*)

The priority registers (PRSn) set the priority of each master port on a per slave port basis and reside in each slave port. The priority register can be accessed only with 32-bit accesses. After the CRSn[RO] bit is set, the PRSn register can only be read; attempts to write to it have no effect on PRSn and result in a bus-error response to the master initiating the write.

Two available masters must not be programmed with the same priority level. Attempts to program two or more masters with the same priority level result in a bus-error response and the PRSn is not updated.

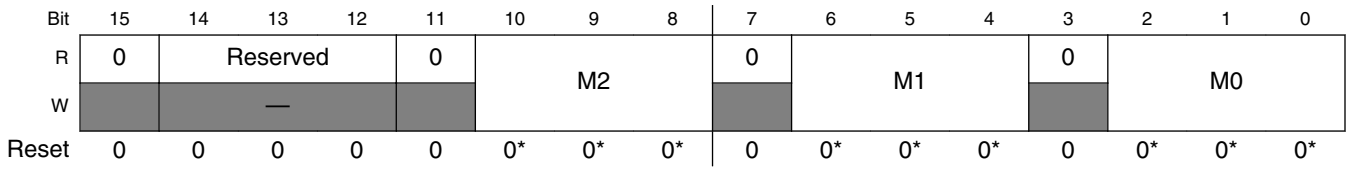
**NOTE**

Valid values for the M*n* priority fields depend on which masters are available on the chip. This information can be found in the chip-specific information for the crossbar.

- If the chip contains less than five masters, values 0 to 3 are valid. Writing other values will result in an error.
- If the chip contains five or more masters, valid values are 0 to n-1, where n is the number of masters attached to the AXBS module. Other values will result in an error.

Address: 4000_4000h base + 0h offset + (256d × i), where i=0d to 7d

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | Reserved | | | 0 | Reserved | | | 0 | M5 | | | 0 | M4 | | |
| W | | — | | | | — | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0* | 0* | 0* | 0 | 0* | 0* | 0* |




* Notes:
- See the chip-specific crossbar information for the reset value of the PRSn registers.

## AXBS_PRS*n* field descriptions

| Field | Description |
|---|---|
| 31<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 30–28<br>Reserved | This field is reserved. |
| 27<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 26–24<br>Reserved | This field is reserved. |
| 23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22–20<br>M5 | Master 5 Priority. Sets the arbitration priority for this port on the associated slave port.<br>000 This master has level 1, or highest, priority when accessing the slave port.<br>001 This master has level 2 priority when accessing the slave port.<br>010 This master has level 3 priority when accessing the slave port.<br>011 This master has level 4 priority when accessing the slave port.<br>100 This master has level 5 priority when accessing the slave port.<br>101 This master has level 6 priority when accessing the slave port.<br>110 This master has level 7 priority when accessing the slave port.<br>111 This master has level 8, or lowest, priority when accessing the slave port. |
| 19<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 18–16<br>M4 | Master 4 Priority. Sets the arbitration priority for this port on the associated slave port.<br>000 This master has level 1, or highest, priority when accessing the slave port.<br>001 This master has level 2 priority when accessing the slave port.<br>010 This master has level 3 priority when accessing the slave port.<br>011 This master has level 4 priority when accessing the slave port.<br>100 This master has level 5 priority when accessing the slave port.<br>101 This master has level 6 priority when accessing the slave port.<br>110 This master has level 7 priority when accessing the slave port.<br>111 This master has level 8, or lowest, priority when accessing the slave port. |
| 15<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 14–12<br>Reserved | This field is reserved. |
| 11<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

*Table continues on the next page...*

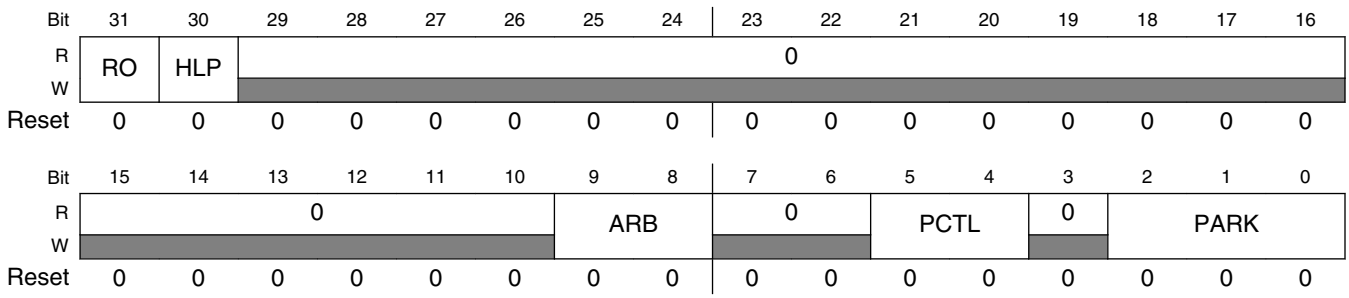
## AXBS_PRS*n* field descriptions (continued)

| Field | Description |
|---|---|
| 10–8<br>M2 | Master 2 Priority. Sets the arbitration priority for this port on the associated slave port.<br><br>000 This master has level 1, or highest, priority when accessing the slave port.<br>001 This master has level 2 priority when accessing the slave port.<br>010 This master has level 3 priority when accessing the slave port.<br>011 This master has level 4 priority when accessing the slave port.<br>100 This master has level 5 priority when accessing the slave port.<br>101 This master has level 6 priority when accessing the slave port.<br>110 This master has level 7 priority when accessing the slave port.<br>111 This master has level 8, or lowest, priority when accessing the slave port. |
| 7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6–4<br>M1 | Master 1 Priority. Sets the arbitration priority for this port on the associated slave port.<br><br>000 This master has level 1, or highest, priority when accessing the slave port.<br>001 This master has level 2 priority when accessing the slave port.<br>010 This master has level 3 priority when accessing the slave port.<br>011 This master has level 4 priority when accessing the slave port.<br>100 This master has level 5 priority when accessing the slave port.<br>101 This master has level 6 priority when accessing the slave port.<br>110 This master has level 7 priority when accessing the slave port.<br>111 This master has level 8, or lowest, priority when accessing the slave port. |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2–0<br>M0 | Master 0 Priority. Sets the arbitration priority for this port on the associated slave port.<br><br>000 This master has level 1, or highest, priority when accessing the slave port.<br>001 This master has level 2 priority when accessing the slave port.<br>010 This master has level 3 priority when accessing the slave port.<br>011 This master has level 4 priority when accessing the slave port.<br>100 This master has level 5 priority when accessing the slave port.<br>101 This master has level 6 priority when accessing the slave port.<br>110 This master has level 7 priority when accessing the slave port.<br>111 This master has level 8, or lowest, priority when accessing the slave port. |

## 18.2.2 Control Register (AXBS_CRS*n*)

These registers control several features of each slave port and must be accessed using 32-bit accesses. After CRSn[RO] is set, the PRSn can only be read; attempts to write to it have no effect and result in an error response.

Address: 4000_4000h base + 10h offset + (256d × i), where i=0d to 7d




**AXBS_CRS*n* field descriptions**

| Field | Description |
|---|---|
| 31<br>RO | Read Only<br><br>Forces the slave port's CSRn and PRSn registers to be read-only. After set, only a hardware reset clears it.<br><br>0 The slave port's registers are writeable<br>1 The slave port's registers are read-only and cannot be written. Attempted writes have no effect on the registers and result in a bus error response. |
| 30<br>HLP | Halt Low Priority<br><br>Sets the initial arbitration priority for low power mode requests . Setting this bit will not affect the request for low power mode from attaining highest priority once it has control of the slave ports.<br><br>0 The low power mode request has the highest priority for arbitration on this slave port<br>1 The low power mode request has the lowest initial priority for arbitration on this slave port |
| 29–10<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 9–8<br>ARB | Arbitration Mode<br><br>Selects the arbitration policy for the slave port.<br><br>00 Fixed priority<br>01 Round-robin, or rotating, priority<br>10 Reserved<br>11 Reserved |
| 7–6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5–4<br>PCTL | Parking Control |

*Table continues on the next page...*

**AXBS_CRS*n* field descriptions (continued)**

| Field | Description |
|---|---|
| | Determines the slave port's parking control. The low-power park feature results in an overall power savings if the slave port is not saturated. However, this forces an extra latency clock when any master tries to access the slave port while not in use because it is not parked on any master.<br><br>00 When no master makes a request, the arbiter parks the slave port on the master port defined by the PARK field<br>01 When no master makes a request, the arbiter parks the slave port on the last master to be in control of the slave port<br>10 When no master makes a request, the slave port is not parked on a master and the arbiter drives all outputs to a constant safe state<br>11 Reserved |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2–0<br>PARK | Park<br><br>Determines which master port the current slave port parks on when no masters are actively making requests and the PCTL bits are cleared.<br><br>**NOTE:** Select only master ports that are present on the chip. Otherwise, undefined behavior might occur.<br><br>000 Park on master port M0<br>001 Park on master port M1<br>010 Park on master port M2<br>011 Park on master port M3<br>100 Park on master port M4<br>101 Park on master port M5<br>110 Park on master port M6<br>111 Park on master port M7 |

## 18.2.3 Master General Purpose Control Register (AXBS_MGPCR*n*)

The MGPCR controls only whether the master's undefined length burst accesses are allowed to complete uninterrupted or whether they can be broken by requests from higher priority masters. The MGPCR can be accessed only in Supervisor mode with 32-bit accesses.

Address: 4000_4000h base + 800h offset + (256d × i), where i=0d to 7d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | AULB | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AXBS_MGPCR*n* field descriptions**

| Field | Description |
|---|---|
| 31–3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2–0<br>AULB | Arbitrates On Undefined Length Bursts<br><br>Determines whether, and when, the crossbar switch arbitrates away the slave port the master owns when the master is performing undefined length burst accesses.<br><br>000 No arbitration is allowed during an undefined length burst<br>001 Arbitration is allowed at any time during an undefined length burst<br>010 Arbitration is allowed after four beats of an undefined length burst<br>011 Arbitration is allowed after eight beats of an undefined length burst<br>100 Arbitration is allowed after 16 beats of an undefined length burst<br>101 Reserved<br>110 Reserved<br>111 Reserved |

## 18.3 Functional Description

### 18.3.1 General operation

When a master accesses the crossbar switch the access is immediately taken. If the targeted slave port of the access is available, then the access is immediately presented on the slave port. Single-clock or zero-wait-state accesses are possible through the crossbar. If the targeted slave port of the access is busy or parked on a different master port, the requesting master simply sees wait states inserted until the targeted slave port can service the master's request. The latency in servicing the request depends on each master's priority level and the responding slave's access time.

Because the crossbar switch appears to be just another slave to the master device, the master device has no knowledge of whether it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting, it simply waits.

A master is given control of the targeted slave port only after a previous access to a different slave port completes, regardless of its priority on the newly targeted slave port. This prevents deadlock from occurring when:

- A higher priority master has:
  - An outstanding request to one slave port that has a long response time and
  - A pending access to a different slave port, and
- A lower priority master is also making a request to the same slave port as the pending access of the higher priority master.

After the master has control of the slave port it is targeting, the master remains in control of the slave port until it relinquishes the slave port by running an IDLE cycle or by targeting a different slave port for its next access.

The master can also lose control of the slave port if another higher-priority master makes a request to the slave port. However, if the master is running a fixed- or undefined-length burst transfer it retains control of the slave port until that transfer completes. Based on MGPCR[AULB], the master either retains control of the slave port when doing undefined-length, incrementing burst transfers or loses the bus to a higher-priority master.

The crossbar terminates all master IDLE transfers, as opposed to allowing the termination to come from one of the slave buses. Additionally, when no master is requesting access to a slave port, the crossbar drives IDLE transfers onto the slave bus, even though a default master may be granted access to the slave port.

When a slave bus is being idled by the crossbar, it can park the slave port on the master port indicated by CRS*n*[PARK]. This is done to save the initial clock of arbitration delay that otherwise would be seen if the master had to arbitrate to gain control of the slave port. The slave port can also be put into Low Power Park mode to save power, by using CRS*n*[PCTL].

### 18.3.2 Register coherency

The operation of the crossbar is affected as soon as a register is written. The values of the registers do not track with slave-port-related master accesses, but instead track only with slave accesses.

The MGPCRx[AULB] bits are the exception to this rule. The update of these bits is only recognized when the master on that master port runs an IDLE cycle, even though the slave bus cycle to write them will have already terminated successfully. If the MGPCRx[AULB] bits are written between two burst accesses, the new AULB encodings do not take effect until an IDLE cycle is initiated by the master on that master port.

### 18.3.3 Arbitration

The crossbar switch supports two arbitration algorithms:

- Fixed priority
- Round robin

The arbitration scheme is independently programmable for each slave port.

### 18.3.3.1 Arbitration during undefined length bursts

Arbitration points during an undefined length burst are defined by the current master's MGPCR[AULB] field setting. When a defined length is imposed on the burst via the AULB bits, the undefined length burst is treated as a single or series of single back-to-back fixed-length burst accesses.

The following figure illustrates an example:

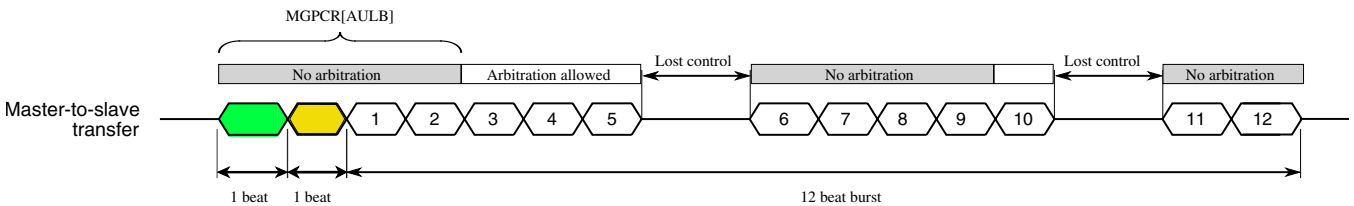



**Figure 18-28. Undefined length burst example**

In this example, a master runs an undefined length burst and the MGPCR[AULB] bits indicate arbitration occurs after the fourth beat of the burst. The master runs two sequential beats and then starts what will be a 12-beat undefined length burst access to a new address within the same slave port region as the previous access. The crossbar does not allow an arbitration point until the fourth overall access, or the second beat of the second burst. At that point, all remaining accesses are open for arbitration until the master loses control of the slave port.

Assume the master loses control of the slave port after the fifth beat of the second burst. After the master regains control of the slave port no arbitration point is available until after the master has run four more beats of its burst. After the fourth beat of the now continued burst, or the ninth beat of the second burst from the master's perspective, is taken, all beats of the burst are once again open for arbitration until the master loses control of the slave port.

Assume the master again loses control of the slave port on the fifth beat of the third now continued burst, or the 10th beat of the second burst from the master's perspective. After the master regains control of the slave port, it is allowed to complete its final two beats of its burst without facing arbitration.

**Note**

Fixed-length burst accesses are not affected by the AULB bits. All fixed-length burst accesses lock out arbitration until the last beat of the fixed-length burst.

### 18.3.3.2 Fixed-priority operation

When operating in fixed-priority mode, each master is assigned a unique priority level in the priority registers (PRS*n*) . If two masters request access to the same slave port, the master with the highest priority in the selected priority register gains control over the slave port.

**NOTE**

In this arbitration mode, a higher-priority master can monopolize a slave port, preventing accesses from any lower-priority master to the port.

When a master makes a request to a slave port, the slave port checks whether the new requesting master's priority level is higher than that of the master that currently has control over the slave port, unless the slave port is in a parked state. The slave port performs an arbitration check at every clock edge to ensure that the proper master, if any, has control of the slave port.

The following table describes possible scenarios based on the requesting master port:

**Table 18-29. How AXBS grants control of a slave port to a master**

| When | Then AXBS grants control to the requesting master |
|---|---|
| Both of the following are true:<br>• The current master is not running a transfer.<br>• The new requesting master's priority level is higher than that of the current master. | At the next clock edge |
| Both of the following are true:<br>• The current master is running a fixed length burst transfer or a locked transfer.<br>• The requesting master's priority level is higher than that of the current master. | At the end of the burst transfer or locked transfer |
| Both of the following are true:<br>• The current master is running an undefined length burst transfer.<br>• The requesting master's priority level is higher than that of the current master. | At the next arbitration point for the undefined length burst transfer<br>**NOTE:** Arbitration points for an undefined length burst are defined in the MGPCR for each master. |
| The requesting master's priority level is lower than the current master. | At the conclusion of one of the following cycles:<br>• An IDLE cycle<br>• A non-IDLE cycle to a location other than the current slave port |

### 18.3.3.3 Round-robin priority operation

When operating in round-robin mode, each master is assigned a relative priority based on the master port number. This relative priority is compared to the master port number (ID) of the last master to perform a transfer on the slave bus. The highest priority requesting

master becomes owner of the slave bus at the next transfer boundary, accounting for locked and fixed-length burst transfers. Priority is based on how far ahead the ID of the requesting master is to the ID of the last master.

After granted access to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line is granted access to the slave port at the next transfer boundary, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the crossbar is implemented with master ports 0, 1, 4, and 5. If the last master of the slave port was master 1, and master 0, 4, and 5 make simultaneous requests, they are serviced in the order: 4 then 5 then 0.

Parking may continue to be used in a round-robin mode, but does not affect the round-robin pointer unless the parked master actually performs a transfer. Handoff occurs to the next master in line after one cycle of arbitration. If the slave port is put into low-power park mode, the round-robin pointer is reset to point at master port 0, giving it the highest priority.

#### 18.3.3.4 Priority assignment

Each master port must be assigned a unique 3-bit priority level. If an attempt is made to program multiple master ports with the same priority level within the priority registers (PRS*n*), the crossbar switch responds with a bus error and the registers are not updated.

## 18.4 Initialization/application information

No initialization is required for the crossbar switch. Hardware reset ensures all the register bits used by the crossbar switch are properly initialized to a valid state. However, settings and priorities may be programmed to achieve maximum system performance.

# Chapter 19
# Memory Protection Unit (MPU)

## 19.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The memory protection unit (MPU) provides hardware access control for all memory references generated in the device.

## 19.2 Overview

The MPU concurrently monitors all system bus transactions and evaluates their appropriateness using pre-programmed region descriptors that define memory spaces and their access rights. Memory references that have sufficient access control rights are allowed to complete, while references that are not mapped to any region descriptor or have insufficient rights are terminated with a protection error response.

### 19.2.1 Block diagram

A simplified block diagram of the MPU module is shown in the following figure.

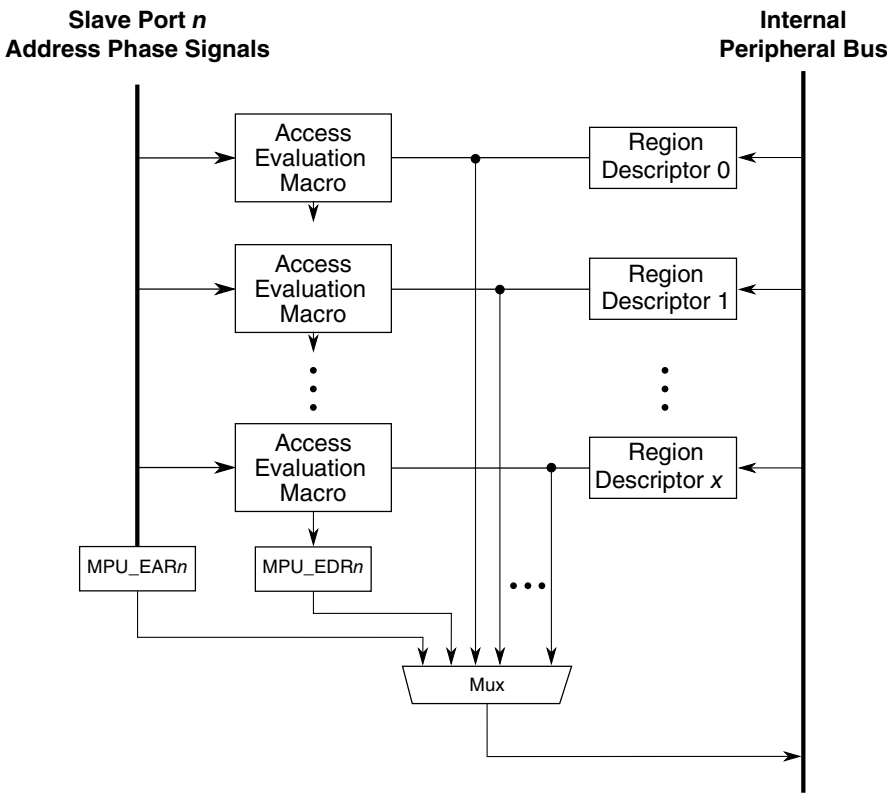



**Figure 19-1. MPU block diagram**

The hardware's two-dimensional connection matrix is clearly visible with the basic access evaluation macro shown as the replicated submodule block. The crossbar switch slave ports are shown on the left, the region descriptor registers in the middle, and the peripheral bus interface on the right side. The evaluation macro contains two magnitude comparators connected to the start and end address registers from each region descriptor as well as the combinational logic blocks to determine the region hit and the access protection error. For details of the access evaluation macro, see Access evaluation macro.

## 19.2.2 Features

The MPU implements a two-dimensional hardware array of memory region descriptors and the crossbar slave ports to continuously monitor the legality of every memory reference generated by each bus master in the system.

The feature set includes:

- 12 program-visible 128-bit region descriptors, accessible by four 32-bit words each
  - Each region descriptor defines a modulo-32 byte space, aligned anywhere in memory

        - Region sizes can vary from 32 bytes to 4 Gbytes
    - Two access control permissions defined in a single descriptor word
        - Masters 0–3: read, write, and execute attributes for supervisor and user accesses
        - Masters 4–7: read and write attributes
    - Hardware-assisted maintenance of the descriptor valid bit minimizes coherency issues
    - Alternate programming model view of the access control permissions word
    - Priority given to granting permission over denying access for overlapping region descriptors
- Detects access protection errors if a memory reference does not hit in any memory region, or if the reference is illegal in all hit memory regions. If an access error occurs, the reference is terminated with an error response, and the MPU inhibits the bus cycle being sent to the targeted slave device.
- Error registers, per slave port, capture the last faulting address, attributes, and other information
- Global MPU enable/disable control bit

## 19.3 Memory map/register definition

The programming model is partitioned into three groups:

- Control/status registers
- The data structure containing the region descriptors
- The alternate view of the region descriptor access control values

The programming model can only be referenced using 32-bit accesses. Attempted references using different access sizes, to undefined, that is, reserved, addresses, or with a non-supported access type, such as a write to a read-only register, or a read of a write-only register, generate an error termination.

The programming model can be accessed only in supervisor mode.

**NOTE**

See the chip configuration details for any chip-specific register information this module.

## MPU memory map

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4000_D000 | Control/Error Status Register (MPU_CESR) | 32 | R/W | 0081_5101h | 19.3.1/394 |
| 4000_D010 | Error Address Register, slave port n (MPU_EAR0) | 32 | R | Undefined | 19.3.2/395 |
| 4000_D014 | Error Detail Register, slave port n (MPU_EDR0) | 32 | R | Undefined | 19.3.3/396 |
| 4000_D018 | Error Address Register, slave port n (MPU_EAR1) | 32 | R | Undefined | 19.3.2/395 |
| 4000_D01C | Error Detail Register, slave port n (MPU_EDR1) | 32 | R | Undefined | 19.3.3/396 |
| 4000_D020 | Error Address Register, slave port n (MPU_EAR2) | 32 | R | Undefined | 19.3.2/395 |
| 4000_D024 | Error Detail Register, slave port n (MPU_EDR2) | 32 | R | Undefined | 19.3.3/396 |
| 4000_D028 | Error Address Register, slave port n (MPU_EAR3) | 32 | R | Undefined | 19.3.2/395 |
| 4000_D02C | Error Detail Register, slave port n (MPU_EDR3) | 32 | R | Undefined | 19.3.3/396 |
| 4000_D030 | Error Address Register, slave port n (MPU_EAR4) | 32 | R | Undefined | 19.3.2/395 |
| 4000_D034 | Error Detail Register, slave port n (MPU_EDR4) | 32 | R | Undefined | 19.3.3/396 |
| 4000_D400 | Region Descriptor n, Word 0 (MPU_RGD0_WORD0) | 32 | R/W | 0000_0000h | 19.3.4/397 |
| 4000_D404 | Region Descriptor n, Word 1 (MPU_RGD0_WORD1) | 32 | R/W | See section | 19.3.5/398 |
| 4000_D408 | Region Descriptor n, Word 2 (MPU_RGD0_WORD2) | 32 | R/W | See section | 19.3.6/398 |
| 4000_D40C | Region Descriptor n, Word 3 (MPU_RGD0_WORD3) | 32 | R/W | See section | 19.3.7/401 |
| 4000_D410 | Region Descriptor n, Word 0 (MPU_RGD1_WORD0) | 32 | R/W | 0000_0000h | 19.3.4/397 |
| 4000_D414 | Region Descriptor n, Word 1 (MPU_RGD1_WORD1) | 32 | R/W | See section | 19.3.5/398 |
| 4000_D418 | Region Descriptor n, Word 2 (MPU_RGD1_WORD2) | 32 | R/W | See section | 19.3.6/398 |
| 4000_D41C | Region Descriptor n, Word 3 (MPU_RGD1_WORD3) | 32 | R/W | See section | 19.3.7/401 |
| 4000_D420 | Region Descriptor n, Word 0 (MPU_RGD2_WORD0) | 32 | R/W | 0000_0000h | 19.3.4/397 |
| 4000_D424 | Region Descriptor n, Word 1 (MPU_RGD2_WORD1) | 32 | R/W | See section | 19.3.5/398 |
| 4000_D428 | Region Descriptor n, Word 2 (MPU_RGD2_WORD2) | 32 | R/W | See section | 19.3.6/398 |
| 4000_D42C | Region Descriptor n, Word 3 (MPU_RGD2_WORD3) | 32 | R/W | See section | 19.3.7/401 |
| 4000_D430 | Region Descriptor n, Word 0 (MPU_RGD3_WORD0) | 32 | R/W | 0000_0000h | 19.3.4/397 |
| 4000_D434 | Region Descriptor n, Word 1 (MPU_RGD3_WORD1) | 32 | R/W | See section | 19.3.5/398 |
| 4000_D438 | Region Descriptor n, Word 2 (MPU_RGD3_WORD2) | 32 | R/W | See section | 19.3.6/398 |
| 4000_D43C | Region Descriptor n, Word 3 (MPU_RGD3_WORD3) | 32 | R/W | See section | 19.3.7/401 |
| 4000_D440 | Region Descriptor n, Word 0 (MPU_RGD4_WORD0) | 32 | R/W | 0000_0000h | 19.3.4/397 |
| 4000_D444 | Region Descriptor n, Word 1 (MPU_RGD4_WORD1) | 32 | R/W | See section | 19.3.5/398 |
| 4000_D448 | Region Descriptor n, Word 2 (MPU_RGD4_WORD2) | 32 | R/W | See section | 19.3.6/398 |
| 4000_D44C | Region Descriptor n, Word 3 (MPU_RGD4_WORD3) | 32 | R/W | See section | 19.3.7/401 |
| 4000_D450 | Region Descriptor n, Word 0 (MPU_RGD5_WORD0) | 32 | R/W | 0000_0000h | 19.3.4/397 |
| 4000_D454 | Region Descriptor n, Word 1 (MPU_RGD5_WORD1) | 32 | R/W | See section | 19.3.5/398 |
| 4000_D458 | Region Descriptor n, Word 2 (MPU_RGD5_WORD2) | 32 | R/W | See section | 19.3.6/398 |
| 4000_D45C | Region Descriptor n, Word 3 (MPU_RGD5_WORD3) | 32 | R/W | See section | 19.3.7/401 |
| 4000_D460 | Region Descriptor n, Word 0 (MPU_RGD6_WORD0) | 32 | R/W | 0000_0000h | 19.3.4/397 |
| 4000_D464 | Region Descriptor n, Word 1 (MPU_RGD6_WORD1) | 32 | R/W | See section | 19.3.5/398 |
| 4000_D468 | Region Descriptor n, Word 2 (MPU_RGD6_WORD2) | 32 | R/W | See section | 19.3.6/398 |

*Table continues on the next page...*

## MPU memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4000_D46C | Region Descriptor n, Word 3 (MPU_RGD6_WORD3) | 32 | R/W | See section | 19.3.7/401 |
| 4000_D470 | Region Descriptor n, Word 0 (MPU_RGD7_WORD0) | 32 | R/W | 0000_0000h | 19.3.4/397 |
| 4000_D474 | Region Descriptor n, Word 1 (MPU_RGD7_WORD1) | 32 | R/W | See section | 19.3.5/398 |
| 4000_D478 | Region Descriptor n, Word 2 (MPU_RGD7_WORD2) | 32 | R/W | See section | 19.3.6/398 |
| 4000_D47C | Region Descriptor n, Word 3 (MPU_RGD7_WORD3) | 32 | R/W | See section | 19.3.7/401 |
| 4000_D480 | Region Descriptor n, Word 0 (MPU_RGD8_WORD0) | 32 | R/W | 0000_0000h | 19.3.4/397 |
| 4000_D484 | Region Descriptor n, Word 1 (MPU_RGD8_WORD1) | 32 | R/W | See section | 19.3.5/398 |
| 4000_D488 | Region Descriptor n, Word 2 (MPU_RGD8_WORD2) | 32 | R/W | See section | 19.3.6/398 |
| 4000_D48C | Region Descriptor n, Word 3 (MPU_RGD8_WORD3) | 32 | R/W | See section | 19.3.7/401 |
| 4000_D490 | Region Descriptor n, Word 0 (MPU_RGD9_WORD0) | 32 | R/W | 0000_0000h | 19.3.4/397 |
| 4000_D494 | Region Descriptor n, Word 1 (MPU_RGD9_WORD1) | 32 | R/W | See section | 19.3.5/398 |
| 4000_D498 | Region Descriptor n, Word 2 (MPU_RGD9_WORD2) | 32 | R/W | See section | 19.3.6/398 |
| 4000_D49C | Region Descriptor n, Word 3 (MPU_RGD9_WORD3) | 32 | R/W | See section | 19.3.7/401 |
| 4000_D4A0 | Region Descriptor n, Word 0 (MPU_RGD10_WORD0) | 32 | R/W | 0000_0000h | 19.3.4/397 |
| 4000_D4A4 | Region Descriptor n, Word 1 (MPU_RGD10_WORD1) | 32 | R/W | See section | 19.3.5/398 |
| 4000_D4A8 | Region Descriptor n, Word 2 (MPU_RGD10_WORD2) | 32 | R/W | See section | 19.3.6/398 |
| 4000_D4AC | Region Descriptor n, Word 3 (MPU_RGD10_WORD3) | 32 | R/W | See section | 19.3.7/401 |
| 4000_D4B0 | Region Descriptor n, Word 0 (MPU_RGD11_WORD0) | 32 | R/W | 0000_0000h | 19.3.4/397 |
| 4000_D4B4 | Region Descriptor n, Word 1 (MPU_RGD11_WORD1) | 32 | R/W | See section | 19.3.5/398 |
| 4000_D4B8 | Region Descriptor n, Word 2 (MPU_RGD11_WORD2) | 32 | R/W | See section | 19.3.6/398 |
| 4000_D4BC | Region Descriptor n, Word 3 (MPU_RGD11_WORD3) | 32 | R/W | See section | 19.3.7/401 |
| 4000_D800 | Region Descriptor Alternate Access Control n (MPU_RGDAAC0) | 32 | R/W | See section | 19.3.8/402 |
| 4000_D804 | Region Descriptor Alternate Access Control n (MPU_RGDAAC1) | 32 | R/W | See section | 19.3.8/402 |
| 4000_D808 | Region Descriptor Alternate Access Control n (MPU_RGDAAC2) | 32 | R/W | See section | 19.3.8/402 |
| 4000_D80C | Region Descriptor Alternate Access Control n (MPU_RGDAAC3) | 32 | R/W | See section | 19.3.8/402 |
| 4000_D810 | Region Descriptor Alternate Access Control n (MPU_RGDAAC4) | 32 | R/W | See section | 19.3.8/402 |
| 4000_D814 | Region Descriptor Alternate Access Control n (MPU_RGDAAC5) | 32 | R/W | See section | 19.3.8/402 |
| 4000_D818 | Region Descriptor Alternate Access Control n (MPU_RGDAAC6) | 32 | R/W | See section | 19.3.8/402 |
| 4000_D81C | Region Descriptor Alternate Access Control n (MPU_RGDAAC7) | 32 | R/W | See section | 19.3.8/402 |
| 4000_D820 | Region Descriptor Alternate Access Control n (MPU_RGDAAC8) | 32 | R/W | See section | 19.3.8/402 |
| 4000_D824 | Region Descriptor Alternate Access Control n (MPU_RGDAAC9) | 32 | R/W | See section | 19.3.8/402 |

*Table continues on the next page...*

**MPU memory map (continued)**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4000_D828 | Region Descriptor Alternate Access Control n (MPU_RGDAAC10) | 32 | R/W | See section | 19.3.8/402 |
| 4000_D82C | Region Descriptor Alternate Access Control n (MPU_RGDAAC11) | 32 | R/W | See section | 19.3.8/402 |

## 19.3.1 Control/Error Status Register (MPU_CESR)

Address: 4000_D000h base + 0h offset = 4000_D000h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SPERR | | | | | 0 | | | 1 | 0 | | | HRL | | | |
| W | w1c | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | NSP | | | | NRGD | | | | 0 | | | | | | | VLD |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**MPU_CESR field descriptions**

| Field | Description |
|---|---|
| 31–27<br>SPERR | Slave Port n Error<br><br>Indicates a captured error in EARn and EDRn. This bit is set when the hardware detects an error and records the faulting address and attributes. It is cleared by writing one to it. If another error is captured at the exact same cycle as the write, the flag remains set. A find-first-one instruction or equivalent can detect the presence of a captured error.<br><br>The following shows the correspondence between the bit number and slave port number:<br><br>• Bit 31 corresponds to slave port 0.<br>• Bit 30 corresponds to slave port 1.<br>• Bit 29 corresponds to slave port 2.<br>• Bit 28 corresponds to slave port 3.<br>• Bit 27 corresponds to slave port 4.<br><br>0 No error has occurred for slave port n.<br>1 An error has occurred for slave port n. |
| 26–24<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 1. |
| 22–20<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

*Table continues on the next page...*

**MPU_CESR field descriptions (continued)**

| Field | Description |
|---|---|
| 19–16<br>HRL | Hardware Revision Level<br><br>Specifies the MPU's hardware and definition revision level. It can be read by software to determine the functional definition of the module. |
| 15–12<br>NSP | Number Of Slave Ports<br><br>Specifies the number of slave ports connected to the MPU. |
| 11–8<br>NRGD | Number Of Region Descriptors<br><br>Indicates the number of region descriptors implemented in the MPU.<br><br>0000 8 region descriptors<br>0001 12 region descriptors<br>0010 16 region descriptors |
| 7–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>VLD | Valid<br><br>Global enable/disable for the MPU.<br><br>0 MPU is disabled. All accesses from all bus masters are allowed.<br>1 MPU is enabled |

## 19.3.2 Error Address Register, slave port n (MPU_EAR*n*)

When the MPU detects an access error on slave port n, the 32-bit reference address is captured in this read-only register and the corresponding bit in CESR[SPERR] set. Additional information about the faulting access is captured in the corresponding EDRn at the same time. This register and the corresponding EDRn contain the most recent access error; there are no hardware interlocks with CESR[SPERR], as the error registers are always loaded upon the occurrence of each protection violation.

Address: 4000_D000h base + 10h offset + (8d × i), where i=0d to 4d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EADDR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**MPU_EAR*n* field descriptions**

| Field | Description |
|---|---|
| 31–0<br>EADDR | Error Address<br><br>Indicates the reference address from slave port n that generated the access error |

## 19.3.3 Error Detail Register, slave port n (MPU_EDR*n*)

When the MPU detects an access error on slave port n, 32 bits of error detail are captured in this read-only register and the corresponding bit in CESR[SPERR] is set. Information on the faulting address is captured in the corresponding EARn register at the same time. This register and the corresponding EARn register contain the most recent access error; there are no hardware interlocks with CESR[SPERR] as the error registers are always loaded upon the occurrence of each protection violation.

Address: 4000_D000h base + 14h offset + (8d × i), where i=0d to 4d

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EACD | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EPID | | | | | | | | EMN | | | | EATTR | | | ERW |
| W | | | | | | | | | | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**MPU_EDR*n* field descriptions**

| Field | Description |
|---|---|
| 31–16<br>EACD | Error Access Control Detail<br><br>Indicates the region descriptor with the access error.<br>• If EDRn contains a captured error and EACD is cleared, an access did not hit in any region descriptor.<br>• If only a single EACD bit is set, the protection error was caused by a single non-overlapping region descriptor.<br>• If two or more EACD bits are set, the protection error was caused by an overlapping set of region descriptors. |
| 15–8<br>EPID | Error Process Identification<br><br>Records the process identifier of the faulting reference. The process identifier is typically driven only by processor cores; for other bus masters, this field is cleared. |

*Table continues on the next page...*

**MPU_EDR*n* field descriptions (continued)**

| Field | Description |
|---|---|
| 7–4<br>EMN | Error Master Number<br><br>Indicates the bus master that generated the access error. |
| 3–1<br>EATTR | Error Attributes<br><br>Indicates attribute information about the faulting reference.<br><br>**NOTE:** All other encodings are reserved.<br><br>000 User mode, instruction access<br>001 User mode, data access<br>010 Supervisor mode, instruction access<br>011 Supervisor mode, data access |
| 0<br>ERW | Error Read/Write<br><br>Indicates the access type of the faulting reference.<br><br>0 Read<br>1 Write |

## 19.3.4 Region Descriptor n, Word 0 (MPU_RGD*n*_WORD0)

The first word of the region descriptor defines the 0-modulo-32 byte start address of the memory region. Writes to this register clear the region descriptor's valid bit (RGDn_WORD3[VLD]).

Address: 4000_D000h base + 400h offset + (16d × i), where i=0d to 11d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SRTADDR | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MPU_RGD*n*_WORD0 field descriptions**

| Field | Description |
|---|---|
| 31–5<br>SRTADDR | Start Address<br><br>Defines the most significant bits of the 0-modulo-32 byte start address of the memory region. |
| 4–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 19.3.5 Region Descriptor n, Word 1 (MPU_RGD*n*_WORD1)

The second word of the region descriptor defines the 31-modulo-32 byte end address of the memory region. Writes to this register clear the region descriptor's valid bit (RGDn_WORD3[VLD]).

Address: 4000_D000h base + 404h offset + (16d × i), where i=0d to 11d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ENDADDR | | | | | | | | | | | | | | | | | | | | | | | | | | | Reserved | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

* Notes:
- • Reset value for MPU_RGD0_WORD1 is FFFF_FFFFh
  - Reset value for MPU_RGD[1:7]_WORD1 is 0000_001Fh

**MPU_RGD*n*_WORD1 field descriptions**

| Field | Description |
|---|---|
| 31–5<br>ENDADDR | End Address<br><br>Defines the most significant bits of the 31-modulo-32 byte end address of the memory region.<br><br>**NOTE:** The MPU does not verify that ENDADDR ≥ SRTADDR. |
| 4–0<br>Reserved | This field is reserved. |

## 19.3.6 Region Descriptor n, Word 2 (MPU_RGD*n*_WORD2)

The third word of the region descriptor defines the access control rights of the memory region. The access control privileges depend on two broad classifications of bus masters:

- Bus masters 0–3 have a 5-bit field defining separate privilege rights for user and supervisor mode accesses, as well as the optional inclusion of a process identification field within the definition.
- Bus masters 4–7 are limited to separate read and write permissions.

For the privilege rights of bus masters 0–3, there are three flags associated with this function:

- Read (r) refers to accessing the referenced memory address using an operand (data) fetch

- Write (w) refers to updating the referenced memory address using a store (data) instruction
- Execute (x) refers to reading the referenced memory address using an instruction fetch

Writes to RGDn_WORD2 clear the region descriptor's valid bit (RGDn_WORD3[VLD]). If only updating the access controls, write to RGDAACn instead because stores to these locations do not affect the descriptor's valid bit.

Address: 4000_D000h base + 408h offset + (16d × i), where i=0d to 11d

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | M7RE | M7WE | M6RE | M6WE | M5RE | M5WE | M4RE | M4WE | M3PE | M3SM | | M3UM | | | M2PE | M2SM |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | M2SM | M2UM | | | M1PE | M1SM | | M1UM | | | M0PE | M0SM | | M0UM | | |
| Reset | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

* Notes:
- • Reset value for MPU_RGD0_WORD2 is 0001_F01Fh
  • Reset value for MPU_RGD[1:7]_WORD2 is 0000_0000h

## MPU_RGD*n*_WORD2 field descriptions

| Field | Description |
|---|---|
| 31<br>M7RE | Bus Master 7 Read Enable<br><br>0 Bus master 7 reads terminate with an access error and the read is not performed<br>1 Bus master 7 reads allowed |
| 30<br>M7WE | Bus Master 7 Write Enable<br><br>0 Bus master 7 writes terminate with an access error and the write is not performed<br>1 Bus master 7 writes allowed |
| 29<br>M6RE | Bus Master 6 Read Enable<br><br>0 Bus master 6 reads terminate with an access error and the read is not performed<br>1 Bus master 6 reads allowed |
| 28<br>M6WE | Bus Master 6 Write Enable<br><br>0 Bus master 6 writes terminate with an access error and the write is not performed<br>1 Bus master 6 writes allowed |
| 27<br>M5RE | Bus Master 5 Read Enable |

*Table continues on the next page...*

## MPU_RGD*n*_WORD2 field descriptions (continued)

| Field | Description |
|---|---|
| | 0 Bus master 5 reads terminate with an access error and the read is not performed<br>1 Bus master 5 reads allowed |
| 26<br>M5WE | Bus Master 5 Write Enable<br><br>0 Bus master 5 writes terminate with an access error and the write is not performed<br>1 Bus master 5 writes allowed |
| 25<br>M4RE | Bus Master 4 Read Enable<br><br>0 Bus master 4 reads terminate with an access error and the read is not performed<br>1 Bus master 4 reads allowed |
| 24<br>M4WE | Bus Master 4 Write Enable<br><br>0 Bus master 4 writes terminate with an access error and the write is not performed<br>1 Bus master 4 writes allowed |
| 23<br>M3PE | Bus Master 3 Process Identifier Enable<br><br>0 Do not include the process identifier in the evaluation<br>1 Include the process identifier and mask (RGDn_WORD3) in the region hit evaluation |
| 22–21<br>M3SM | Bus Master 3 Supervisor Mode Access Control<br><br>Defines the access controls for bus master 3 in Supervisor mode.<br><br>00 r/w/x; read, write and execute allowed<br>01 r/x; read and execute allowed, but no write<br>10 r/w; read and write allowed, but no execute<br>11 Same as User mode defined in M3UM |
| 20–18<br>M3UM | Bus Master 3 User Mode Access Control<br><br>Defines the access controls for bus master 3 in User mode. M3UM consists of three independent bits, enabling read (r), write (w), and execute (x) permissions.<br><br>0 An attempted access of that mode may be terminated with an access error (if not allowed by another descriptor) and the access not performed.<br>1 Allows the given access type to occur |
| 17<br>M2PE | Bus Master 2 Process Identifier Enable<br><br>See M3PE description. |
| 16–15<br>M2SM | Bus Master 2 Supervisor Mode Access Control<br><br>See M3SM description. |
| 14–12<br>M2UM | Bus Master 2 User Mode Access control<br><br>See M3UM description. |
| 11<br>M1PE | Bus Master 1 Process Identifier enable<br><br>See M1PE description. |
| 10–9<br>M1SM | Bus Master 1 Supervisor Mode Access Control<br><br>See M3SM description. |

*Table continues on the next page...*

**MPU_RGD*n*_WORD2 field descriptions (continued)**

| Field | Description |
|---|---|
| 8–6<br>M1UM | Bus Master 1 User Mode Access Control<br><br>See M3UM description. |
| 5<br>M0PE | Bus Master 0 Process Identifier enable<br><br>See M0PE description. |
| 4–3<br>M0SM | Bus Master 0 Supervisor Mode Access Control<br><br>See M3SM description. |
| 2–0<br>M0UM | Bus Master 0 User Mode Access Control<br><br>See M3UM description. |

## 19.3.7 Region Descriptor n, Word 3 (MPU_RGD*n*_WORD3)

The fourth word of the region descriptor contains the optional process identifier and mask, plus the region descriptor's valid bit.

Address: 4000_D000h base + 40Ch offset + (16d × i), where i=0d to 11d

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

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

* Notes:

- Reset value for MPU_RGD0_WORD3 is 0000_0001h
- Reset value for MPU_RGD[1:7]_WORD3 is 0000_0000h

**MPU_RGD*n*_WORD3 field descriptions**

| Field | Description |
|---|---|
| 31–24<br>PID | Process Identifier<br><br>Specifies the process identifier that is included in the region hit determination if RGDn_WORD2[MxPE] is set. PIDMASK can mask individual bits in this field. |
| 23–16<br>PIDMASK | Process Identifier Mask<br><br>Provides a masking capability so that multiple process identifiers can be included as part of the region hit determination. If a bit in PIDMASK is set, then the corresponding PID bit is ignored in the comparison. This field and PID are included in the region hit determination if RGDn_WORD2[MxPE] is set. For more information on the handling of the PID and PIDMASK, see "Access Evaluation - Hit Determination." |

*Table continues on the next page...*

**MPU_RGD*n*_WORD3 field descriptions (continued)**

| Field | Description |
|---|---|
| 15–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>VLD | Valid<br><br>Signals the region descriptor is valid. Any write to RGDn_WORD0–2 clears this bit.<br><br>0 Region descriptor is invalid<br>1 Region descriptor is valid |

## 19.3.8 Region Descriptor Alternate Access Control n (MPU_RGDAAC*n*)

Because software may adjust only the access controls within a region descriptor (RGDn_WORD2) as different tasks execute, an alternate programming view of this 32-bit entity is available. Writing to this register does not affect the descriptor's valid bit.

Address: 4000_D000h base + 800h offset + (4d × i), where i=0d to 11d

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | M7RE | M7WE | M6RE | M6WE | M5RE | M5WE | M4RE | M4WE | M3PE | M3SM | | M3UM | | | M2PE | M2SM |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | M2SM | M2UM | | | M1PE | M1SM | | | M1UM | | | M0PE | M0SM | | M0UM | |
| Reset | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

* Notes:

- Reset value for MPU_RGDAAC0 is 0001_F01Fh
- Reset value for MPU_RGDAAC[1:7] is 0000_0000h

**MPU_RGDAAC*n* field descriptions**

| Field | Description |
|---|---|
| 31<br>M7RE | Bus Master 7 Read Enable<br><br>0 Bus master 7 reads terminate with an access error and the read is not performed<br>1 Bus master 7 reads allowed |

*Table continues on the next page...*

## MPU_RGDAAC*n* field descriptions (continued)

| Field | Description |
|---|---|
| 30<br>M7WE | Bus Master 7 Write Enable<br><br>0 Bus master 7 writes terminate with an access error and the write is not performed<br>1 Bus master 7 writes allowed |
| 29<br>M6RE | Bus Master 6 Read Enable<br><br>0 Bus master 6 reads terminate with an access error and the read is not performed<br>1 Bus master 6 reads allowed |
| 28<br>M6WE | Bus Master 6 Write Enable<br><br>0 Bus master 6 writes terminate with an access error and the write is not performed<br>1 Bus master 6 writes allowed |
| 27<br>M5RE | Bus Master 5 Read Enable<br><br>0 Bus master 5 reads terminate with an access error and the read is not performed<br>1 Bus master 5 reads allowed |
| 26<br>M5WE | Bus Master 5 Write Enable<br><br>0 Bus master 5 writes terminate with an access error and the write is not performed<br>1 Bus master 5 writes allowed |
| 25<br>M4RE | Bus Master 4 Read Enable<br><br>0 Bus master 4 reads terminate with an access error and the read is not performed<br>1 Bus master 4 reads allowed |
| 24<br>M4WE | Bus Master 4 Write Enable<br><br>0 Bus master 4 writes terminate with an access error and the write is not performed<br>1 Bus master 4 writes allowed |
| 23<br>M3PE | Bus Master 3 Process Identifier Enable<br><br>0 Do not include the process identifier in the evaluation<br>1 Include the process identifier and mask (RGDn.RGDAAC) in the region hit evaluation |
| 22–21<br>M3SM | Bus Master 3 Supervisor Mode Access Control<br><br>Defines the access controls for bus master 3 in Supervisor mode.<br><br>00 r/w/x; read, write and execute allowed<br>01 r/x; read and execute allowed, but no write<br>10 r/w; read and write allowed, but no execute<br>11 Same as User mode defined in M3UM |
| 20–18<br>M3UM | Bus Master 3 User Mode Access Control<br><br>Defines the access controls for bus master 3 in user mode. M3UM consists of three independent bits, enabling read (r), write (w), and execute (x) permissions.<br><br>0 An attempted access of that mode may be terminated with an access error (if not allowed by another descriptor) and the access not performed.<br>1 Allows the given access type to occur |
| 17<br>M2PE | Bus Master 2 Process Identifier Enable |

*Table continues on the next page...*

**MPU_RGDAAC*n* field descriptions (continued)**

| Field | Description |
|---|---|
| | See M3PE description. |
| 16–15<br>M2SM | Bus Master 2 Supervisor Mode Access Control<br><br>See M3SM description. |
| 14–12<br>M2UM | Bus Master 2 User Mode Access Control<br><br>See M3UM description. |
| 11<br>M1PE | Bus Master 1 Process Identifier Enable<br><br>See M3PE description. |
| 10–9<br>M1SM | Bus Master 1 Supervisor Mode Access Control<br><br>See M3SM description. |
| 8–6<br>M1UM | Bus Master 1 User Mode Access Control<br><br>See M3UM description. |
| 5<br>M0PE | Bus Master 0 Process Identifier Enable<br><br>See M3PE description. |
| 4–3<br>M0SM | Bus Master 0 Supervisor Mode Access Control<br><br>See M3SM description. |
| 2–0<br>M0UM | Bus Master 0 User Mode Access Control<br><br>See M3UM description. |

## 19.4 Functional description

In this section, the functional operation of the MPU is detailed, including the operation of the access evaluation macro and the handling of error-terminated bus cycles.

### 19.4.1 Access evaluation macro

The basic operation of the MPU is performed in the access evaluation macro, a hardware structure replicated in the two-dimensional connection matrix. As shown in the following figure, the access evaluation macro inputs the crossbar bus address phase signals and the contents of a region descriptor (RGD*n*) and performs two major functions:

- Region hit determination
- Detection of an access protection violation

The following figure shows a functional block diagram.

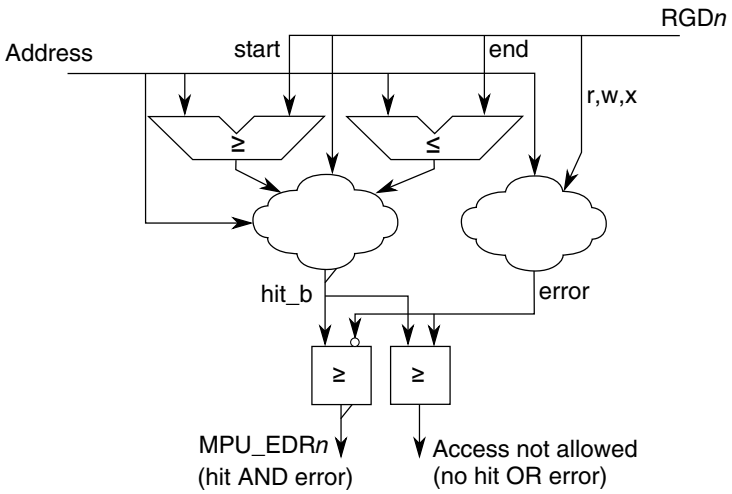



**Figure 19-80. MPU access evaluation macro**

### 19.4.1.1 Hit determination

To determine whether the current reference hits in the given region, two magnitude comparators are used with the region's start and end addresses. The boolean equation for this portion of the hit determination is:

```
region_hit = ((addr[31:5] >= RGDn_Word0[SRTADDR]) & (addr[31:5] <= RGDn_Word1[ENDADDR])) &
RGDn_Word3[VLD]
```

where addr is the current reference address, RGD*n*_Word0[SRTADDR] and RGD*n*_Word1[ENDADDR] are the start and end addresses, and RGD*n*_Word3[VLD] is the valid bit.

**NOTE**

The MPU does not verify that ENDADDR ≥ SRTADDR.

In addition to the comparison of the reference address versus the region descriptor's start and end addresses, the optional process identifier is examined against the region descriptor's PID and PIDMASK fields. A process identifier hit term is formed as follows:

```
pid_hit = ~RGDn_Word2[MxPE] |
          ((current_pid |
          RGDn_Word3[PIDMASK]) == (RGDn_Word3[PID] | RGDn_Word3[PIDMASK]))
```

where the current_pid is the selected process identifier from the current bus master, and RGD*n*_Word3[PID] and RGD*n*_Word3[PIDMASK] are the process identifier fields from region descriptor *n*. For bus masters that do not output a process identifier, the MPU forces the pid_hit term to assert.

### 19.4.1.2 Privilege violation determination

While the access evaluation macro is determining region hit, the logic is also evaluating if the current access is allowed by the permissions defined in the region descriptor. Using the master and supervisor/user mode signals, a set of effective permissions is generated from the appropriate fields in the region descriptor. The protection violation logic then evaluates the access against the effective permissions using the specification shown below.

**Table 19-80. Protection violation definition**

| Description | MxUM | | | Protection violation? |
|---|---|---|---|---|
| | r | w | x | |
| Instruction fetch read | — | — | 0 | Yes, no execute permission |
| | — | — | 1 | No, access is allowed |
| Data read | 0 | — | — | Yes, no read permission |
| | 1 | — | — | No, access is allowed |
| Data write | — | 0 | — | Yes, no write permission |
| | — | 1 | — | No, access is allowed |

## 19.4.2 Putting it all together and error terminations

For each slave port monitored, the MPU performs a reduction-AND of all the individual terms from each access evaluation macro. This expression then terminates the bus cycle with an error and reports a protection error for three conditions:

- If the access does not hit in any region descriptor, a protection error is reported.
- If the access hits in a single region descriptor and that region signals a protection violation, a protection error is reported.
- If the access hits in multiple (overlapping) regions and all regions signal protection violations, a protection error is reported.

As shown in the third condition, granting permission is a higher priority than denying access for overlapping regions. This approach is more flexible to system software in region descriptor assignments. For an example of the use of overlapping region descriptors, see Application information.

### 19.4.3 Power management

Disabling the MPU by clearing CESR[VLD] minimizes power dissipation. To minimize the power dissipation of an enabled MPU, invalidate unused region descriptors by clearing the associated RGDn_Word3[VLD] bits.

## 19.5 Initialization information

At system startup, load the appropriate number of region descriptors, including setting RGD*n*_Word3[VLD]. Setting CESR[VLD] enables the module.

If the system requires that all the loaded region descriptors be enabled simultaneously, first ensure that the entire MPU is disabled (CESR[VLD]=0).

**Note**

A region descriptor must be set to allow access to the MPU registers if further changes are needed.

## 19.6 Application information

In an operational system, interfacing with the MPU is generally classified into the following activities:

- Creating a new memory region—Load the appropriate region descriptor into an available RGD*n*, using four sequential 32-bit writes. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes. (Clearing RGD*n*_Word3[VLD] deletes/removes an existing memory region.)
- Altering only access privileges—To not affect the valid bit, write to the alternate version of the access control word (RGDAAC*n*), so there are no coherency issues involved with the update. When the write completes, the memory region's access rights switch instantaneously to the new value.
- Changing a region's start and end addresses—Write a minimum of three words to the region descriptor (RGD*n*_Word{0,1,3}). Word 0 and 1 redefine the start and end addresses, respectively. Word 3 re-enables the region descriptor valid bit. In most situations, all four words of the region descriptor are rewritten.
- Accessing the MPU—Allocate a region descriptor to restrict MPU access to supervisor mode from a specific master.

- Detecting an access error—The current bus cycle is terminated with an error response and EAR*n* and EDR*n* capture information on the faulting reference. The error-terminated bus cycle typically initiates an error response in the originating bus master. For example, a processor core may respond with a bus error exception, while a data movement bus master may respond with an error interrupt. The processor can retrieve the captured error address and detail information simply by reading E{A,D}R*n*. CESR[SPERR] signals which error registers contain captured fault data.
- Overlapping region descriptors—Applying overlapping regions often reduces the number of descriptors required for a given set of access controls. In the overlapping memory space, the protection rights of the corresponding region descriptors are logically summed together (the boolean OR operator).

  The following dual-core system example contains four bus masters:

  - The two processors: CP0, CP1
  - Two DMA engines: DMA1, a traditional data movement engine transferring data between RAM and peripherals and DMA2, a second engine transferring data to/from the RAM only

  Consider the following region descriptor assignments:

**Table 19-81. Overlapping region descriptor example**

| Region description | RGDn | | CP0 | CP1 | DMA1 | DMA2 | |
|---|---|---|---|---|---|---|---|
| CP0 code | 0 | | rwx | r-- | — | — | Flash |
| CP1 code | 1 | | r-- | rwx | — | — | |
| CP0 data & stack | 2 | | rw- | — | — | — | RAM |
| CP0 → CP1 shared data | 2 | 3 | r-- | r-- | — | — | |
| CP1 → CP0 shared data | 4 | | | | | | |
| CP1 data & stack | 4 | | — | rw- | — | — | |
| Shared DMA data | 5 | | rw- | rw- | rw | rw | |
| MPU | 6 | | rw- | rw- | — | — | Peripheral space |
| Peripherals | 7 | | rw- | rw- | rw | — | |

In this example, there are eight descriptors used to span nine regions in the three main spaces of the system memory map: flash, RAM, and peripheral space. Each region indicates the specific permissions for each of the four bus masters and this definition provides an appropriate set of shared, private and executable memory spaces.

Of particular interest are the two overlapping spaces: region descriptors 2 & 3 and 3 & 4.

The space defined by RGD2 with no overlap is a private data and stack area that provides read/write access to CP0 only. The overlapping space between RGD2 and RGD3 defines a shared data space for passing data from CP0 to CP1 and the access controls are defined by the logical OR of the two region descriptors. Thus, CP0 has (rw- | r--) = (rw-) permissions, while CP1 has (--- | r--) = (r--) permission in this space. Both DMA engines are excluded from this shared processor data region. The overlapping spaces between RGD3 and RGD4 defines another shared data space, this one for passing data from CP1 to CP0. For this overlapping space, CP0 has (r-- | ---) = (r--) permission, while CP1 has (rw- | r--) = (rw-) permission. The non-overlapped space of RGD4 defines a private data and stack area for CP1 only.

The space defined by RGD5 is a shared data region, accessible by all four bus masters. Finally, the slave peripheral space mapped onto the IPS bus is partitioned into two regions:

- One containing the MPU's programming model accessible only to the two processor cores
- The remaining peripheral region accessible to both processors and the traditional DMA1 master

This example shows one possible application of the capabilities of the MPU in a typical system.

# Chapter 20
# Peripheral Bridge (AIPS-Lite)

## 20.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The peripheral bridge converts the crossbar switch interface to an interface that can access most of the slave peripherals on this chip.

The peripheral bridge occupies 64 MB of the address space, which is divided into 4-KB peripheral slots. (Not all peripheral slots might be used. See Memory map for details on slot assignments.) The bridge includes separate clock enable inputs for each of the slots to accommodate slower peripherals.

### 20.1.1 Features

Key features of the peripheral bridge are:

- Supports peripheral slots with 8-, 16-, and 32-bit datapath width

### 20.1.2 General operation

The slave devices connected to the peripheral bridge are modules which contain a programming model of control and status registers. The system masters read and write these registers through the peripheral bridge. The peripheral bridge performs a bus protocol conversion of the master transactions and generates the following as inputs to the peripherals:

- Module enables
- Module addresses

- Transfer attributes
- Byte enables
- Write data

The peripheral bridge selects and captures read data from the peripheral interface and returns it to the crossbar switch.

The register maps of the peripherals are located on 4-KB boundaries. Each peripheral is allocated one or more 4-KB block(s) of the memory map.

The AIPS-Lite module uses the accessed peripheral's data width to perform proper data byte lane routing; no bus decomposition (bus sizing) is performed.

## 20.2 Functional description

The peripheral bridge functions as a bus protocol translator between the crossbar switch and the slave peripheral bus.

The peripheral bridge manages all transactions destined for the attached slave devices and generates select signals for modules on the peripheral bus by decoding accesses within the attached address space.

### 20.2.1 Access support

Aligned and misaligned 32-bit, 16-bit, and byte accesses are supported for 32-bit peripherals. Misaligned accesses are supported to allow memory to be placed on the slave peripheral bus. Peripheral registers must not be misaligned, although no explicit checking is performed by the peripheral bridge. All accesses are performed with a single transfer.

All accesses to the peripheral slots must be sized less than or equal to the designated peripheral slot size. If an access is attempted that is larger than the targeted port, an error response is generated.

# Chapter 21
# Direct Memory Access Multiplexer (DMAMUX)

## 21.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

### 21.1.1 Overview

The direct memory access multiplexer (DMAMUX) routes DMA sources, called slots, to any of the 16 DMA channels. This process is illustrated in the following figure.

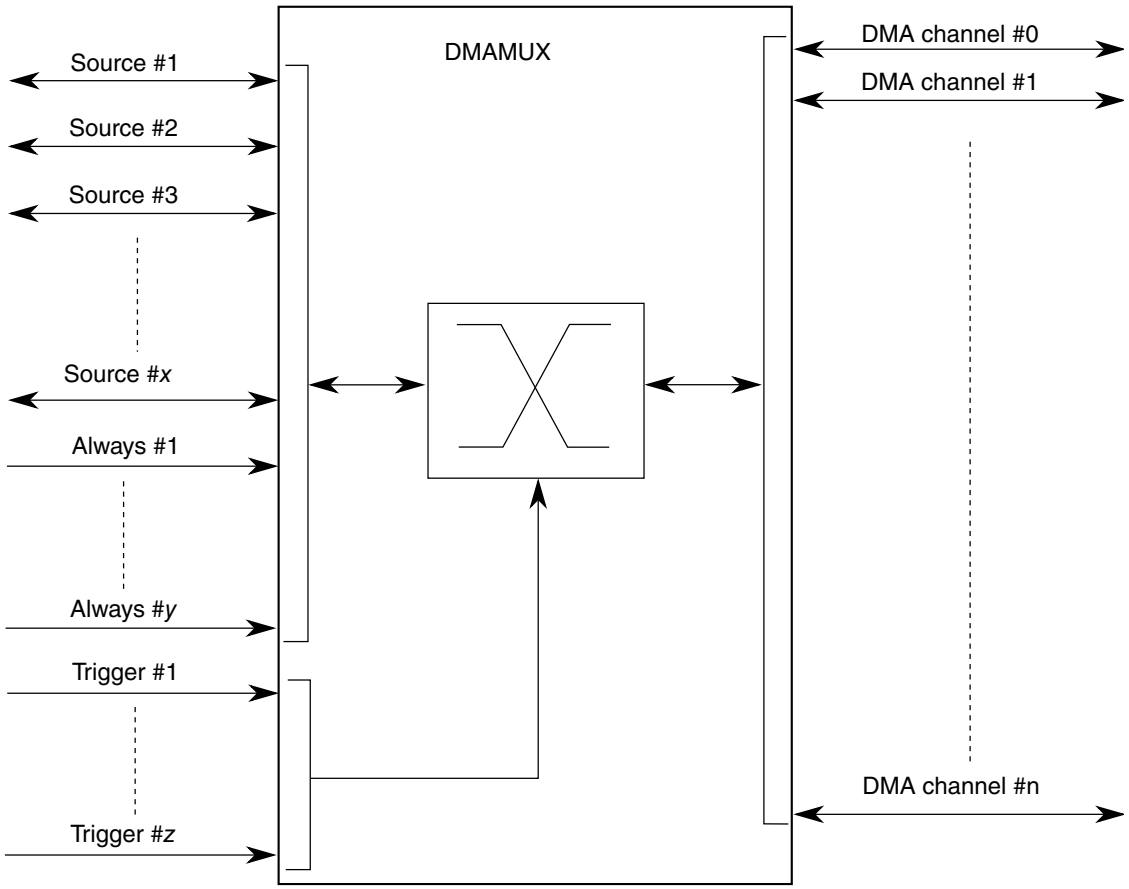



**Figure 21-1. DMAMUX block diagram**

## 21.1.2 Features

The DMAMUX module provides these features:

- Up to 52 peripheral slots and up to 10 always-on slots can be routed to 16 channels.
- 16 independently selectable DMA channel routers.
  - The first 4 channels additionally provide a trigger functionality.
- Each channel router can be assigned to one of the 52 possible peripheral DMA slots or to one of the 10 always-on slots.

## 21.1.3 Modes of operation

The following operating modes are available:

- Disabled mode

In this mode, the DMA channel is disabled. Because disabling and enabling of DMA channels is done primarily via the DMA configuration registers, this mode is used mainly as the reset state for a DMA channel in the DMA channel MUX. It may also be used to temporarily suspend a DMA channel while reconfiguration of the system takes place, for example, changing the period of a DMA trigger.

- Normal mode

  In this mode, a DMA source is routed directly to the specified DMA channel. The operation of the DMAMUX in this mode is completely transparent to the system.

- Periodic Trigger mode

  In this mode, a DMA source may only request a DMA transfer, such as when a transmit buffer becomes empty or a receive buffer becomes full, periodically. Configuration of the period is done in the registers of the periodic interrupt timer (PIT). This mode is available only for channels 0–3.

## 21.2 External signal description

The DMAMUX has no external pins.

## 21.3 Memory map/register definition

This section provides a detailed description of all memory-mapped registers in the DMAMUX.

**DMAMUX memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4002_1000 | Channel Configuration register (DMAMUX_CHCFG0) | 8 | R/W | 00h | 21.3.1/416 |
| 4002_1001 | Channel Configuration register (DMAMUX_CHCFG1) | 8 | R/W | 00h | 21.3.1/416 |
| 4002_1002 | Channel Configuration register (DMAMUX_CHCFG2) | 8 | R/W | 00h | 21.3.1/416 |
| 4002_1003 | Channel Configuration register (DMAMUX_CHCFG3) | 8 | R/W | 00h | 21.3.1/416 |
| 4002_1004 | Channel Configuration register (DMAMUX_CHCFG4) | 8 | R/W | 00h | 21.3.1/416 |
| 4002_1005 | Channel Configuration register (DMAMUX_CHCFG5) | 8 | R/W | 00h | 21.3.1/416 |
| 4002_1006 | Channel Configuration register (DMAMUX_CHCFG6) | 8 | R/W | 00h | 21.3.1/416 |
| 4002_1007 | Channel Configuration register (DMAMUX_CHCFG7) | 8 | R/W | 00h | 21.3.1/416 |
| 4002_1008 | Channel Configuration register (DMAMUX_CHCFG8) | 8 | R/W | 00h | 21.3.1/416 |

*Table continues on the next page...*

**DMAMUX memory map (continued)**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4002_1009 | Channel Configuration register (DMAMUX_CHCFG9) | 8 | R/W | 00h | 21.3.1/416 |
| 4002_100A | Channel Configuration register (DMAMUX_CHCFG10) | 8 | R/W | 00h | 21.3.1/416 |
| 4002_100B | Channel Configuration register (DMAMUX_CHCFG11) | 8 | R/W | 00h | 21.3.1/416 |
| 4002_100C | Channel Configuration register (DMAMUX_CHCFG12) | 8 | R/W | 00h | 21.3.1/416 |
| 4002_100D | Channel Configuration register (DMAMUX_CHCFG13) | 8 | R/W | 00h | 21.3.1/416 |
| 4002_100E | Channel Configuration register (DMAMUX_CHCFG14) | 8 | R/W | 00h | 21.3.1/416 |
| 4002_100F | Channel Configuration register (DMAMUX_CHCFG15) | 8 | R/W | 00h | 21.3.1/416 |

## 21.3.1 Channel Configuration register (DMAMUX_CHCFG*n*)

Each of the DMA channels can be independently enabled/disabled and associated with one of the DMA slots (peripheral slots or always-on slots) in the system.

**NOTE**

Setting multiple CHCFG registers with the same source value will result in unpredictable behavior.

Before changing the trigger or source settings, a DMA channel must be disabled via the CHCFGn[ENBL] bit.

Address: 4002_1000h base + 0h offset + (1d × i), where i=0d to 15d

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | ENBL | TRIG | SOURCE | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMAMUX_CHCFG*n* field descriptions**

| Field | Description |
|---|---|
| 7<br>ENBL | DMA Channel Enable<br><br>Enables the DMA channel.<br><br>0 DMA channel is disabled. This mode is primarily used during configuration of the DMAMux. The DMA has separate channel enables/disables, which should be used to disable or reconfigure a DMA channel.<br>1 DMA channel is enabled |
| 6<br>TRIG | DMA Channel Trigger Enable<br><br>Enables the periodic trigger capability for the triggered DMA channel. |

*Table continues on the next page...*

**DMAMUX_CHCFG*n* field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Triggering is disabled. If triggering is disabled and the ENBL bit is set, the DMA Channel will simply route the specified source to the DMA channel. (Normal mode)<br>1 Triggering is enabled. If triggering is enabled and the ENBL bit is set, the DMAMUX is in Periodic Trigger mode. |
| 5–0<br>SOURCE | DMA Channel Source (Slot)<br><br>Specifies which DMA source, if any, is routed to a particular DMA channel. See your device's chip configuration details for information about the peripherals and their slot numbers. |

## 21.4 Functional description

The primary purpose of the DMAMUX is to provide flexibility in the system's use of the available DMA channels. As such, configuration of the DMAMUX is intended to be a static procedure done during execution of the system boot code. However, if the procedure outlined in Enabling and configuring sources is followed, the configuration of the DMAMUX may be changed during the normal operation of the system.

Functionally, the DMAMUX channels may be divided into two classes:

- Channels that implement the normal routing functionality plus periodic triggering capability
- Channels that implement only the normal routing functionality

### 21.4.1 DMA channels with periodic triggering capability

Besides the normal routing functionality, the first 4 channels of the DMAMUX provide a special periodic triggering capability that can be used to provide an automatic mechanism to transmit bytes, frames, or packets at fixed intervals without the need for processor intervention. The trigger is generated by the periodic interrupt timer (PIT); as such, the configuration of the periodic triggering interval is done via configuration registers in the PIT. See the section on periodic interrupt timer for more information on this topic.

**Note**

Because of the dynamic nature of the system (due to DMA channel priorities, bus arbitration, interrupt service routine lengths, etc.), the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

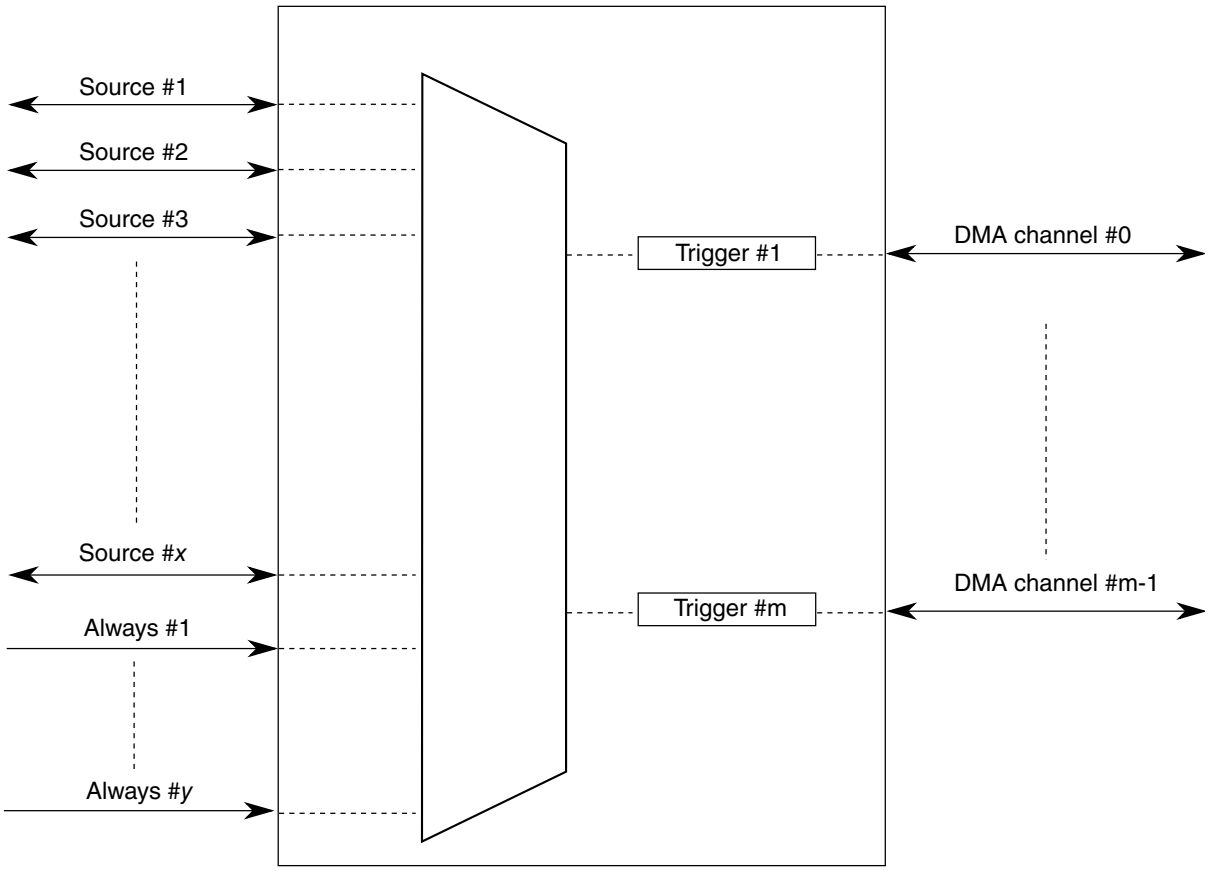



**Figure 21-19. DMAMUX triggered channels**

The DMA channel triggering capability allows the system to schedule regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. This trigger works by gating the request from the peripheral to the DMA until a trigger event has been seen. This is illustrated in the following figure.

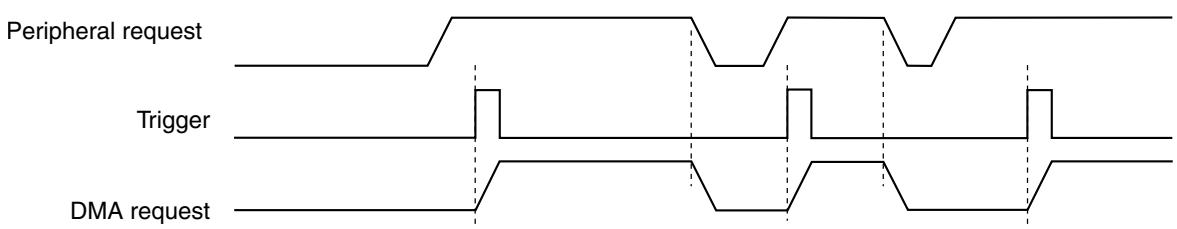



**Figure 21-20. DMAMUX channel triggering: normal operation**

After the DMA request has been serviced, the peripheral will negate its request, effectively resetting the gating mechanism until the peripheral reasserts its request and the next trigger event is seen. This means that if a trigger is seen, but the peripheral is not requesting a transfer, then that trigger will be ignored. This situation is illustrated in the following figure.

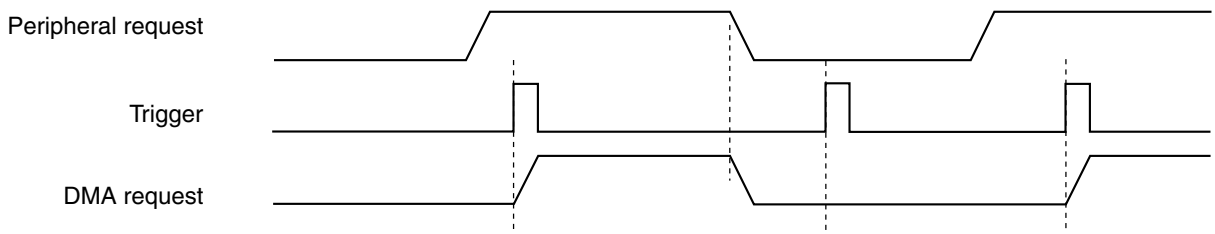



**Figure 21-21. DMAMUX channel triggering: ignored trigger**

This triggering capability may be used with any peripheral that supports DMA transfers, and is most useful for two types of situations:

- Periodically polling external devices on a particular bus

  As an example, the transmit side of an SPI is assigned to a DMA channel with a trigger, as described above. After it has been set up, the SPI will request DMA transfers, presumably from memory, as long as its transmit buffer is empty. By using a trigger on this channel, the SPI transfers can be automatically performed every 5 µs (as an example). On the receive side of the SPI, the SPI and DMA can be configured to transfer receive data into memory, effectively implementing a method to periodically read data from external devices and transfer the results into memory without processor intervention.

- Using the GPIO ports to drive or sample waveforms

  By configuring the DMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in on-chip memory. Conversely, using the DMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in tabular form in on-chip memory.

A more detailed description of the capability of each trigger, including resolution, range of values, and so on, may be found in the periodic interrupt timer section.

## 21.4.2 DMA channels with no triggering capability

The other channels of the DMAMUX provide the normal routing functionality as described in Modes of operation.

### 21.4.3 Always-enabled DMA sources

In addition to the peripherals that can be used as DMA sources, there are 10 additional DMA sources that are always enabled. Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the sources that are always enabled provide no such "throttling" of the data transfers. These sources are most useful in the following cases:

- Performing DMA transfers to/from GPIO—Moving data from/to one or more GPIO pins, either unthrottled (that is, as fast as possible), or periodically (using the DMA triggering capability).
- Performing DMA transfers from memory to memory—Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- Performing DMA transfers from memory to the external bus, or vice-versa—Similar to memory to memory transfers, this is typically done as quickly as possible.
- Any DMA transfer that requires software activation—Any DMA transfer that should be explicitly started by software.

In cases where software should initiate the start of a DMA transfer, an always-enabled DMA source can be used to provide maximum flexibility. When activating a DMA channel via software, subsequent executions of the minor loop require that a new start event be sent. This can either be a new software activation, or a transfer request from the DMA channel MUX. The options for doing this are:

- Transfer all data in a single minor loop.

  By configuring the DMA to transfer all of the data in a single minor loop (that is, major loop counter = 1), no reactivation of the channel is necessary. The disadvantage to this option is the reduced granularity in determining the load that the DMA transfer will impose on the system. For this option, the DMA channel must be disabled in the DMA channel MUX.

- Use explicit software reactivation.

  In this option, the DMA is configured to transfer the data using both minor and major loops, but the processor is required to reactivate the channel by writing to the DMA registers *after every minor loop*. For this option, the DMA channel must be disabled in the DMA channel MUX.

- Use an always-enabled DMA source.

In this option, the DMA is configured to transfer the data using both minor and major loops, and the DMA channel MUX does the channel reactivation. For this option, the DMA channel should be enabled and pointing to an "always enabled" source. Note that the reactivation of the channel can be continuous (DMA triggering is disabled) or can use the DMA triggering capability. In this manner, it is possible to execute periodic transfers of packets of data from one source to another, without processor intervention.

## 21.5 Initialization/application information

This section provides instructions for initializing the DMA channel MUX.

### 21.5.1 Reset

The reset state of each individual bit is shown in Memory map/register definition. In summary, after reset, all channels are disabled and must be explicitly enabled before use.

### 21.5.2 Enabling and configuring sources

To enable a source with periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 4 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Configure the corresponding timer.
5. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

**NOTE**

The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

To configure source #5 transmit for use with DMA channel 1, with periodic triggering capability:

1. Write 0x00 to CHCFG1 (base address + 0x01).

2. Configure channel 1 in the DMA, including enabling the channel.
3. Configure a timer for the desired trigger interval.
4. Write 0xC5 to CHCFG1 (base address + 0x01).

The following code example illustrates steps 1 and 4 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0xFC084000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCONFIG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCONFIG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCONFIG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCONFIG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCONFIG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCONFIG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCONFIG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCONFIG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCONFIG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCONFIG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCONFIG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCONFIG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCONFIG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCONFIG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCONFIG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCONFIG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
:
:
*CHCONFIG1 = 0x00;
*CHCONFIG1 = 0xC5;
```

To enable a source without periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 4 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that CHCFG[ENBL] is set while CHCFG[TRIG] is cleared.

**NOTE**

The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

To configure source #5 transmit for use with DMA channel 1, with no periodic triggering capability:

1. Write 0x00 to CHCFG1 (base address + 0x01).
2. Configure channel 1 in the DMA, including enabling the channel.
3. Write 0x85 to CHCFG1 (base address + 0x01).

The following code example illustrates steps 1 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0xFC084000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCONFIG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCONFIG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCONFIG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCONFIG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCONFIG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCONFIG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCONFIG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCONFIG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCONFIG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCONFIG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCONFIG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCONFIG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCONFIG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCONFIG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCONFIG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCONFIG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
:
:
*CHCONFIG1 = 0x00;
*CHCONFIG1 = 0x85;
```

To disable a source:

A particular DMA source may be disabled by not writing the corresponding source value into any of the CHCFG registers. Additionally, some module-specific configuration may be necessary. See the appropriate section for more details.

To switch the source of a DMA channel:

1. Disable the DMA channel in the DMA and reconfigure the channel for the new source.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] bits of the DMA channel.
3. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

To switch DMA channel 8 from source #5 transmit to source #7 transmit:

1. In the DMA configuration registers, disable DMA channel 8 and reconfigure it to handle the transfers to peripheral slot 7. This example assumes channel 8 doesn't have triggering capability.
2. Write 0x00 to CHCFG8 (base address + 0x08).
3. Write 0x87 to CHCFG8 (base address + 0x08). (In this example, setting CHCFG[TRIG] would have no effect due to the assumption that channel 8 does not support the periodic triggering functionality).

The following code example illustrates steps 2 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0xFC084000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCONFIG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCONFIG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCONFIG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCONFIG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCONFIG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCONFIG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCONFIG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCONFIG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCONFIG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCONFIG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCONFIG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCONFIG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCONFIG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCONFIG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCONFIG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCONFIG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
:
:
*CHCONFIG8 = 0x00;
*CHCONFIG8 = 0x87;
```

# Chapter 22
# Direct Memory Access Controller (eDMA)

## 22.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes:

- A DMA engine that performs:
  - Source- and destination-address calculations
  - Data-movement operations
- Local memory containing transfer control descriptors for each of the 16 channels

### 22.1.1 Block diagram

This diagram illustrates the eDMA module.

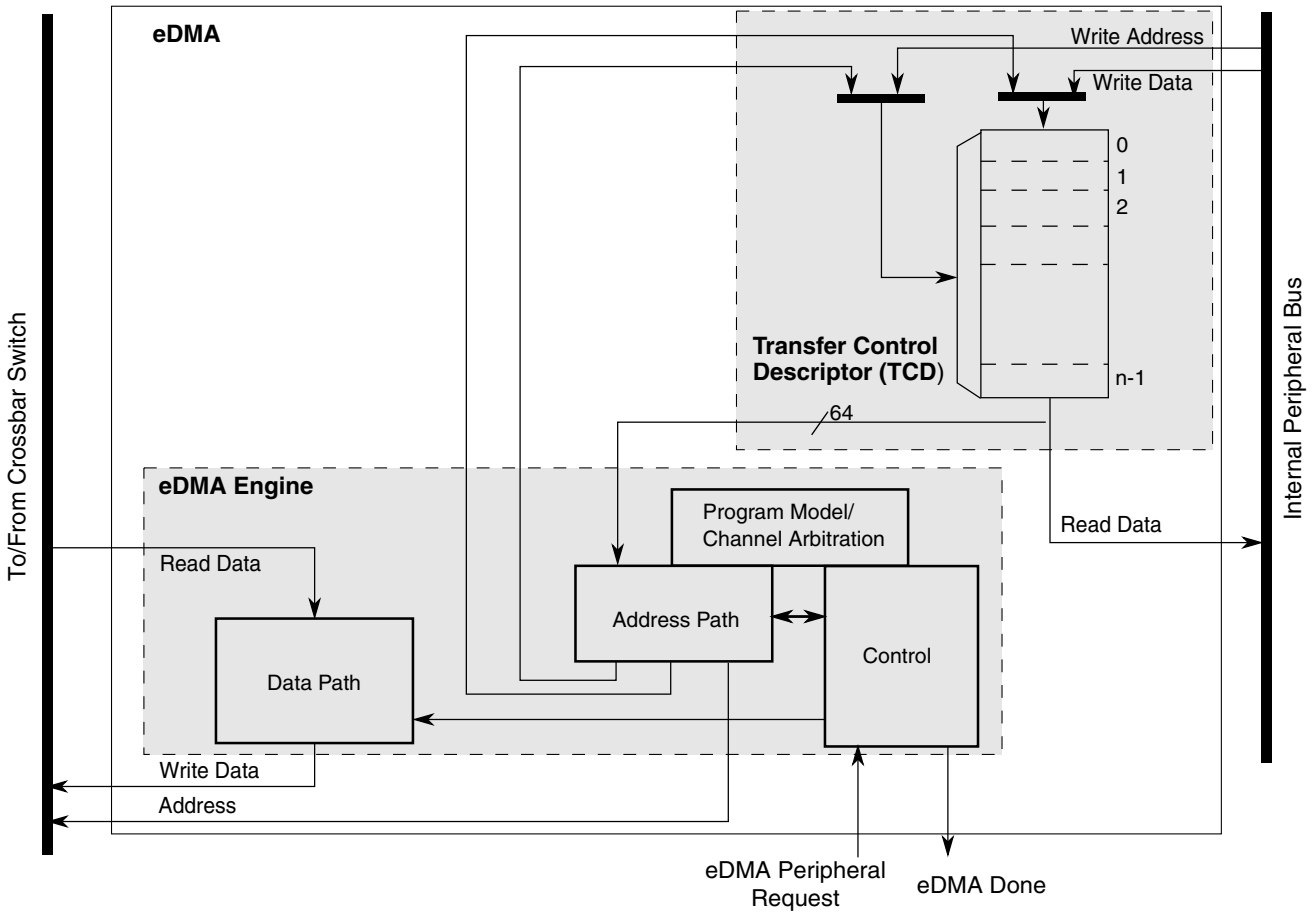



**Figure 22-1. eDMA block diagram**

## 22.1.2 Block parts

The eDMA module is partitioned into two major modules: the eDMA engine and the transfer-control descriptor local memory.

The eDMA engine is further partitioned into four submodules:

**Table 22-1. eDMA engine submodules**

| Submodule | Function |
|---|---|
| Address path | This block implements registered versions of two channel transfer control descriptors, channel x and channel y, and manages all master bus-address calculations. All the channels provide the same functionality. This structure allows data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel activation is asserted while the first channel is active. After a channel is activated, it runs until the minor loop is completed, unless preempted by a higher priority channel. This provides a mechanism (enabled by DCHPRI*n*[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution.<br><br>When any channel is selected to execute, the contents of its TCD are read from local memory and loaded into the address path channel x registers for a normal start and into channel y registers for a preemption start. After the minor loop completes execution, the address path hardware writes the new values for the TCD*n*_{SADDR, DADDR, CITER} back to local memory. If the major iteration count is exhausted, additional processing is performed, including the final address pointer updates, reloading the TCD*n*_CITER field, and a possible fetch of the next TCD*n* from memory as part of a scatter/gather operation. |
| Data path | This block implements the bus master read/write datapath. It includes 16 bytes of register storage and the necessary multiplex logic to support any required data alignment. The internal read data bus is the primary input, and the internal write data bus is the primary output.<br><br>The address and data path modules directly support the 2-stage pipelined internal bus. The address path module represents the 1st stage of the bus pipeline (address phase), while the data path module implements the 2nd stage of the pipeline (data phase). |
| Program model/channel arbitration | This block implements the first section of the eDMA programming model as well as the channel arbitration logic. The programming model registers are connected to the internal peripheral bus. The eDMA peripheral request inputs and interrupt request outputs are also connected to this block (via control logic). |
| Control | This block provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read/ destination write operations until the number of bytes specified in the minor loop byte count has moved. For descriptors where the sizes are not equal, multiple accesses of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-bit data and the destination is 32-bit data, two reads are performed, then one 32-bit write. |

The transfer-control descriptor local memory is further partitioned into:

**Table 22-2. Transfer control descriptor memory**

| Submodule | Description |
|---|---|
| Memory controller | This logic implements the required dual-ported controller, managing accesses from the eDMA engine as well as references from the internal peripheral bus. As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the peripheral transaction is stalled. |
| Memory array | TCD storage for each channel's transfer profile. |

### 22.1.3 Features

The eDMA is a highly programmable data-transfer engine optimized to minimize the required intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known and not defined within the data packet itself. The eDMA module features:

- All data movement via dual-address transfers: read from source, write to destination
  - Programmable source and destination addresses and transfer size
  - Support for enhanced addressing modes
- 16-channel implementation that performs complex data transfers with minimal intervention from a host processor
  - Internal data buffer, used as temporary storage to support 16- and 32-byte transfers
  - Connections to the crossbar switch for bus mastering the data movement
- Transfer control descriptor (TCD) organized to support two-deep, nested transfer operations
  - 32-byte TCD stored in local memory for each channel
  - An inner data transfer loop defined by a minor byte transfer count
  - An outer data transfer loop defined by a major iteration count
- Channel activation via one of three methods:
  - Explicit software initiation
  - Initiation via a channel-to-channel linking mechanism for continuous transfers
  - Peripheral-paced hardware requests, one per channel
- Fixed-priority and round-robin channel arbitration
- Channel completion reported via optional interrupt requests
  - One interrupt per channel, optionally asserted at completion of major iteration count
  - Optional error terminations per channel and logically summed together to form one error interrupt to the interrupt controller
- Optional support for scatter/gather DMA processing

- Support for complex data structures
- Support to cancel transfers via software

In the discussion of this module, *n* is used to reference the channel number.

## 22.2 Modes of operation

The eDMA operates in the following modes:

**Table 22-3. Modes of operation**

| Mode | Description |
|---|---|
| Normal | In Normal mode, the eDMA transfers data between a source and a destination. The source and destination can be a memory block or an I/O block capable of operation with the eDMA.<br>A service request initiates a transfer of a specific number of bytes (NBYTES) as specified in the transfer control descriptor (TCD). The minor loop is the sequence of read-write operations that transfers these NBYTES per service request. Each service request executes one iteration of the major loop, which transfers NBYTES of data. |
| Debug | DMA operation is configurable in Debug mode via the control register:<br>• If CR[EDBG] is cleared, the DMA continues to operate.<br>• If CR[EDBG] is set, the eDMA stops transferring data. If Debug mode is entered while a channel is active, the eDMA continues operation until the channel retires. |
| Wait | Before entering Wait mode, the DMA attempts to complete its current transfer. After the transfer completes, the device enters Wait mode. |

## 22.3 Memory map/register definition

The eDMA's programming model is partitioned into two regions:

- The first region defines a number of registers providing control functions
- The second region corresponds to the local transfer control descriptor (TCD) memory

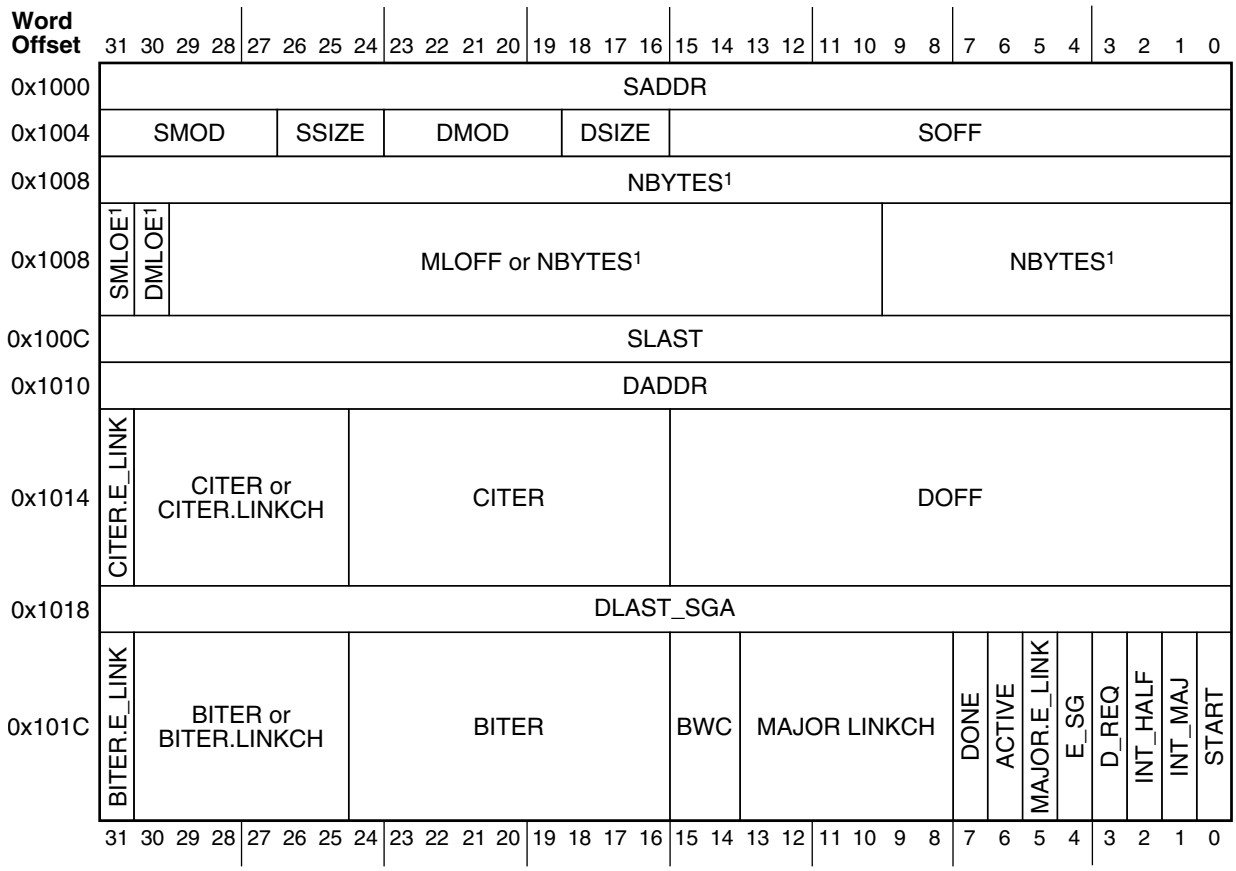
  Each channel requires a 32-byte transfer control descriptor for defining the desired data movement operation. The channel descriptors are stored in the local memory in sequential order: channel 0, channel 1,... channel 15 . Each TCD*n* definition is presented as 11 registers of 16 or 32 bits.

  TCD Initialization: Prior to activating a channel, you must initialize its TCD with the appropriate transfer profile.

Here is the TCD structure:

**DMA Basics: TCD Structure**

- One DMA engine has a number of channels to react to DMA requests
- Each channel has its own TCD




1 The fields implemented in Word 2 depend on whether DMA_CR[EMLM] is set to 0 or 1.

Reserved memory and bit fields:

- Reading reserved bits in a register returns the value of zero.
- Writes to reserved bits in a register are ignored.
- Reading or writing a reserved memory location generates a bus error.

## DMA memory map

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4000_8000 | Control Register (DMA_CR) | 32 | R/W | 0000_0000h | 22.3.1/441 |
| 4000_8004 | Error Status Register (DMA_ES) | 32 | R | 0000_0000h | 22.3.2/444 |
| 4000_800C | Enable Request Register (DMA_ERQ) | 32 | R/W | 0000_0000h | 22.3.3/446 |
| 4000_8014 | Enable Error Interrupt Register (DMA_EEI) | 32 | R/W | 0000_0000h | 22.3.4/448 |
| 4000_8018 | Clear Enable Error Interrupt Register (DMA_CEEI) | 8 | W (always reads 0) | 00h | 22.3.5/450 |

*Table continues on the next page...*

## DMA memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4000_8019 | Set Enable Error Interrupt Register (DMA_SEEI) | 8 | W (always reads 0) | 00h | 22.3.6/451 |
| 4000_801A | Clear Enable Request Register (DMA_CERQ) | 8 | W (always reads 0) | 00h | 22.3.7/452 |
| 4000_801B | Set Enable Request Register (DMA_SERQ) | 8 | W (always reads 0) | 00h | 22.3.8/453 |
| 4000_801C | Clear DONE Status Bit Register (DMA_CDNE) | 8 | W (always reads 0) | 00h | 22.3.9/454 |
| 4000_801D | Set START Bit Register (DMA_SSRT) | 8 | W (always reads 0) | 00h | 22.3.10/455 |
| 4000_801E | Clear Error Register (DMA_CERR) | 8 | W (always reads 0) | 00h | 22.3.11/456 |
| 4000_801F | Clear Interrupt Request Register (DMA_CINT) | 8 | W (always reads 0) | 00h | 22.3.12/457 |
| 4000_8024 | Interrupt Request Register (DMA_INT) | 32 | R/W | 0000_0000h | 22.3.13/458 |
| 4000_802C | Error Register (DMA_ERR) | 32 | R/W | 0000_0000h | 22.3.14/460 |
| 4000_8034 | Hardware Request Status Register (DMA_HRS) | 32 | R/W | 0000_0000h | 22.3.15/463 |
| 4000_8100 | Channel n Priority Register (DMA_DCHPRI3) | 8 | R/W | See section | 22.3.16/465 |
| 4000_8101 | Channel n Priority Register (DMA_DCHPRI2) | 8 | R/W | See section | 22.3.16/465 |
| 4000_8102 | Channel n Priority Register (DMA_DCHPRI1) | 8 | R/W | See section | 22.3.16/465 |
| 4000_8103 | Channel n Priority Register (DMA_DCHPRI0) | 8 | R/W | See section | 22.3.16/465 |
| 4000_8104 | Channel n Priority Register (DMA_DCHPRI7) | 8 | R/W | See section | 22.3.16/465 |
| 4000_8105 | Channel n Priority Register (DMA_DCHPRI6) | 8 | R/W | See section | 22.3.16/465 |
| 4000_8106 | Channel n Priority Register (DMA_DCHPRI5) | 8 | R/W | See section | 22.3.16/465 |
| 4000_8107 | Channel n Priority Register (DMA_DCHPRI4) | 8 | R/W | See section | 22.3.16/465 |
| 4000_8108 | Channel n Priority Register (DMA_DCHPRI11) | 8 | R/W | See section | 22.3.16/465 |
| 4000_8109 | Channel n Priority Register (DMA_DCHPRI10) | 8 | R/W | See section | 22.3.16/465 |
| 4000_810A | Channel n Priority Register (DMA_DCHPRI9) | 8 | R/W | See section | 22.3.16/465 |
| 4000_810B | Channel n Priority Register (DMA_DCHPRI8) | 8 | R/W | See section | 22.3.16/465 |
| 4000_810C | Channel n Priority Register (DMA_DCHPRI15) | 8 | R/W | See section | 22.3.16/465 |
| 4000_810D | Channel n Priority Register (DMA_DCHPRI14) | 8 | R/W | See section | 22.3.16/465 |
| 4000_810E | Channel n Priority Register (DMA_DCHPRI13) | 8 | R/W | See section | 22.3.16/465 |
| 4000_810F | Channel n Priority Register (DMA_DCHPRI12) | 8 | R/W | See section | 22.3.16/465 |
| 4000_9000 | TCD Source Address (DMA_TCD0_SADDR) | 32 | R/W | Undefined | 22.3.17/466 |
| 4000_9004 | TCD Signed Source Address Offset (DMA_TCD0_SOFF) | 16 | R/W | Undefined | 22.3.18/466 |

*Table continues on the next page...*

## DMA memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4000_9006 | TCD Transfer Attributes (DMA_TCD0_ATTR) | 16 | R/W | Undefined | 22.3.19/467 |
| 4000_9008 | TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD0_NBYTES_MLNO) | 32 | R/W | Undefined | 22.3.20/468 |
| 4000_9008 | TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD0_NBYTES_MLOFFNO) | 32 | R/W | Undefined | 22.3.21/468 |
| 4000_9008 | TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD0_NBYTES_MLOFFYES) | 32 | R/W | Undefined | 22.3.22/470 |
| 4000_900C | TCD Last Source Address Adjustment (DMA_TCD0_SLAST) | 32 | R/W | Undefined | 22.3.23/471 |
| 4000_9010 | TCD Destination Address (DMA_TCD0_DADDR) | 32 | R/W | Undefined | 22.3.24/471 |
| 4000_9014 | TCD Signed Destination Address Offset (DMA_TCD0_DOFF) | 16 | R/W | Undefined | 22.3.25/472 |
| 4000_9016 | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD0_CITER_ELINKYES) | 16 | R/W | Undefined | 22.3.26/472 |
| 4000_9016 | DMA_TCD0_CITER_ELINKNO | 16 | R/W | Undefined | 22.3.27/474 |
| 4000_9018 | TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD0_DLASTSGA) | 32 | R/W | Undefined | 22.3.28/475 |
| 4000_901C | TCD Control and Status (DMA_TCD0_CSR) | 16 | R/W | Undefined | 22.3.29/475 |
| 4000_901E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD0_BITER_ELINKYES) | 16 | R/W | Undefined | 22.3.30/478 |
| 4000_901E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD0_BITER_ELINKNO) | 16 | R/W | Undefined | 22.3.31/479 |
| 4000_9020 | TCD Source Address (DMA_TCD1_SADDR) | 32 | R/W | Undefined | 22.3.17/466 |
| 4000_9024 | TCD Signed Source Address Offset (DMA_TCD1_SOFF) | 16 | R/W | Undefined | 22.3.18/466 |
| 4000_9026 | TCD Transfer Attributes (DMA_TCD1_ATTR) | 16 | R/W | Undefined | 22.3.19/467 |
| 4000_9028 | TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD1_NBYTES_MLNO) | 32 | R/W | Undefined | 22.3.20/468 |
| 4000_9028 | TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD1_NBYTES_MLOFFNO) | 32 | R/W | Undefined | 22.3.21/468 |
| 4000_9028 | TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD1_NBYTES_MLOFFYES) | 32 | R/W | Undefined | 22.3.22/470 |
| 4000_902C | TCD Last Source Address Adjustment (DMA_TCD1_SLAST) | 32 | R/W | Undefined | 22.3.23/471 |
| 4000_9030 | TCD Destination Address (DMA_TCD1_DADDR) | 32 | R/W | Undefined | 22.3.24/471 |
| 4000_9034 | TCD Signed Destination Address Offset (DMA_TCD1_DOFF) | 16 | R/W | Undefined | 22.3.25/472 |
| 4000_9036 | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD1_CITER_ELINKYES) | 16 | R/W | Undefined | 22.3.26/472 |
| 4000_9036 | DMA_TCD1_CITER_ELINKNO | 16 | R/W | Undefined | 22.3.27/474 |
| 4000_9038 | TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD1_DLASTSGA) | 32 | R/W | Undefined | 22.3.28/475 |
| 4000_903C | TCD Control and Status (DMA_TCD1_CSR) | 16 | R/W | Undefined | 22.3.29/475 |

*Table continues on the next page...*

## DMA memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4000_903E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD1_BITER_ELINKYES) | 16 | R/W | Undefined | 22.3.30/478 |
| 4000_903E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD1_BITER_ELINKNO) | 16 | R/W | Undefined | 22.3.31/479 |
| 4000_9040 | TCD Source Address (DMA_TCD2_SADDR) | 32 | R/W | Undefined | 22.3.17/466 |
| 4000_9044 | TCD Signed Source Address Offset (DMA_TCD2_SOFF) | 16 | R/W | Undefined | 22.3.18/466 |
| 4000_9046 | TCD Transfer Attributes (DMA_TCD2_ATTR) | 16 | R/W | Undefined | 22.3.19/467 |
| 4000_9048 | TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD2_NBYTES_MLNO) | 32 | R/W | Undefined | 22.3.20/468 |
| 4000_9048 | TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD2_NBYTES_MLOFFNO) | 32 | R/W | Undefined | 22.3.21/468 |
| 4000_9048 | TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD2_NBYTES_MLOFFYES) | 32 | R/W | Undefined | 22.3.22/470 |
| 4000_904C | TCD Last Source Address Adjustment (DMA_TCD2_SLAST) | 32 | R/W | Undefined | 22.3.23/471 |
| 4000_9050 | TCD Destination Address (DMA_TCD2_DADDR) | 32 | R/W | Undefined | 22.3.24/471 |
| 4000_9054 | TCD Signed Destination Address Offset (DMA_TCD2_DOFF) | 16 | R/W | Undefined | 22.3.25/472 |
| 4000_9056 | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD2_CITER_ELINKYES) | 16 | R/W | Undefined | 22.3.26/472 |
| 4000_9056 | DMA_TCD2_CITER_ELINKNO | 16 | R/W | Undefined | 22.3.27/474 |
| 4000_9058 | TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD2_DLASTSGA) | 32 | R/W | Undefined | 22.3.28/475 |
| 4000_905C | TCD Control and Status (DMA_TCD2_CSR) | 16 | R/W | Undefined | 22.3.29/475 |
| 4000_905E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD2_BITER_ELINKYES) | 16 | R/W | Undefined | 22.3.30/478 |
| 4000_905E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD2_BITER_ELINKNO) | 16 | R/W | Undefined | 22.3.31/479 |
| 4000_9060 | TCD Source Address (DMA_TCD3_SADDR) | 32 | R/W | Undefined | 22.3.17/466 |
| 4000_9064 | TCD Signed Source Address Offset (DMA_TCD3_SOFF) | 16 | R/W | Undefined | 22.3.18/466 |
| 4000_9066 | TCD Transfer Attributes (DMA_TCD3_ATTR) | 16 | R/W | Undefined | 22.3.19/467 |
| 4000_9068 | TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD3_NBYTES_MLNO) | 32 | R/W | Undefined | 22.3.20/468 |
| 4000_9068 | TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD3_NBYTES_MLOFFNO) | 32 | R/W | Undefined | 22.3.21/468 |
| 4000_9068 | TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD3_NBYTES_MLOFFYES) | 32 | R/W | Undefined | 22.3.22/470 |
| 4000_906C | TCD Last Source Address Adjustment (DMA_TCD3_SLAST) | 32 | R/W | Undefined | 22.3.23/471 |
| 4000_9070 | TCD Destination Address (DMA_TCD3_DADDR) | 32 | R/W | Undefined | 22.3.24/471 |

*Table continues on the next page...*

## DMA memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4000_9074 | TCD Signed Destination Address Offset (DMA_TCD3_DOFF) | 16 | R/W | Undefined | 22.3.25/472 |
| 4000_9076 | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD3_CITER_ELINKYES) | 16 | R/W | Undefined | 22.3.26/472 |
| 4000_9076 | DMA_TCD3_CITER_ELINKNO | 16 | R/W | Undefined | 22.3.27/474 |
| 4000_9078 | TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD3_DLASTSGA) | 32 | R/W | Undefined | 22.3.28/475 |
| 4000_907C | TCD Control and Status (DMA_TCD3_CSR) | 16 | R/W | Undefined | 22.3.29/475 |
| 4000_907E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD3_BITER_ELINKYES) | 16 | R/W | Undefined | 22.3.30/478 |
| 4000_907E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD3_BITER_ELINKNO) | 16 | R/W | Undefined | 22.3.31/479 |
| 4000_9080 | TCD Source Address (DMA_TCD4_SADDR) | 32 | R/W | Undefined | 22.3.17/466 |
| 4000_9084 | TCD Signed Source Address Offset (DMA_TCD4_SOFF) | 16 | R/W | Undefined | 22.3.18/466 |
| 4000_9086 | TCD Transfer Attributes (DMA_TCD4_ATTR) | 16 | R/W | Undefined | 22.3.19/467 |
| 4000_9088 | TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD4_NBYTES_MLNO) | 32 | R/W | Undefined | 22.3.20/468 |
| 4000_9088 | TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD4_NBYTES_MLOFFNO) | 32 | R/W | Undefined | 22.3.21/468 |
| 4000_9088 | TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD4_NBYTES_MLOFFYES) | 32 | R/W | Undefined | 22.3.22/470 |
| 4000_908C | TCD Last Source Address Adjustment (DMA_TCD4_SLAST) | 32 | R/W | Undefined | 22.3.23/471 |
| 4000_9090 | TCD Destination Address (DMA_TCD4_DADDR) | 32 | R/W | Undefined | 22.3.24/471 |
| 4000_9094 | TCD Signed Destination Address Offset (DMA_TCD4_DOFF) | 16 | R/W | Undefined | 22.3.25/472 |
| 4000_9096 | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD4_CITER_ELINKYES) | 16 | R/W | Undefined | 22.3.26/472 |
| 4000_9096 | DMA_TCD4_CITER_ELINKNO | 16 | R/W | Undefined | 22.3.27/474 |
| 4000_9098 | TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD4_DLASTSGA) | 32 | R/W | Undefined | 22.3.28/475 |
| 4000_909C | TCD Control and Status (DMA_TCD4_CSR) | 16 | R/W | Undefined | 22.3.29/475 |
| 4000_909E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD4_BITER_ELINKYES) | 16 | R/W | Undefined | 22.3.30/478 |
| 4000_909E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD4_BITER_ELINKNO) | 16 | R/W | Undefined | 22.3.31/479 |
| 4000_90A0 | TCD Source Address (DMA_TCD5_SADDR) | 32 | R/W | Undefined | 22.3.17/466 |
| 4000_90A4 | TCD Signed Source Address Offset (DMA_TCD5_SOFF) | 16 | R/W | Undefined | 22.3.18/466 |
| 4000_90A6 | TCD Transfer Attributes (DMA_TCD5_ATTR) | 16 | R/W | Undefined | 22.3.19/467 |
| 4000_90A8 | TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD5_NBYTES_MLNO) | 32 | R/W | Undefined | 22.3.20/468 |

*Table continues on the next page...*

## DMA memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4000_90A8 | TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD5_NBYTES_MLOFFNO) | 32 | R/W | Undefined | 22.3.21/468 |
| 4000_90A8 | TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD5_NBYTES_MLOFFYES) | 32 | R/W | Undefined | 22.3.22/470 |
| 4000_90AC | TCD Last Source Address Adjustment (DMA_TCD5_SLAST) | 32 | R/W | Undefined | 22.3.23/471 |
| 4000_90B0 | TCD Destination Address (DMA_TCD5_DADDR) | 32 | R/W | Undefined | 22.3.24/471 |
| 4000_90B4 | TCD Signed Destination Address Offset (DMA_TCD5_DOFF) | 16 | R/W | Undefined | 22.3.25/472 |
| 4000_90B6 | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD5_CITER_ELINKYES) | 16 | R/W | Undefined | 22.3.26/472 |
| 4000_90B6 | DMA_TCD5_CITER_ELINKNO | 16 | R/W | Undefined | 22.3.27/474 |
| 4000_90B8 | TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD5_DLASTSGA) | 32 | R/W | Undefined | 22.3.28/475 |
| 4000_90BC | TCD Control and Status (DMA_TCD5_CSR) | 16 | R/W | Undefined | 22.3.29/475 |
| 4000_90BE | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD5_BITER_ELINKYES) | 16 | R/W | Undefined | 22.3.30/478 |
| 4000_90BE | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD5_BITER_ELINKNO) | 16 | R/W | Undefined | 22.3.31/479 |
| 4000_90C0 | TCD Source Address (DMA_TCD6_SADDR) | 32 | R/W | Undefined | 22.3.17/466 |
| 4000_90C4 | TCD Signed Source Address Offset (DMA_TCD6_SOFF) | 16 | R/W | Undefined | 22.3.18/466 |
| 4000_90C6 | TCD Transfer Attributes (DMA_TCD6_ATTR) | 16 | R/W | Undefined | 22.3.19/467 |
| 4000_90C8 | TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD6_NBYTES_MLNO) | 32 | R/W | Undefined | 22.3.20/468 |
| 4000_90C8 | TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD6_NBYTES_MLOFFNO) | 32 | R/W | Undefined | 22.3.21/468 |
| 4000_90C8 | TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD6_NBYTES_MLOFFYES) | 32 | R/W | Undefined | 22.3.22/470 |
| 4000_90CC | TCD Last Source Address Adjustment (DMA_TCD6_SLAST) | 32 | R/W | Undefined | 22.3.23/471 |
| 4000_90D0 | TCD Destination Address (DMA_TCD6_DADDR) | 32 | R/W | Undefined | 22.3.24/471 |
| 4000_90D4 | TCD Signed Destination Address Offset (DMA_TCD6_DOFF) | 16 | R/W | Undefined | 22.3.25/472 |
| 4000_90D6 | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD6_CITER_ELINKYES) | 16 | R/W | Undefined | 22.3.26/472 |
| 4000_90D6 | DMA_TCD6_CITER_ELINKNO | 16 | R/W | Undefined | 22.3.27/474 |
| 4000_90D8 | TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD6_DLASTSGA) | 32 | R/W | Undefined | 22.3.28/475 |
| 4000_90DC | TCD Control and Status (DMA_TCD6_CSR) | 16 | R/W | Undefined | 22.3.29/475 |
| 4000_90DE | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD6_BITER_ELINKYES) | 16 | R/W | Undefined | 22.3.30/478 |

*Table continues on the next page...*

## DMA memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4000_90DE | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD6_BITER_ELINKNO) | 16 | R/W | Undefined | 22.3.31/479 |
| 4000_90E0 | TCD Source Address (DMA_TCD7_SADDR) | 32 | R/W | Undefined | 22.3.17/466 |
| 4000_90E4 | TCD Signed Source Address Offset (DMA_TCD7_SOFF) | 16 | R/W | Undefined | 22.3.18/466 |
| 4000_90E6 | TCD Transfer Attributes (DMA_TCD7_ATTR) | 16 | R/W | Undefined | 22.3.19/467 |
| 4000_90E8 | TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD7_NBYTES_MLNO) | 32 | R/W | Undefined | 22.3.20/468 |
| 4000_90E8 | TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD7_NBYTES_MLOFFNO) | 32 | R/W | Undefined | 22.3.21/468 |
| 4000_90E8 | TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD7_NBYTES_MLOFFYES) | 32 | R/W | Undefined | 22.3.22/470 |
| 4000_90EC | TCD Last Source Address Adjustment (DMA_TCD7_SLAST) | 32 | R/W | Undefined | 22.3.23/471 |
| 4000_90F0 | TCD Destination Address (DMA_TCD7_DADDR) | 32 | R/W | Undefined | 22.3.24/471 |
| 4000_90F4 | TCD Signed Destination Address Offset (DMA_TCD7_DOFF) | 16 | R/W | Undefined | 22.3.25/472 |
| 4000_90F6 | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD7_CITER_ELINKYES) | 16 | R/W | Undefined | 22.3.26/472 |
| 4000_90F6 | DMA_TCD7_CITER_ELINKNO | 16 | R/W | Undefined | 22.3.27/474 |
| 4000_90F8 | TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD7_DLASTSGA) | 32 | R/W | Undefined | 22.3.28/475 |
| 4000_90FC | TCD Control and Status (DMA_TCD7_CSR) | 16 | R/W | Undefined | 22.3.29/475 |
| 4000_90FE | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD7_BITER_ELINKYES) | 16 | R/W | Undefined | 22.3.30/478 |
| 4000_90FE | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD7_BITER_ELINKNO) | 16 | R/W | Undefined | 22.3.31/479 |
| 4000_9100 | TCD Source Address (DMA_TCD8_SADDR) | 32 | R/W | Undefined | 22.3.17/466 |
| 4000_9104 | TCD Signed Source Address Offset (DMA_TCD8_SOFF) | 16 | R/W | Undefined | 22.3.18/466 |
| 4000_9106 | TCD Transfer Attributes (DMA_TCD8_ATTR) | 16 | R/W | Undefined | 22.3.19/467 |
| 4000_9108 | TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD8_NBYTES_MLNO) | 32 | R/W | Undefined | 22.3.20/468 |
| 4000_9108 | TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD8_NBYTES_MLOFFNO) | 32 | R/W | Undefined | 22.3.21/468 |
| 4000_9108 | TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD8_NBYTES_MLOFFYES) | 32 | R/W | Undefined | 22.3.22/470 |
| 4000_910C | TCD Last Source Address Adjustment (DMA_TCD8_SLAST) | 32 | R/W | Undefined | 22.3.23/471 |
| 4000_9110 | TCD Destination Address (DMA_TCD8_DADDR) | 32 | R/W | Undefined | 22.3.24/471 |
| 4000_9114 | TCD Signed Destination Address Offset (DMA_TCD8_DOFF) | 16 | R/W | Undefined | 22.3.25/472 |
| 4000_9116 | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD8_CITER_ELINKYES) | 16 | R/W | Undefined | 22.3.26/472 |

*Table continues on the next page...*

## DMA memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4000_9116 | DMA_TCD8_CITER_ELINKNO | 16 | R/W | Undefined | 22.3.27/474 |
| 4000_9118 | TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD8_DLASTSGA) | 32 | R/W | Undefined | 22.3.28/475 |
| 4000_911C | TCD Control and Status (DMA_TCD8_CSR) | 16 | R/W | Undefined | 22.3.29/475 |
| 4000_911E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD8_BITER_ELINKYES) | 16 | R/W | Undefined | 22.3.30/478 |
| 4000_911E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD8_BITER_ELINKNO) | 16 | R/W | Undefined | 22.3.31/479 |
| 4000_9120 | TCD Source Address (DMA_TCD9_SADDR) | 32 | R/W | Undefined | 22.3.17/466 |
| 4000_9124 | TCD Signed Source Address Offset (DMA_TCD9_SOFF) | 16 | R/W | Undefined | 22.3.18/466 |
| 4000_9126 | TCD Transfer Attributes (DMA_TCD9_ATTR) | 16 | R/W | Undefined | 22.3.19/467 |
| 4000_9128 | TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD9_NBYTES_MLNO) | 32 | R/W | Undefined | 22.3.20/468 |
| 4000_9128 | TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD9_NBYTES_MLOFFNO) | 32 | R/W | Undefined | 22.3.21/468 |
| 4000_9128 | TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD9_NBYTES_MLOFFYES) | 32 | R/W | Undefined | 22.3.22/470 |
| 4000_912C | TCD Last Source Address Adjustment (DMA_TCD9_SLAST) | 32 | R/W | Undefined | 22.3.23/471 |
| 4000_9130 | TCD Destination Address (DMA_TCD9_DADDR) | 32 | R/W | Undefined | 22.3.24/471 |
| 4000_9134 | TCD Signed Destination Address Offset (DMA_TCD9_DOFF) | 16 | R/W | Undefined | 22.3.25/472 |
| 4000_9136 | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD9_CITER_ELINKYES) | 16 | R/W | Undefined | 22.3.26/472 |
| 4000_9136 | DMA_TCD9_CITER_ELINKNO | 16 | R/W | Undefined | 22.3.27/474 |
| 4000_9138 | TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD9_DLASTSGA) | 32 | R/W | Undefined | 22.3.28/475 |
| 4000_913C | TCD Control and Status (DMA_TCD9_CSR) | 16 | R/W | Undefined | 22.3.29/475 |
| 4000_913E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD9_BITER_ELINKYES) | 16 | R/W | Undefined | 22.3.30/478 |
| 4000_913E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD9_BITER_ELINKNO) | 16 | R/W | Undefined | 22.3.31/479 |
| 4000_9140 | TCD Source Address (DMA_TCD10_SADDR) | 32 | R/W | Undefined | 22.3.17/466 |
| 4000_9144 | TCD Signed Source Address Offset (DMA_TCD10_SOFF) | 16 | R/W | Undefined | 22.3.18/466 |
| 4000_9146 | TCD Transfer Attributes (DMA_TCD10_ATTR) | 16 | R/W | Undefined | 22.3.19/467 |
| 4000_9148 | TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD10_NBYTES_MLNO) | 32 | R/W | Undefined | 22.3.20/468 |
| 4000_9148 | TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD10_NBYTES_MLOFFNO) | 32 | R/W | Undefined | 22.3.21/468 |
| 4000_9148 | TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD10_NBYTES_MLOFFYES) | 32 | R/W | Undefined | 22.3.22/470 |

*Table continues on the next page...*

## DMA memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4000_914C | TCD Last Source Address Adjustment (DMA_TCD10_SLAST) | 32 | R/W | Undefined | 22.3.23/471 |
| 4000_9150 | TCD Destination Address (DMA_TCD10_DADDR) | 32 | R/W | Undefined | 22.3.24/471 |
| 4000_9154 | TCD Signed Destination Address Offset (DMA_TCD10_DOFF) | 16 | R/W | Undefined | 22.3.25/472 |
| 4000_9156 | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD10_CITER_ELINKYES) | 16 | R/W | Undefined | 22.3.26/472 |
| 4000_9156 | DMA_TCD10_CITER_ELINKNO | 16 | R/W | Undefined | 22.3.27/474 |
| 4000_9158 | TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD10_DLASTSGA) | 32 | R/W | Undefined | 22.3.28/475 |
| 4000_915C | TCD Control and Status (DMA_TCD10_CSR) | 16 | R/W | Undefined | 22.3.29/475 |
| 4000_915E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD10_BITER_ELINKYES) | 16 | R/W | Undefined | 22.3.30/478 |
| 4000_915E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD10_BITER_ELINKNO) | 16 | R/W | Undefined | 22.3.31/479 |
| 4000_9160 | TCD Source Address (DMA_TCD11_SADDR) | 32 | R/W | Undefined | 22.3.17/466 |
| 4000_9164 | TCD Signed Source Address Offset (DMA_TCD11_SOFF) | 16 | R/W | Undefined | 22.3.18/466 |
| 4000_9166 | TCD Transfer Attributes (DMA_TCD11_ATTR) | 16 | R/W | Undefined | 22.3.19/467 |
| 4000_9168 | TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD11_NBYTES_MLNO) | 32 | R/W | Undefined | 22.3.20/468 |
| 4000_9168 | TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD11_NBYTES_MLOFFNO) | 32 | R/W | Undefined | 22.3.21/468 |
| 4000_9168 | TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD11_NBYTES_MLOFFYES) | 32 | R/W | Undefined | 22.3.22/470 |
| 4000_916C | TCD Last Source Address Adjustment (DMA_TCD11_SLAST) | 32 | R/W | Undefined | 22.3.23/471 |
| 4000_9170 | TCD Destination Address (DMA_TCD11_DADDR) | 32 | R/W | Undefined | 22.3.24/471 |
| 4000_9174 | TCD Signed Destination Address Offset (DMA_TCD11_DOFF) | 16 | R/W | Undefined | 22.3.25/472 |
| 4000_9176 | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD11_CITER_ELINKYES) | 16 | R/W | Undefined | 22.3.26/472 |
| 4000_9176 | DMA_TCD11_CITER_ELINKNO | 16 | R/W | Undefined | 22.3.27/474 |
| 4000_9178 | TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD11_DLASTSGA) | 32 | R/W | Undefined | 22.3.28/475 |
| 4000_917C | TCD Control and Status (DMA_TCD11_CSR) | 16 | R/W | Undefined | 22.3.29/475 |
| 4000_917E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD11_BITER_ELINKYES) | 16 | R/W | Undefined | 22.3.30/478 |
| 4000_917E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD11_BITER_ELINKNO) | 16 | R/W | Undefined | 22.3.31/479 |

*Table continues on the next page...*

## DMA memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4000_9180 | TCD Source Address (DMA_TCD12_SADDR) | 32 | R/W | Undefined | 22.3.17/466 |
| 4000_9184 | TCD Signed Source Address Offset (DMA_TCD12_SOFF) | 16 | R/W | Undefined | 22.3.18/466 |
| 4000_9186 | TCD Transfer Attributes (DMA_TCD12_ATTR) | 16 | R/W | Undefined | 22.3.19/467 |
| 4000_9188 | TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD12_NBYTES_MLNO) | 32 | R/W | Undefined | 22.3.20/468 |
| 4000_9188 | TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD12_NBYTES_MLOFFNO) | 32 | R/W | Undefined | 22.3.21/468 |
| 4000_9188 | TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD12_NBYTES_MLOFFYES) | 32 | R/W | Undefined | 22.3.22/470 |
| 4000_918C | TCD Last Source Address Adjustment (DMA_TCD12_SLAST) | 32 | R/W | Undefined | 22.3.23/471 |
| 4000_9190 | TCD Destination Address (DMA_TCD12_DADDR) | 32 | R/W | Undefined | 22.3.24/471 |
| 4000_9194 | TCD Signed Destination Address Offset (DMA_TCD12_DOFF) | 16 | R/W | Undefined | 22.3.25/472 |
| 4000_9196 | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD12_CITER_ELINKYES) | 16 | R/W | Undefined | 22.3.26/472 |
| 4000_9196 | DMA_TCD12_CITER_ELINKNO | 16 | R/W | Undefined | 22.3.27/474 |
| 4000_9198 | TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD12_DLASTSGA) | 32 | R/W | Undefined | 22.3.28/475 |
| 4000_919C | TCD Control and Status (DMA_TCD12_CSR) | 16 | R/W | Undefined | 22.3.29/475 |
| 4000_919E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD12_BITER_ELINKYES) | 16 | R/W | Undefined | 22.3.30/478 |
| 4000_919E | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD12_BITER_ELINKNO) | 16 | R/W | Undefined | 22.3.31/479 |
| 4000_91A0 | TCD Source Address (DMA_TCD13_SADDR) | 32 | R/W | Undefined | 22.3.17/466 |
| 4000_91A4 | TCD Signed Source Address Offset (DMA_TCD13_SOFF) | 16 | R/W | Undefined | 22.3.18/466 |
| 4000_91A6 | TCD Transfer Attributes (DMA_TCD13_ATTR) | 16 | R/W | Undefined | 22.3.19/467 |
| 4000_91A8 | TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD13_NBYTES_MLNO) | 32 | R/W | Undefined | 22.3.20/468 |
| 4000_91A8 | TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD13_NBYTES_MLOFFNO) | 32 | R/W | Undefined | 22.3.21/468 |
| 4000_91A8 | TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD13_NBYTES_MLOFFYES) | 32 | R/W | Undefined | 22.3.22/470 |
| 4000_91AC | TCD Last Source Address Adjustment (DMA_TCD13_SLAST) | 32 | R/W | Undefined | 22.3.23/471 |
| 4000_91B0 | TCD Destination Address (DMA_TCD13_DADDR) | 32 | R/W | Undefined | 22.3.24/471 |
| 4000_91B4 | TCD Signed Destination Address Offset (DMA_TCD13_DOFF) | 16 | R/W | Undefined | 22.3.25/472 |
| 4000_91B6 | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD13_CITER_ELINKYES) | 16 | R/W | Undefined | 22.3.26/472 |
| 4000_91B6 | DMA_TCD13_CITER_ELINKNO | 16 | R/W | Undefined | 22.3.27/474 |

*Table continues on the next page...*

## DMA memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4000_91B8 | TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD13_DLASTSGA) | 32 | R/W | Undefined | 22.3.28/475 |
| 4000_91BC | TCD Control and Status (DMA_TCD13_CSR) | 16 | R/W | Undefined | 22.3.29/475 |
| 4000_91BE | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD13_BITER_ELINKYES) | 16 | R/W | Undefined | 22.3.30/478 |
| 4000_91BE | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD13_BITER_ELINKNO) | 16 | R/W | Undefined | 22.3.31/479 |
| 4000_91C0 | TCD Source Address (DMA_TCD14_SADDR) | 32 | R/W | Undefined | 22.3.17/466 |
| 4000_91C4 | TCD Signed Source Address Offset (DMA_TCD14_SOFF) | 16 | R/W | Undefined | 22.3.18/466 |
| 4000_91C6 | TCD Transfer Attributes (DMA_TCD14_ATTR) | 16 | R/W | Undefined | 22.3.19/467 |
| 4000_91C8 | TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD14_NBYTES_MLNO) | 32 | R/W | Undefined | 22.3.20/468 |
| 4000_91C8 | TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD14_NBYTES_MLOFFNO) | 32 | R/W | Undefined | 22.3.21/468 |
| 4000_91C8 | TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD14_NBYTES_MLOFFYES) | 32 | R/W | Undefined | 22.3.22/470 |
| 4000_91CC | TCD Last Source Address Adjustment (DMA_TCD14_SLAST) | 32 | R/W | Undefined | 22.3.23/471 |
| 4000_91D0 | TCD Destination Address (DMA_TCD14_DADDR) | 32 | R/W | Undefined | 22.3.24/471 |
| 4000_91D4 | TCD Signed Destination Address Offset (DMA_TCD14_DOFF) | 16 | R/W | Undefined | 22.3.25/472 |
| 4000_91D6 | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD14_CITER_ELINKYES) | 16 | R/W | Undefined | 22.3.26/472 |
| 4000_91D6 | DMA_TCD14_CITER_ELINKNO | 16 | R/W | Undefined | 22.3.27/474 |
| 4000_91D8 | TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD14_DLASTSGA) | 32 | R/W | Undefined | 22.3.28/475 |
| 4000_91DC | TCD Control and Status (DMA_TCD14_CSR) | 16 | R/W | Undefined | 22.3.29/475 |
| 4000_91DE | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD14_BITER_ELINKYES) | 16 | R/W | Undefined | 22.3.30/478 |
| 4000_91DE | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD14_BITER_ELINKNO) | 16 | R/W | Undefined | 22.3.31/479 |
| 4000_91E0 | TCD Source Address (DMA_TCD15_SADDR) | 32 | R/W | Undefined | 22.3.17/466 |
| 4000_91E4 | TCD Signed Source Address Offset (DMA_TCD15_SOFF) | 16 | R/W | Undefined | 22.3.18/466 |
| 4000_91E6 | TCD Transfer Attributes (DMA_TCD15_ATTR) | 16 | R/W | Undefined | 22.3.19/467 |
| 4000_91E8 | TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD15_NBYTES_MLNO) | 32 | R/W | Undefined | 22.3.20/468 |
| 4000_91E8 | TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD15_NBYTES_MLOFFNO) | 32 | R/W | Undefined | 22.3.21/468 |

*Table continues on the next page...*

**DMA memory map (continued)**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4000_91E8 | TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD15_NBYTES_MLOFFYES) | 32 | R/W | Undefined | 22.3.22/470 |
| 4000_91EC | TCD Last Source Address Adjustment (DMA_TCD15_SLAST) | 32 | R/W | Undefined | 22.3.23/471 |
| 4000_91F0 | TCD Destination Address (DMA_TCD15_DADDR) | 32 | R/W | Undefined | 22.3.24/471 |
| 4000_91F4 | TCD Signed Destination Address Offset (DMA_TCD15_DOFF) | 16 | R/W | Undefined | 22.3.25/472 |
| 4000_91F6 | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD15_CITER_ELINKYES) | 16 | R/W | Undefined | 22.3.26/472 |
| 4000_91F6 | DMA_TCD15_CITER_ELINKNO | 16 | R/W | Undefined | 22.3.27/474 |
| 4000_91F8 | TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD15_DLASTSGA) | 32 | R/W | Undefined | 22.3.28/475 |
| 4000_91FC | TCD Control and Status (DMA_TCD15_CSR) | 16 | R/W | Undefined | 22.3.29/475 |
| 4000_91FE | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD15_BITER_ELINKYES) | 16 | R/W | Undefined | 22.3.30/478 |
| 4000_91FE | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD15_BITER_ELINKNO) | 16 | R/W | Undefined | 22.3.31/479 |

## 22.3.1 Control Register (DMA_CR)

The CR defines the basic operating configuration of the DMA.

Arbitration can be configured to use either a fixed-priority or a round-robin scheme. For fixed-priority arbitration, the highest priority channel requesting service is selected to execute. The channel priority registers assign the priorities; see the DCHPRIn registers. For round-robin arbitration, the channel priorities are ignored and channels are cycled through (from high to low channel number) without regard to priority.

**NOTE**

For proper operation, writes to the CR register must be performed only when the DMA channels are inactive; that is, when TCDn_CSR[ACTIVE] bits are cleared.

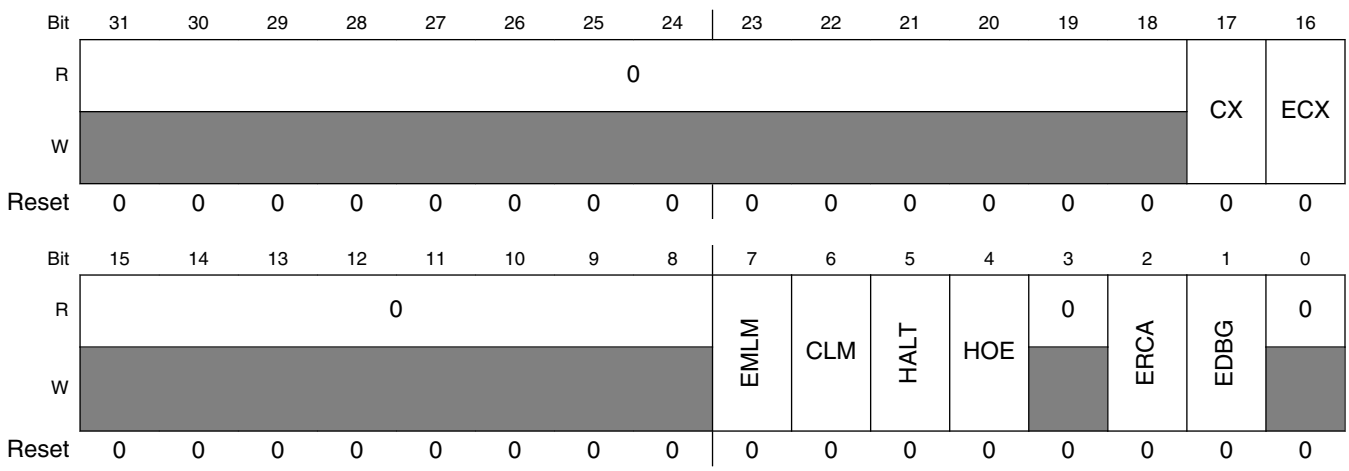
Minor loop offsets are address offset values added to the final source address (TCDn_SADDR) or destination address (TCDn_DADDR) upon minor loop completion. When minor loop offsets are enabled, the minor loop offset (MLOFF) is added to the final source address (TCDn_SADDR), to the final destination address (TCDn_DADDR), or to both prior to the addresses being written back into the TCD. If the major loop is

complete, the minor loop offset is ignored and the major loop address offsets (TCDn_SLAST and TCDn_DLAST_SGA) are used to compute the next TCDn_SADDR and TCDn_DADDR values.

When minor loop mapping is enabled (EMLM is 1), TCDn word2 is redefined. A portion of TCDn word2 is used to specify multiple fields: a source enable bit (SMLOE) to specify the minor loop offset should be applied to the source address (TCDn_SADDR) upon minor loop completion, a destination enable bit (DMLOE) to specify the minor loop offset should be applied to the destination address (TCDn_DADDR) upon minor loop completion, and the sign extended minor loop offset value (MLOFF). The same offset value (MLOFF) is used for both source and destination minor loop offsets. When either minor loop offset is enabled (SMLOE set or DMLOE set), the NBYTES field is reduced to 10 bits. When both minor loop offsets are disabled (SMLOE cleared and DMLOE cleared), the NBYTES field is a 30-bit vector.

When minor loop mapping is disabled (EMLM is 0), all 32 bits of TCDn word2 are assigned to the NBYTES field.

Address: 4000_8000h base + 0h offset = 4000_8000h




## DMA_CR field descriptions

| Field | Description |
|---|---|
| 31–18<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 17<br>CX | Cancel Transfer<br><br>0 Normal operation<br>1 Cancel the remaining data transfer. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The CX bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed. |
| 16<br>ECX | Error Cancel Transfer |

*Table continues on the next page...*

## DMA_CR field descriptions (continued)

| Field | Description |
|---|---|
| | 0 Normal operation<br>1 Cancel the remaining data transfer in the same fashion as the CX bit. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel is honored. In addition to cancelling the transfer, ECX treats the cancel as an error condition, thus updating the Error Status register (DMAx_ES) and generating an optional error interrupt. |
| 15–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>EMLM | Enable Minor Loop Mapping<br>0 Disabled. TCDn.word2 is defined as a 32-bit NBYTES field.<br>1 Enabled. TCDn.word2 is redefined to include individual enable fields, an offset field, and the NBYTES field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The NBYTES field is reduced when either offset is enabled. |
| 6<br>CLM | Continuous Link Mode<br>0 A minor loop channel link made to itself goes through channel arbitration before being activated again.<br>1 A minor loop channel link made to itself does not go through channel arbitration before being activated again. Upon minor loop completion, the channel activates again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop. |
| 5<br>HALT | Halt DMA Operations<br>0 Normal operation<br>1 Stall the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this bit is cleared. |
| 4<br>HOE | Halt On Error<br>0 Normal operation<br>1 Any error causes the HALT bit to set. Subsequently, all service requests are ignored until the HALT bit is cleared. |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>ERCA | Enable Round Robin Channel Arbitration<br>0 Fixed priority arbitration is used for channel selection .<br>1 Round robin arbitration is used for channel selection . |
| 1<br>EDBG | Enable Debug<br>0 When in debug mode, the DMA continues to operate.<br>1 When in debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the system exits debug mode or the EDBG bit is cleared. |
| 0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 22.3.2 Error Status Register (DMA_ES)

The ES provides information concerning the last recorded channel error. Channel errors can be caused by:

- A configuration error, that is:
  - An illegal setting in the transfer-control descriptor, or
  - An illegal priority register setting in fixed-arbitration
- An error termination to a bus master read or write cycle

See the Error Reporting and Handling section for more details.

Address: 4000_8000h base + 4h offset = 4000_8004h

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

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | CPE | 0 | | ERRCHN | | | | SAE | SOE | DAE | DOE | NCE | SGE | SBE | DBE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_ES field descriptions**

| Field | Description |
|---|---|
| 31<br>VLD | Logical OR of all ERR status bits<br><br>0 No ERR bits are set<br>1 At least one ERR bit is set indicating a valid error exists that has not been cleared |
| 30–17<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 16<br>ECX | Transfer Canceled<br><br>0 No canceled transfers<br>1 The last recorded entry was a canceled transfer by the error cancel transfer input |
| 15<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 14<br>CPE | Channel Priority Error<br><br>0 No channel priority error<br>1 The last recorded error was a configuration error in the channel priorities . Channel priorities are not unique. |
| 13–12<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

*Table continues on the next page...*

## DMA_ES field descriptions (continued)

| Field | Description |
|---|---|
| 11–8<br>ERRCHN | Error Channel Number or Canceled Channel Number<br><br>The channel number of the last recorded error (excluding CPE errors) or last recorded error canceled transfer. |
| 7<br>SAE | Source Address Error<br><br>0 No source address configuration error.<br>1 The last recorded error was a configuration error detected in the TCDn_SADDR field. TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE]. |
| 6<br>SOE | Source Offset Error<br><br>0 No source offset configuration error<br>1 The last recorded error was a configuration error detected in the TCDn_SOFF field. TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE]. |
| 5<br>DAE | Destination Address Error<br><br>0 No destination address configuration error<br>1 The last recorded error was a configuration error detected in the TCDn_DADDR field. TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE]. |
| 4<br>DOE | Destination Offset Error<br><br>0 No destination offset configuration error<br>1 The last recorded error was a configuration error detected in the TCDn_DOFF field. TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE]. |
| 3<br>NCE | NBYTES/CITER Configuration Error<br><br>0 No NBYTES/CITER configuration error<br>1 The last recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields.<br>• TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE], or<br>• TCDn_CITER[CITER] is equal to zero, or<br>• TCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK] |
| 2<br>SGE | Scatter/Gather Configuration Error<br><br>0 No scatter/gather configuration error<br>1 The last recorded error was a configuration error detected in the TCDn_DLASTSGA field. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCDn_CSR[ESG] is enabled. TCDn_DLASTSGA is not on a 32 byte boundary. |
| 1<br>SBE | Source Bus Error<br><br>0 No source bus error<br>1 The last recorded error was a bus error on a source read |
| 0<br>DBE | Destination Bus Error<br><br>0 No destination bus error<br>1 The last recorded error was a bus error on a destination write |

## 22.3.3 Enable Request Register (DMA_ERQ)

The ERQ register provides a bit map for the 16 implemented channels to enable the request signal for each channel. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ. The {S,C}ERQ registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to the ERQ.

DMA request input signals and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the DMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

Address: 4000_8000h base + Ch offset = 4000_800Ch

| Bit | 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ERQ15 | ERQ14 | ERQ13 | ERQ12 | ERQ11 | ERQ10 | ERQ9 | ERQ8 | ERQ7 | ERQ6 | ERQ5 | ERQ4 | ERQ3 | ERQ2 | ERQ1 | ERQ0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_ERQ field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15<br>ERQ15 | Enable DMA Request 15<br><br>0 The DMA request signal for the corresponding channel is disabled<br>1 The DMA request signal for the corresponding channel is enabled |
| 14<br>ERQ14 | Enable DMA Request 14<br><br>0 The DMA request signal for the corresponding channel is disabled<br>1 The DMA request signal for the corresponding channel is enabled |
| 13<br>ERQ13 | Enable DMA Request 13<br><br>0 The DMA request signal for the corresponding channel is disabled<br>1 The DMA request signal for the corresponding channel is enabled |

*Table continues on the next page...*

## DMA_ERQ field descriptions (continued)

| Field | Description |
|---|---|
| 12<br>ERQ12 | Enable DMA Request 12<br><br>0 The DMA request signal for the corresponding channel is disabled<br>1 The DMA request signal for the corresponding channel is enabled |
| 11<br>ERQ11 | Enable DMA Request 11<br><br>0 The DMA request signal for the corresponding channel is disabled<br>1 The DMA request signal for the corresponding channel is enabled |
| 10<br>ERQ10 | Enable DMA Request 10<br><br>0 The DMA request signal for the corresponding channel is disabled<br>1 The DMA request signal for the corresponding channel is enabled |
| 9<br>ERQ9 | Enable DMA Request 9<br><br>0 The DMA request signal for the corresponding channel is disabled<br>1 The DMA request signal for the corresponding channel is enabled |
| 8<br>ERQ8 | Enable DMA Request 8<br><br>0 The DMA request signal for the corresponding channel is disabled<br>1 The DMA request signal for the corresponding channel is enabled |
| 7<br>ERQ7 | Enable DMA Request 7<br><br>0 The DMA request signal for the corresponding channel is disabled<br>1 The DMA request signal for the corresponding channel is enabled |
| 6<br>ERQ6 | Enable DMA Request 6<br><br>0 The DMA request signal for the corresponding channel is disabled<br>1 The DMA request signal for the corresponding channel is enabled |
| 5<br>ERQ5 | Enable DMA Request 5<br><br>0 The DMA request signal for the corresponding channel is disabled<br>1 The DMA request signal for the corresponding channel is enabled |
| 4<br>ERQ4 | Enable DMA Request 4<br><br>0 The DMA request signal for the corresponding channel is disabled<br>1 The DMA request signal for the corresponding channel is enabled |
| 3<br>ERQ3 | Enable DMA Request 3<br><br>0 The DMA request signal for the corresponding channel is disabled<br>1 The DMA request signal for the corresponding channel is enabled |
| 2<br>ERQ2 | Enable DMA Request 2<br><br>0 The DMA request signal for the corresponding channel is disabled<br>1 The DMA request signal for the corresponding channel is enabled |
| 1<br>ERQ1 | Enable DMA Request 1<br><br>0 The DMA request signal for the corresponding channel is disabled<br>1 The DMA request signal for the corresponding channel is enabled |

*Table continues on the next page...*

**DMA_ERQ field descriptions (continued)**

| Field | Description |
|---|---|
| 0<br>ERQ0 | Enable DMA Request 0<br><br>0 The DMA request signal for the corresponding channel is disabled<br>1 The DMA request signal for the corresponding channel is enabled |

## 22.3.4 Enable Error Interrupt Register (DMA_EEI)

The EEI register provides a bit map for the 16 channels to enable the error interrupt signal for each channel. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI. The {S,C}EEI are provided so the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI register.

The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.

Address: 4000_8000h base + 14h offset = 4000_8014h

| Bit | 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EEI15 | EEI14 | EEI13 | EEI12 | EEI11 | EEI10 | EEI9 | EEI8 | EEI7 | EEI6 | EEI5 | EEI4 | EEI3 | EEI2 | EEI1 | EEI0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_EEI field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15<br>EEI15 | Enable Error Interrupt 15<br><br>0 The error signal for corresponding channel does not generate an error interrupt<br>1 The assertion of the error signal for corresponding channel generates an error interrupt request |
| 14<br>EEI14 | Enable Error Interrupt 14<br><br>0 The error signal for corresponding channel does not generate an error interrupt<br>1 The assertion of the error signal for corresponding channel generates an error interrupt request |

*Table continues on the next page...*

## DMA_EEI field descriptions (continued)

| Field | Description |
|---|---|
| 13<br>EEI13 | Enable Error Interrupt 13<br><br>0 The error signal for corresponding channel does not generate an error interrupt<br>1 The assertion of the error signal for corresponding channel generates an error interrupt request |
| 12<br>EEI12 | Enable Error Interrupt 12<br><br>0 The error signal for corresponding channel does not generate an error interrupt<br>1 The assertion of the error signal for corresponding channel generates an error interrupt request |
| 11<br>EEI11 | Enable Error Interrupt 11<br><br>0 The error signal for corresponding channel does not generate an error interrupt<br>1 The assertion of the error signal for corresponding channel generates an error interrupt request |
| 10<br>EEI10 | Enable Error Interrupt 10<br><br>0 The error signal for corresponding channel does not generate an error interrupt<br>1 The assertion of the error signal for corresponding channel generates an error interrupt request |
| 9<br>EEI9 | Enable Error Interrupt 9<br><br>0 The error signal for corresponding channel does not generate an error interrupt<br>1 The assertion of the error signal for corresponding channel generates an error interrupt request |
| 8<br>EEI8 | Enable Error Interrupt 8<br><br>0 The error signal for corresponding channel does not generate an error interrupt<br>1 The assertion of the error signal for corresponding channel generates an error interrupt request |
| 7<br>EEI7 | Enable Error Interrupt 7<br><br>0 The error signal for corresponding channel does not generate an error interrupt<br>1 The assertion of the error signal for corresponding channel generates an error interrupt request |
| 6<br>EEI6 | Enable Error Interrupt 6<br><br>0 The error signal for corresponding channel does not generate an error interrupt<br>1 The assertion of the error signal for corresponding channel generates an error interrupt request |
| 5<br>EEI5 | Enable Error Interrupt 5<br><br>0 The error signal for corresponding channel does not generate an error interrupt<br>1 The assertion of the error signal for corresponding channel generates an error interrupt request |
| 4<br>EEI4 | Enable Error Interrupt 4<br><br>0 The error signal for corresponding channel does not generate an error interrupt<br>1 The assertion of the error signal for corresponding channel generates an error interrupt request |
| 3<br>EEI3 | Enable Error Interrupt 3<br><br>0 The error signal for corresponding channel does not generate an error interrupt<br>1 The assertion of the error signal for corresponding channel generates an error interrupt request |
| 2<br>EEI2 | Enable Error Interrupt 2<br><br>0 The error signal for corresponding channel does not generate an error interrupt<br>1 The assertion of the error signal for corresponding channel generates an error interrupt request |

*Table continues on the next page...*

**DMA_EEI field descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>EEI1 | Enable Error Interrupt 1<br><br>0 The error signal for corresponding channel does not generate an error interrupt<br>1 The assertion of the error signal for corresponding channel generates an error interrupt request |
| 0<br>EEI0 | Enable Error Interrupt 0<br><br>0 The error signal for corresponding channel does not generate an error interrupt<br>1 The assertion of the error signal for corresponding channel generates an error interrupt request |

## 22.3.5 Clear Enable Error Interrupt Register (DMA_CEEI)

The CEEI provides a simple memory-mapped mechanism to clear a given bit in the EEI to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI to be cleared. Setting the CAEE bit provides a global clear function, forcing the EEI contents to be cleared, disabling all DMA request inputs. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 4000_8000h base + 18h offset = 4000_8018h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | 0 | | | 0 | | | |
| Write | NOP | CAEE | 0 | | CEEI | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_CEEI field descriptions**

| Field | Description |
|---|---|
| 7<br>NOP | No Op enable<br><br>0 Normal operation<br>1 No operation, ignore the other bits in this register |
| 6<br>CAEE | Clear All Enable Error Interrupts<br><br>0 Clear only the EEI bit specified in the CEEI field<br>1 Clear all bits in EEI |
| 5–4<br>Reserved | This field is reserved. |
| 3–0<br>CEEI | Clear Enable Error Interrupt<br><br>Clears the corresponding bit in EEI |

## 22.3.6 Set Enable Error Interrupt Register (DMA_SEEI)

The SEEI provides a simple memory-mapped mechanism to set a given bit in the EEI to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI to be set. Setting the SAEE bit provides a global set function, forcing the entire EEI contents to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 4000_8000h base + 19h offset = 4000_8019h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | 0 | 0 | | | | | |
| Write | NOP | SAEE | 0 | | SEEI | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_SEEI field descriptions**

| Field | Description |
|---|---|
| 7<br>NOP | No Op enable<br><br>0 Normal operation<br>1 No operation, ignore the other bits in this register |
| 6<br>SAEE | Sets All Enable Error Interrupts<br><br>0 Set only the EEI bit specified in the SEEI field.<br>1 Sets all bits in EEI |
| 5–4<br>Reserved | This field is reserved. |
| 3–0<br>SEEI | Set Enable Error Interrupt<br><br>Sets the corresponding bit in EEI |

## 22.3.7 Clear Enable Request Register (DMA_CERQ)

The CERQ provides a simple memory-mapped mechanism to clear a given bit in the ERQ to disable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ to be cleared. Setting the CAER bit provides a global clear function, forcing the entire contents of the ERQ to be cleared, disabling all DMA request inputs. If NOP is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 4000_8000h base + 1Ah offset = 4000_801Ah

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | 0 | | | 0 | | | |
| Write | NOP | CAER | 0 | | CERQ | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_CERQ field descriptions**

| Field | Description |
|---|---|
| 7<br>NOP | No Op enable<br><br>0 Normal operation<br>1 No operation, ignore the other bits in this register |
| 6<br>CAER | Clear All Enable Requests<br><br>0 Clear only the ERQ bit specified in the CERQ field<br>1 Clear all bits in ERQ |
| 5–4<br>Reserved | This field is reserved. |
| 3–0<br>CERQ | Clear Enable Request<br><br>Clears the corresponding bit in ERQ |

## 22.3.8 Set Enable Request Register (DMA_SERQ)

The SERQ provides a simple memory-mapped mechanism to set a given bit in the ERQ to enable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ to be set. Setting the SAER bit provides a global set function, forcing the entire contents of ERQ to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 4000_8000h base + 1Bh offset = 4000_801Bh

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | 0 | | | 0 | | | |
| Write | NOP | SAER | 0 | | SERQ | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_SERQ field descriptions**

| Field | Description |
|---|---|
| 7<br>NOP | No Op enable<br><br>0 Normal operation<br>1 No operation, ignore the other bits in this register |
| 6<br>SAER | Set All Enable Requests<br><br>0 Set only the ERQ bit specified in the SERQ field<br>1 Set all bits in ERQ |
| 5–4<br>Reserved | This field is reserved. |
| 3–0<br>SERQ | Set enable request<br><br>Sets the corresponding bit in ERQ |

## 22.3.9 Clear DONE Status Bit Register (DMA_CDNE)

The CDNE provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding transfer control descriptor to be cleared. Setting the CADN bit provides a global clear function, forcing all DONE bits to be cleared. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 4000_8000h base + 1Ch offset = 4000_801Ch

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | 0 | | | 0 | | | |
| Write | NOP | CADN | 0 | | CDNE | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_CDNE field descriptions**

| Field | Description |
|---|---|
| 7<br>NOP | No Op enable<br><br>0 Normal operation<br>1 No operation, ignore the other bits in this register |
| 6<br>CADN | Clears All DONE Bits<br><br>0 Clears only the TCDn_CSR[DONE] bit specified in the CDNE field<br>1 Clears all bits in TCDn_CSR[DONE] |
| 5–4<br>Reserved | This field is reserved. |
| 3–0<br>CDNE | Clear DONE Bit<br><br>Clears the corresponding bit in TCDn_CSR[DONE] |

## 22.3.10 Set START Bit Register (DMA_SSRT)

The SSRT provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding transfer control descriptor to be set. Setting the SAST bit provides a global set function, forcing all START bits to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 4000_8000h base + 1Dh offset = 4000_801Dh

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | 0 | | | 0 | | | |
| Write | NOP | SAST | 0 | | SSRT | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_SSRT field descriptions**

| Field | Description |
|---|---|
| 7<br>NOP | No Op enable<br><br>0 Normal operation<br>1 No operation, ignore the other bits in this register |
| 6<br>SAST | Set All START Bits (activates all channels)<br><br>0 Set only the TCDn_CSR[START] bit specified in the SSRT field<br>1 Set all bits in TCDn_CSR[START] |
| 5–4<br>Reserved | This field is reserved. |
| 3–0<br>SSRT | Set START Bit<br><br>Sets the corresponding bit in TCDn_CSR[START] |

## 22.3.11 Clear Error Register (DMA_CERR)

The CERR provides a simple memory-mapped mechanism to clear a given bit in the ERR to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the ERR to be cleared. Setting the CAEI bit provides a global clear function, forcing the ERR contents to be cleared, clearing all channel error indicators. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 4000_8000h base + 1Eh offset = 4000_801Eh

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | 0 | | | 0 | | | |
| Write | NOP | CAEI | 0 | | CERR | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_CERR field descriptions**

| Field | Description |
|---|---|
| 7<br>NOP | No Op enable<br><br>0 Normal operation<br>1 No operation, ignore the other bits in this register |
| 6<br>CAEI | Clear All Error Indicators<br><br>0 Clear only the ERR bit specified in the CERR field<br>1 Clear all bits in ERR |
| 5–4<br>Reserved | This field is reserved. |
| 3–0<br>CERR | Clear Error Indicator<br><br>Clears the corresponding bit in ERR |

## 22.3.12 Clear Interrupt Request Register (DMA_CINT)

The CINT provides a simple, memory-mapped mechanism to clear a given bit in the INT to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the INT to be cleared. Setting the CAIR bit provides a global clear function, forcing the entire contents of the INT to be cleared, disabling all DMA interrupt requests. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 4000_8000h base + 1Fh offset = 4000_801Fh

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | 0 | 0 | | | | | |
| Write | NOP | CAIR | 0 | | CINT | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_CINT field descriptions**

| Field | Description |
|---|---|
| 7<br>NOP | No Op enable<br><br>0 Normal operation<br>1 No operation, ignore the other bits in this register |
| 6<br>CAIR | Clear All Interrupt Requests<br><br>0 Clear only the INT bit specified in the CINT field<br>1 Clear all bits in INT |
| 5–4<br>Reserved | This field is reserved. |
| 3–0<br>CINT | Clear Interrupt Request<br><br>Clears the corresponding bit in INT |

## 22.3.13 Interrupt Request Register (DMA_INT)

The INT register provides a bit map for the 16 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptors, the eDMA engine generates an interrupt on data transfer completion. The outputs of this register are directly routed to the interrupt controller (INTC). During the interrupt-service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT register in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT register. On writes to INT, a 1 in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no affect on the corresponding channel's current interrupt status. The CINT register is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT register.

Address: 4000_8000h base + 24h offset = 4000_8024h

| Bit | 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | INT15 | INT14 | INT13 | INT12 | INT11 | INT10 | INT9 | INT8 | INT7 | INT6 | INT5 | INT4 | INT3 | INT2 | INT1 | INT0 |
| W | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_INT field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

*Table continues on the next page...*

## DMA_INT field descriptions (continued)

| Field | Description |
|---|---|
| 15<br>INT15 | Interrupt Request 15<br><br>0 The interrupt request for corresponding channel is cleared<br>1 The interrupt request for corresponding channel is active |
| 14<br>INT14 | Interrupt Request 14<br><br>0 The interrupt request for corresponding channel is cleared<br>1 The interrupt request for corresponding channel is active |
| 13<br>INT13 | Interrupt Request 13<br><br>0 The interrupt request for corresponding channel is cleared<br>1 The interrupt request for corresponding channel is active |
| 12<br>INT12 | Interrupt Request 12<br><br>0 The interrupt request for corresponding channel is cleared<br>1 The interrupt request for corresponding channel is active |
| 11<br>INT11 | Interrupt Request 11<br><br>0 The interrupt request for corresponding channel is cleared<br>1 The interrupt request for corresponding channel is active |
| 10<br>INT10 | Interrupt Request 10<br><br>0 The interrupt request for corresponding channel is cleared<br>1 The interrupt request for corresponding channel is active |
| 9<br>INT9 | Interrupt Request 9<br><br>0 The interrupt request for corresponding channel is cleared<br>1 The interrupt request for corresponding channel is active |
| 8<br>INT8 | Interrupt Request 8<br><br>0 The interrupt request for corresponding channel is cleared<br>1 The interrupt request for corresponding channel is active |
| 7<br>INT7 | Interrupt Request 7<br><br>0 The interrupt request for corresponding channel is cleared<br>1 The interrupt request for corresponding channel is active |
| 6<br>INT6 | Interrupt Request 6<br><br>0 The interrupt request for corresponding channel is cleared<br>1 The interrupt request for corresponding channel is active |
| 5<br>INT5 | Interrupt Request 5<br><br>0 The interrupt request for corresponding channel is cleared<br>1 The interrupt request for corresponding channel is active |
| 4<br>INT4 | Interrupt Request 4<br><br>0 The interrupt request for corresponding channel is cleared<br>1 The interrupt request for corresponding channel is active |

*Table continues on the next page...*

DMA_INT field descriptions (continued)

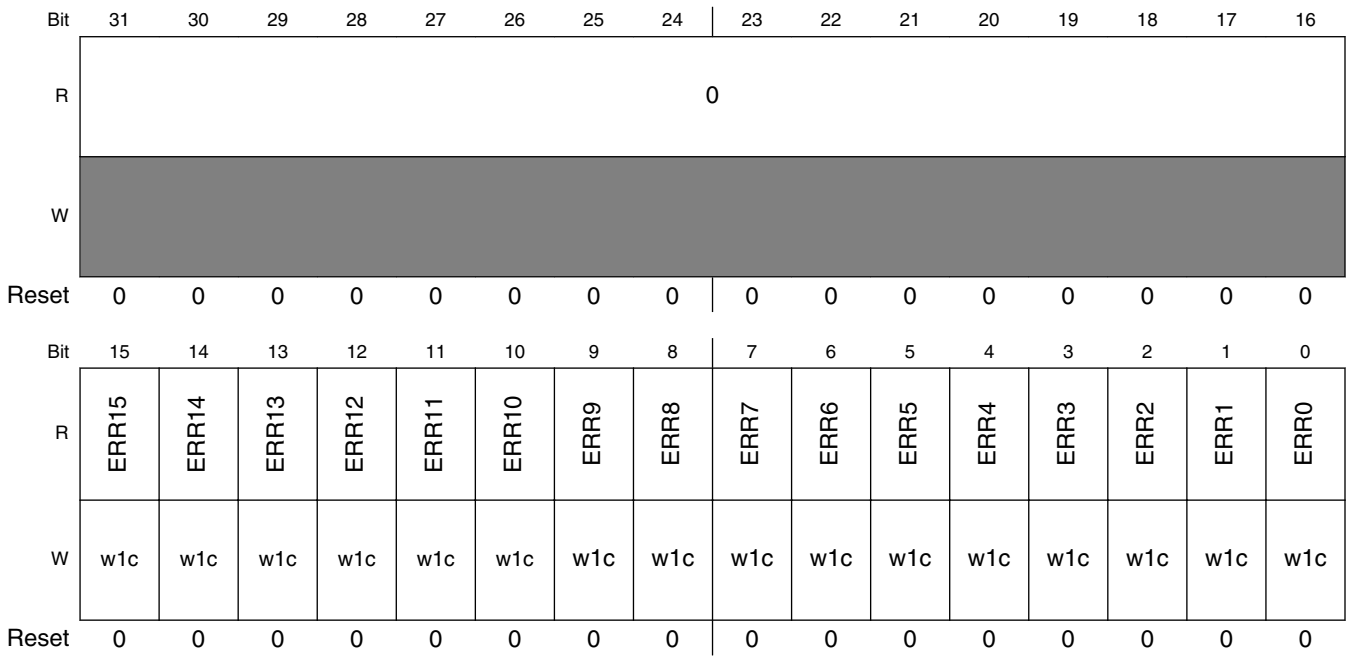
| Field | Description |
|---|---|
| 3<br>INT3 | Interrupt Request 3<br><br>0 The interrupt request for corresponding channel is cleared<br>1 The interrupt request for corresponding channel is active |
| 2<br>INT2 | Interrupt Request 2<br><br>0 The interrupt request for corresponding channel is cleared<br>1 The interrupt request for corresponding channel is active |
| 1<br>INT1 | Interrupt Request 1<br><br>0 The interrupt request for corresponding channel is cleared<br>1 The interrupt request for corresponding channel is active |
| 0<br>INT0 | Interrupt Request 0<br><br>0 The interrupt request for corresponding channel is cleared<br>1 The interrupt request for corresponding channel is active |

## 22.3.14 Error Register (DMA_ERR)

The ERR provides a bit map for the 16 channels, signaling the presence of an error for each channel. The eDMA engine signals the occurrence of an error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the EEI, and then routed to the interrupt controller. During the execution of the interrupt-service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. Typically, a write to the CERR in the interrupt-service routine is used for this purpose. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a non-zero value indicates the presence of a channel error regardless of the state of the EEI. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no affect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be cleared.

Address: 4000_8000h base + 2Ch offset = 4000_802Ch




## DMA_ERR field descriptions

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15<br>ERR15 | Error In Channel 15<br><br>0 An error in the corresponding channel has not occurred<br>1 An error in the corresponding channel has occurred |
| 14<br>ERR14 | Error In Channel 14<br><br>0 An error in the corresponding channel has not occurred<br>1 An error in the corresponding channel has occurred |
| 13<br>ERR13 | Error In Channel 13<br><br>0 An error in the corresponding channel has not occurred<br>1 An error in the corresponding channel has occurred |
| 12<br>ERR12 | Error In Channel 12<br><br>0 An error in the corresponding channel has not occurred<br>1 An error in the corresponding channel has occurred |
| 11<br>ERR11 | Error In Channel 11<br><br>0 An error in the corresponding channel has not occurred<br>1 An error in the corresponding channel has occurred |
| 10<br>ERR10 | Error In Channel 10<br><br>0 An error in the corresponding channel has not occurred<br>1 An error in the corresponding channel has occurred |

*Table continues on the next page...*

## DMA_ERR field descriptions (continued)

| Field | Description |
|---|---|
| 9<br>ERR9 | Error In Channel 9<br><br>0 An error in the corresponding channel has not occurred<br>1 An error in the corresponding channel has occurred |
| 8<br>ERR8 | Error In Channel 8<br><br>0 An error in the corresponding channel has not occurred<br>1 An error in the corresponding channel has occurred |
| 7<br>ERR7 | Error In Channel 7<br><br>0 An error in the corresponding channel has not occurred<br>1 An error in the corresponding channel has occurred |
| 6<br>ERR6 | Error In Channel 6<br><br>0 An error in the corresponding channel has not occurred<br>1 An error in the corresponding channel has occurred |
| 5<br>ERR5 | Error In Channel 5<br><br>0 An error in the corresponding channel has not occurred<br>1 An error in the corresponding channel has occurred |
| 4<br>ERR4 | Error In Channel 4<br><br>0 An error in the corresponding channel has not occurred<br>1 An error in the corresponding channel has occurred |
| 3<br>ERR3 | Error In Channel 3<br><br>0 An error in the corresponding channel has not occurred<br>1 An error in the corresponding channel has occurred |
| 2<br>ERR2 | Error In Channel 2<br><br>0 An error in the corresponding channel has not occurred<br>1 An error in the corresponding channel has occurred |
| 1<br>ERR1 | Error In Channel 1<br><br>0 An error in the corresponding channel has not occurred<br>1 An error in the corresponding channel has occurred |
| 0<br>ERR0 | Error In Channel 0<br><br>0 An error in the corresponding channel has not occurred<br>1 An error in the corresponding channel has occurred |

## 22.3.15 Hardware Request Status Register (DMA_HRS)

The HRS register provides a bit map for the DMA channels, signaling the presence of a hardware request for each channel. The hardware request status bits reflect the current state of the register and qualified (via the ERQ fields) DMA request signals as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

**NOTE**

These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ bits.

Address: 4000_8000h base + 34h offset = 4000_8034h

| Bit | 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | HRS15 | HRS14 | HRS13 | HRS12 | HRS11 | HRS10 | HRS9 | HRS8 | HRS7 | HRS6 | HRS5 | HRS4 | HRS3 | HRS2 | HRS1 | HRS0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_HRS field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | Reserved<br><br>This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15<br>HRS15 | Hardware Request Status Channel 15<br><br>0 A hardware service request for the corresponding channel is not present<br>1 A hardware service request for the corresponding channel is present |
| 14<br>HRS14 | Hardware Request Status Channel 14<br><br>0 A hardware service request for the corresponding channel is not present<br>1 A hardware service request for the corresponding channel is present |
| 13<br>HRS13 | Hardware Request Status Channel 13<br><br>0 A hardware service request for the corresponding channel is not present<br>1 A hardware service request for the corresponding channel is present |

*Table continues on the next page...*

## DMA_HRS field descriptions (continued)

| Field | Description |
|---|---|
| 12<br>HRS12 | Hardware Request Status Channel 12<br><br>0 A hardware service request for the corresponding channel is not present<br>1 A hardware service request for the corresponding channel is present |
| 11<br>HRS11 | Hardware Request Status Channel 11<br><br>0 A hardware service request for the corresponding channel is not present<br>1 A hardware service request for the corresponding channel is present |
| 10<br>HRS10 | Hardware Request Status Channel 10<br><br>0 A hardware service request for the corresponding channel is not present<br>1 A hardware service request for the corresponding channel is present |
| 9<br>HRS9 | Hardware Request Status Channel 9<br><br>0 A hardware service request for the corresponding channel is not present<br>1 A hardware service request for the corresponding channel is present |
| 8<br>HRS8 | Hardware Request Status Channel 8<br><br>0 A hardware service request for the corresponding channel is not present<br>1 A hardware service request for the corresponding channel is present |
| 7<br>HRS7 | Hardware Request Status Channel 7<br><br>0 A hardware service request for the corresponding channel is not present<br>1 A hardware service request for the corresponding channel is present |
| 6<br>HRS6 | Hardware Request Status Channel 6<br><br>0 A hardware service request for the corresponding channel is not present<br>1 A hardware service request for the corresponding channel is present |
| 5<br>HRS5 | Hardware Request Status Channel 5<br><br>0 A hardware service request for the corresponding channel is not present<br>1 A hardware service request for the corresponding channel is present |
| 4<br>HRS4 | Hardware Request Status Channel 4<br><br>0 A hardware service request for the corresponding channel is not present<br>1 A hardware service request for the corresponding channel is present |
| 3<br>HRS3 | Hardware Request Status Channel 3<br><br>0 A hardware service request for the corresponding channel is not present<br>1 A hardware service request for the corresponding channel is present |
| 2<br>HRS2 | Hardware Request Status Channel 2<br><br>0 A hardware service request for the corresponding channel is not present<br>1 A hardware service request for the corresponding channel is present |
| 1<br>HRS1 | Hardware Request Status Channel 1<br><br>0 A hardware service request for the corresponding channel is not present<br>1 A hardware service request for the corresponding channel is present |

*Table continues on the next page...*

**DMA_HRS field descriptions (continued)**

| Field | Description |
|---|---|
| 0<br>HRS0 | Hardware Request Status Channel 0<br><br>0 A hardware service request for the corresponding channel is not present<br>1 A hardware service request for the corresponding channel is present |

## 22.3.16 Channel n Priority Register (DMA_DCHPRI*n*)

When fixed-priority channel arbitration is enabled (CR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel . The channel priorities are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next priority, then 2, 3, etc. Software must program the channel priorities with unique values; otherwise, a configuration error is reported. The range of the priority value is limited to the values of 0 through 15.

Address: 4000_8000h base + 100h offset + (1d × i), where i=0d to 15d

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | ECP | DPA | 0 | | CHPRI | | | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | * | * | * | * |

* Notes:
- CHPRI field: See bit field description

**DMA_DCHPRI*n* field descriptions**

| Field | Description |
|---|---|
| 7<br>ECP | Enable Channel Preemption<br><br>0 Channel n cannot be suspended by a higher priority channel's service request<br>1 Channel n can be temporarily suspended by the service request of a higher priority channel |
| 6<br>DPA | Disable Preempt Ability<br><br>0 Channel n can suspend a lower priority channel<br>1 Channel n cannot suspend any channel, regardless of channel priority |
| 5–4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3–0<br>CHPRI | Channel n Arbitration Priority<br><br>Channel priority when fixed-priority arbitration is enabled<br><br>**NOTE:** Reset value for the channel priority fields, CHPRI, is equal to the corresponding channel number for each priority register, i.e., DCHPRI15[CHPRI] equals 0b1111. |

## 22.3.17 TCD Source Address (DMA_TCD*n*_SADDR)

Address: 4000_8000h base + 1000h offset + (32d × i), where i=0d to 15d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SADDR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

### DMA_TCD*n*_SADDR field descriptions

| Field | Description |
|---|---|
| 31–0<br>SADDR | Source Address<br><br>Memory address pointing to the source data. |

## 22.3.18 TCD Signed Source Address Offset (DMA_TCD*n*_SOFF)

Address: 4000_8000h base + 1004h offset + (32d × i), where i=0d to 15d

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read<br>Write | SOFF | | | | | | | | | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

### DMA_TCD*n*_SOFF field descriptions

| Field | Description |
|---|---|
| 15–0<br>SOFF | Source address signed offset<br><br>Sign-extended offset applied to the current source address to form the next-state value as each source read is completed. |

## 22.3.19 TCD Transfer Attributes (DMA_TCD*n*_ATTR)

Address: 4000_8000h base + 1006h offset + (32d × i), where i=0d to 15d

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read<br>Write | SMOD | | | | | SSIZE | | | DMOD | | | | | DSIZE | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

### DMA_TCD*n*_ATTR field descriptions

| Field | Description |
|---|---|
| 15–11<br>SMOD | Source Address Modulo.<br><br>0 Source address modulo feature is disabled<br>≠0 This value defines a specific address range specified to be the value after SADDR + SOFF calculation is performed or the original register value. The setting of this field provides the ability to implement a circular data queue easily. For data queues requiring power-of-2 size bytes, the queue should start at a 0-modulo-size address and the SMOD field should be set to the appropriate value for the queue, freezing the desired number of upper address bits. The value programmed into this field specifies the number of lower address bits allowed to change. For a circular queue application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a 0-modulo-size range. |
| 10–8<br>SSIZE | Source data transfer size<br><br>The attempted use of a Reserved encoding causes a configuration error.<br><br>000 8-bit<br>001 16-bit<br>010 32-bit<br>011 Reserved<br>100 16-byte<br>101 32-byte<br>110 Reserved<br>111 Reserved |
| 7–3<br>DMOD | Destination Address Modulo<br><br>See the SMOD definition |
| 2–0<br>DSIZE | Destination Data Transfer Size<br><br>See the SSIZE definition |

## 22.3.20 TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD*n*_NBYTES_MLNO)

This register, or one of the next two registers (TCD_NBYTES_MLOFFNO, TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is disabled (CR[EMLM] = 0)

If minor loop mapping is enabled, see the TCD_NBYTES_MLOFFNO and TCD_NBYTES_MLOFFYES register descriptions for TCD word 2's definition.

Address: 4000_8000h base + 1008h offset + (32d × i), where i=0d to 15d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NBYTES | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:

- x = Undefined at reset.

**DMA_TCD*n*_NBYTES_MLNO field descriptions**

| Field | Description |
|---|---|
| 31–0<br>NBYTES | Minor Byte Transfer Count<br><br>Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. (Although, it may be stalled by using the bandwidth control field, or via preemption.) After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.<br><br>**NOTE:** An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer. |

## 22.3.21 TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD*n*_NBYTES_MLOFFNO)

One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- SMLOE = 0 and DMLOE = 0

If minor loop mapping is enabled and SMLOE or DMLOE is set, then refer to the TCD_NBYTES_MLOFFYES register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

Address: 4000_8000h base + 1008h offset + (32d × i), where i=0d to 15d

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SMLOE | DMLOE | NBYTES | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | NBYTES | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

### DMA_TCD*n*_NBYTES_MLOFFNO field descriptions

| Field | Description |
|---|---|
| 31<br>SMLOE | Source Minor Loop Offset Enable<br><br>Selects whether the minor loop offset is applied to the source address upon minor loop completion.<br><br>0 The minor loop offset is not applied to the SADDR<br>1 The minor loop offset is applied to the SADDR |
| 30<br>DMLOE | Destination Minor Loop Offset enable<br><br>Selects whether the minor loop offset is applied to the destination address upon minor loop completion.<br><br>0 The minor loop offset is not applied to the DADDR<br>1 The minor loop offset is applied to the DADDR |
| 29–0<br>NBYTES | Minor Byte Transfer Count<br><br>Number of bytes to be transferred in each service request of the channel.<br><br>As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted; although, it may be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. |

## 22.3.22 TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD*n*_NBYTES_MLOFFYES)

One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFNO), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- Minor loop offset is enabled (SMLOE or DMLOE = 1)

If minor loop mapping is enabled and SMLOE and DMLOE are cleared, then refer to the TCD_NBYTES_MLOFFNO register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

Address: 4000_8000h base + 1008h offset + (32d × i), where i=0d to 15d


* Notes:
- x = Undefined at reset.

**DMA_TCD*n*_NBYTES_MLOFFYES field descriptions**

| Field | Description |
|---|---|
| 31<br>SMLOE | Source Minor Loop Offset Enable<br><br>Selects whether the minor loop offset is applied to the source address upon minor loop completion.<br><br>0 The minor loop offset is not applied to the SADDR<br>1 The minor loop offset is applied to the SADDR |
| 30<br>DMLOE | Destination Minor Loop Offset enable<br><br>Selects whether the minor loop offset is applied to the destination address upon minor loop completion. |

*Table continues on the next page...*

**DMA_TCD*n*_NBYTES_MLOFFYES field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 The minor loop offset is not applied to the DADDR<br>1 The minor loop offset is applied to the DADDR |
| 29–10<br>MLOFF | If SMLOE or DMLOE is set, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes. |
| 9–0<br>NBYTES | Minor Byte Transfer Count<br>Number of bytes to be transferred in each service request of the channel.<br>As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. (Although, it may be stalled by using the bandwidth control field, or via preemption.) After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. |

## 22.3.23 TCD Last Source Address Adjustment (DMA_TCD*n*_SLAST)

Address: 4000_8000h base + 100Ch offset + (32d × i), where i=0d to 15d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SLAST | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**DMA_TCD*n*_SLAST field descriptions**

| Field | Description |
|---|---|
| 31–0<br>SLAST | Last source Address Adjustment<br>Adjustment value added to the source address at the completion of the major iteration count. This value can be applied to restore the source address to the initial value, or adjust the address to reference the next data structure.<br>This register uses two's complement notation; the overflow bit is discarded. |

## 22.3.24 TCD Destination Address (DMA_TCD*n*_DADDR)

Address: 4000_8000h base + 1010h offset + (32d × i), where i=0d to 15d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DADDR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**DMA_TCD*n*_DADDR field descriptions**

| Field | Description |
|---|---|
| 31–0<br>DADDR | Destination Address<br><br>Memory address pointing to the destination data. |

## 22.3.25 TCD Signed Destination Address Offset (DMA_TCD*n*_DOFF)

Address: 4000_8000h base + 1014h offset + (32d × i), where i=0d to 15d

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read<br>Write | DOFF | | | | | | | | | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**DMA_TCD*n*_DOFF field descriptions**

| Field | Description |
|---|---|
| 15–0<br>DOFF | Destination Address Signed offset<br><br>Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed. |

## 22.3.26 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_CITER_ELINKYES)

If TCDn_CITER[ELINK] is set, the TCDn_CITER register is defined as follows.

Address: 4000_8000h base + 1016h offset + (32d × i), where i=0d to 15d

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | ELINK | 0 | | | LINKCH | | | CITER |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | CITER | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

## DMA_TCD*n*_CITER_ELINKYES field descriptions

| Field | Description |
|---|---|
| 15<br>ELINK | Enable channel-to-channel linking on minor-loop complete<br><br>As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel.<br><br>If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking.<br><br>**NOTE:** This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported.<br><br>0 The channel-to-channel linking is disabled<br>1 The channel-to-channel linking is enabled |
| 14–13<br>Reserved | This field is reserved. |
| 12–9<br>LINKCH | Link Channel Number<br><br>If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request to the channel defined by these four bits by setting that channel's TCDn_CSR[START] bit. |
| 8–0<br>CITER | Current Major Iteration Count<br><br>This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations (e.g., final source and destination address calculations), optionally generating an interrupt to signal channel completion before reloading the CITER field from the beginning iteration count (BITER) field.<br><br>**NOTE:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field.<br><br>**NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001. |

## 22.3.27 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_CITER_ELINKNO)

If TCDn_CITER[ELINK] is cleared, the TCDn_CITER register is defined as follows.

Address: 4000_8000h base + 1016h offset + (32d × i), where i=0d to 15d

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | ELINK | CITER | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | CITER | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**DMA_TCD*n*_CITER_ELINKNO field descriptions**

| Field | Description |
|---|---|
| 15<br>ELINK | Enable channel-to-channel linking on minor-loop complete<br><br>As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel.<br><br>If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking.<br><br>**NOTE:** This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported.<br><br>0 The channel-to-channel linking is disabled<br>1 The channel-to-channel linking is enabled |
| 14–0<br>CITER | Current Major Iteration Count<br><br>This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations (e.g., final source and destination address calculations), optionally generating an interrupt to signal channel completion before reloading the CITER field from the beginning iteration count (BITER) field.<br><br>**NOTE:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field.<br><br>**NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001. |

## 22.3.28 TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD*n*_DLASTSGA)

Address: 4000_8000h base + 1018h offset + (32d × i), where i=0d to 15d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DLASTSGA | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**DMA_TCD*n*_DLASTSGA field descriptions**

| Field | Description |
|---|---|
| 31–0<br>DLASTSGA | Destination last address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather).<br><br>If (TCDn_CSR[ESG] = 0), then:<br>• Adjustment value added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure.<br>• This field uses two's complement notation for the final destination address adjustment.<br><br>Otherwise:<br>• This address points to the beginning of a 0-modulo-32-byte region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32-byte, else a configuration error is reported. |

## 22.3.29 TCD Control and Status (DMA_TCD*n*_CSR)

Address: 4000_8000h base + 101Ch offset + (32d × i), where i=0d to 15d

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | BWC | | 0 | | MAJORLINKCH | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | DONE | ACTIVE | MAJORELINK | ESG | DREQ | INTHALF | INTMAJOR | START |
| Reset | 0 | 0 | x* | x* | x* | x* | x* | 0 |

* Notes:
- x = Undefined at reset.

## DMA_TCD*n*_CSR field descriptions

| Field | Description |
|---|---|
| 15–14<br>BWC | Bandwidth Control<br><br>Throttles the amount of bus bandwidth consumed by the eDMA. In general, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the crossbar switch.<br><br>**NOTE:** If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency.<br><br>00 No eDMA engine stalls<br>01 Reserved<br>10 eDMA engine stalls for 4 cycles after each r/w<br>11 eDMA engine stalls for 8 cycles after each r/w |
| 13–12<br>Reserved | This field is reserved. |
| 11–8<br>MAJORLINKCH | Link Channel Number<br><br>If (MAJORELINK = 0) then<br>• No channel-to-channel linking (or chaining) is performed after the major loop counter is exhausted.<br><br>else<br>• After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by these six bits by setting that channel's TCDn_CSR[START] bit. |
| 7<br>DONE | Channel Done<br><br>This flag indicates the eDMA has completed the major loop. The eDMA engine sets it as the CITER count reaches zero; The software clears it, or the hardware when the channel is activated.<br><br>**NOTE:** This bit must be cleared to write the MAJORELINK or ESG bits. |
| 6<br>ACTIVE | Channel Active<br><br>This flag signals the channel is currently in execution. It is set when channel service begins, and the eDMA clears it as the minor loop completes or if any error condition is detected. This bit resets to zero. |
| 5<br>MAJORELINK | Enable channel-to-channel linking on major loop complete<br><br>As the channel completes the major loop, this flag enables the linking to another channel, defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel.<br><br>**NOTE:** To support the dynamic linking coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set.<br><br>0 The channel-to-channel linking is disabled<br>1 The channel-to-channel linking is enabled |
| 4<br>ESG | Enable Scatter/Gather Processing<br><br>As the channel completes the major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLASTSGA as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure loaded as the transfer control descriptor into the local memory.<br><br>**NOTE:** To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set. |

*Table continues on the next page...*

## DMA_TCD*n*_CSR field descriptions (continued)

| Field | Description |
|---|---|
| | 0 The current channel's TCD is normal format.<br>1 The current channel's TCD specifies a scatter gather format. The DLASTSGA field provides a memory pointer to the next TCD to be loaded into this channel after the major loop completes its execution. |
| 3<br>DREQ | Disable Request<br><br>If this flag is set, the eDMA hardware automatically clears the corresponding ERQ bit when the current major iteration count reaches zero.<br><br>0 The channel's ERQ bit is not affected<br>1 The channel's ERQ bit is cleared when the major loop is complete |
| 2<br>INTHALF | Enable an interrupt when major counter is half complete.<br><br>If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT register when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered (aka ping-pong) schemes or other types of data movement where the processor needs an early indication of the transfer's progress. If BITER is set, do not use INTHALF. Use INTMAJOR instead.<br><br>0 The half-point interrupt is disabled<br>1 The half-point interrupt is enabled |
| 1<br>INTMAJOR | Enable an interrupt when major iteration count completes<br><br>If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches zero.<br><br>0 The end-of-major loop interrupt is disabled<br>1 The end-of-major loop interrupt is enabled |
| 0<br>START | Channel Start<br><br>If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution.<br><br>0 The channel is not explicitly started<br>1 The channel is explicitly started via a software initiated service request |

## 22.3.30 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_BITER_ELINKYES)

If the TCDn_BITER[ELINK] bit is set, the TCDn_BITER register is defined as follows.

Address: 4000_8000h base + 101Eh offset + (32d × i), where i=0d to 15d

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Read / Write | ELINK | 0 | | | LINKCH | | | BITER |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read / Write | BITER | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

### DMA_TCD*n*_BITER_ELINKYES field descriptions

| Field | Description |
|---|---|
| 15<br>ELINK | Enables channel-to-channel linking on minor loop complete<br><br>As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking.<br><br>**NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field.<br><br>0 The channel-to-channel linking is disabled<br>1 The channel-to-channel linking is enabled |
| 14–13<br>Reserved | This field is reserved. |
| 12–9<br>LINKCH | Link Channel Number<br><br>If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by these four bits by setting that channel's TCDn_CSR[START] bit.<br><br>**NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. |
| 8–0<br>BITER | Starting Major Iteration Count<br><br>As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. |

*Table continues on the next page...*

**DMA_TCD*n*_BITER_ELINKYES field descriptions (continued)**

| Field | Description |
|---|---|
| | **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001. |

## 22.3.31 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_BITER_ELINKNO)

If the TCDn_BITER[ELINK] bit is cleared, the TCDn_BITER register is defined as follows.

Address: 4000_8000h base + 101Eh offset + (32d × i), where i=0d to 15d

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | ELINK | BITER | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | BITER | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**DMA_TCD*n*_BITER_ELINKNO field descriptions**

| Field | Description |
|---|---|
| 15<br>ELINK | Enables channel-to-channel linking on minor loop complete<br><br>As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking.<br><br>**NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field.<br><br>0 The channel-to-channel linking is disabled<br>1 The channel-to-channel linking is enabled |
| 14–0<br>BITER | Starting Major Iteration Count<br><br>As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. |

*Table continues on the next page...*

**DMA_TCD*n*_BITER_ELINKNO field descriptions (continued)**

| Field | Description |
|---|---|
| | **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001. |

# 22.4 Functional description

## 22.4.1 eDMA basic data flow

The basic flow of a data transfer can be partitioned into three segments.

As shown in the following diagram, the first segment involves the channel activation:

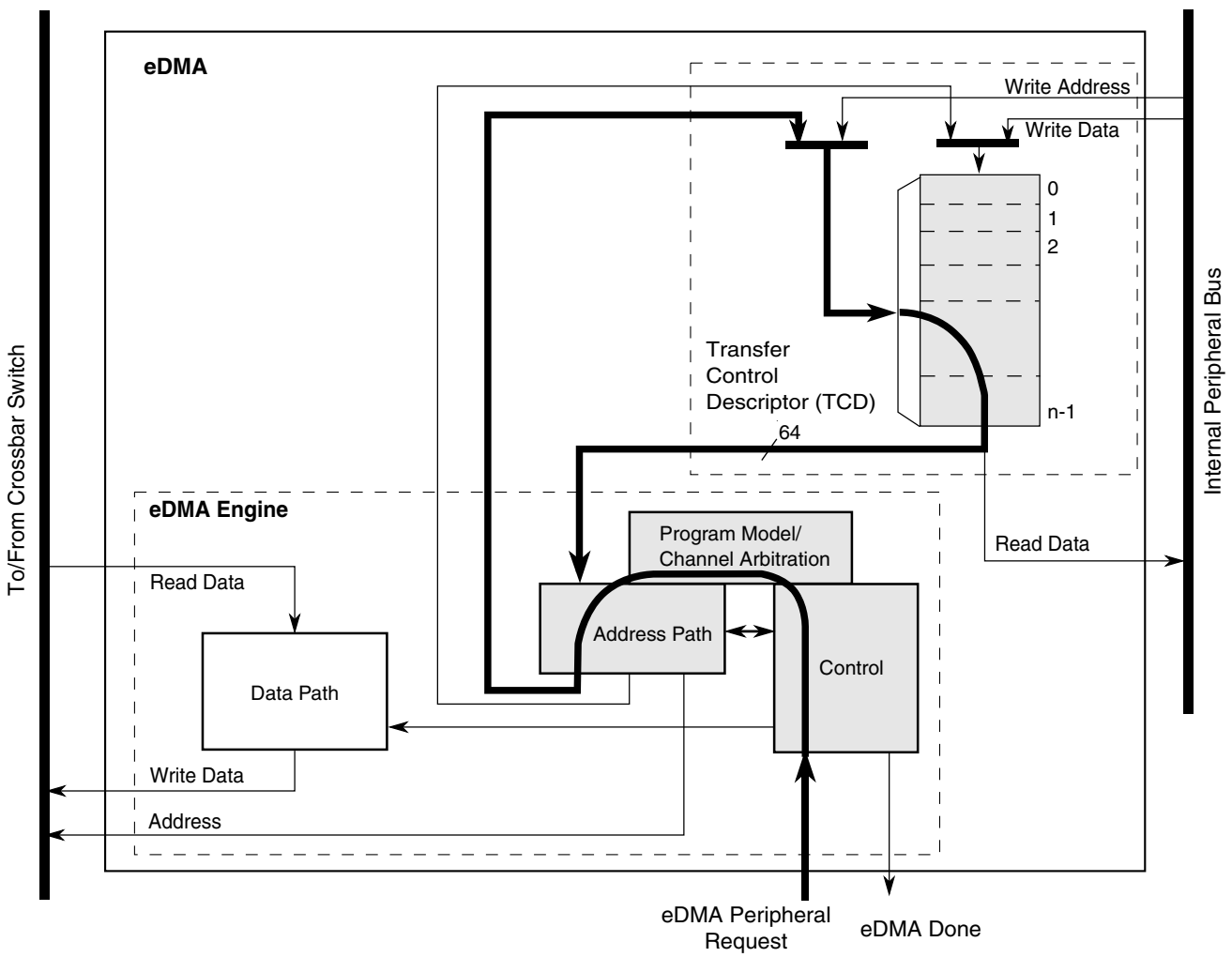



**Figure 22-289. eDMA operation, part 1**

This example uses the assertion of the eDMA peripheral request signal to request service for channel *n*. Channel activation via software and the TCD*n*_CSR[START] bit follows the same basic flow as peripheral requests. The eDMA request input signal is registered internally and then routed through the eDMA engine: first through the control module, then into the program model and channel arbitration. In the next cycle, the channel arbitration performs, using the fixed-priority or round-robin algorithm. After arbitration is complete, the activated channel number is sent through the address path and converted into the required address to access the local memory for TCD*n*. Next, the TCD memory is accessed and the required descriptor read from the local memory and loaded into the eDMA engine address path channel x or y registers. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the address path channel x or y registers.

The following diagram illustrates the second part of the basic data flow:

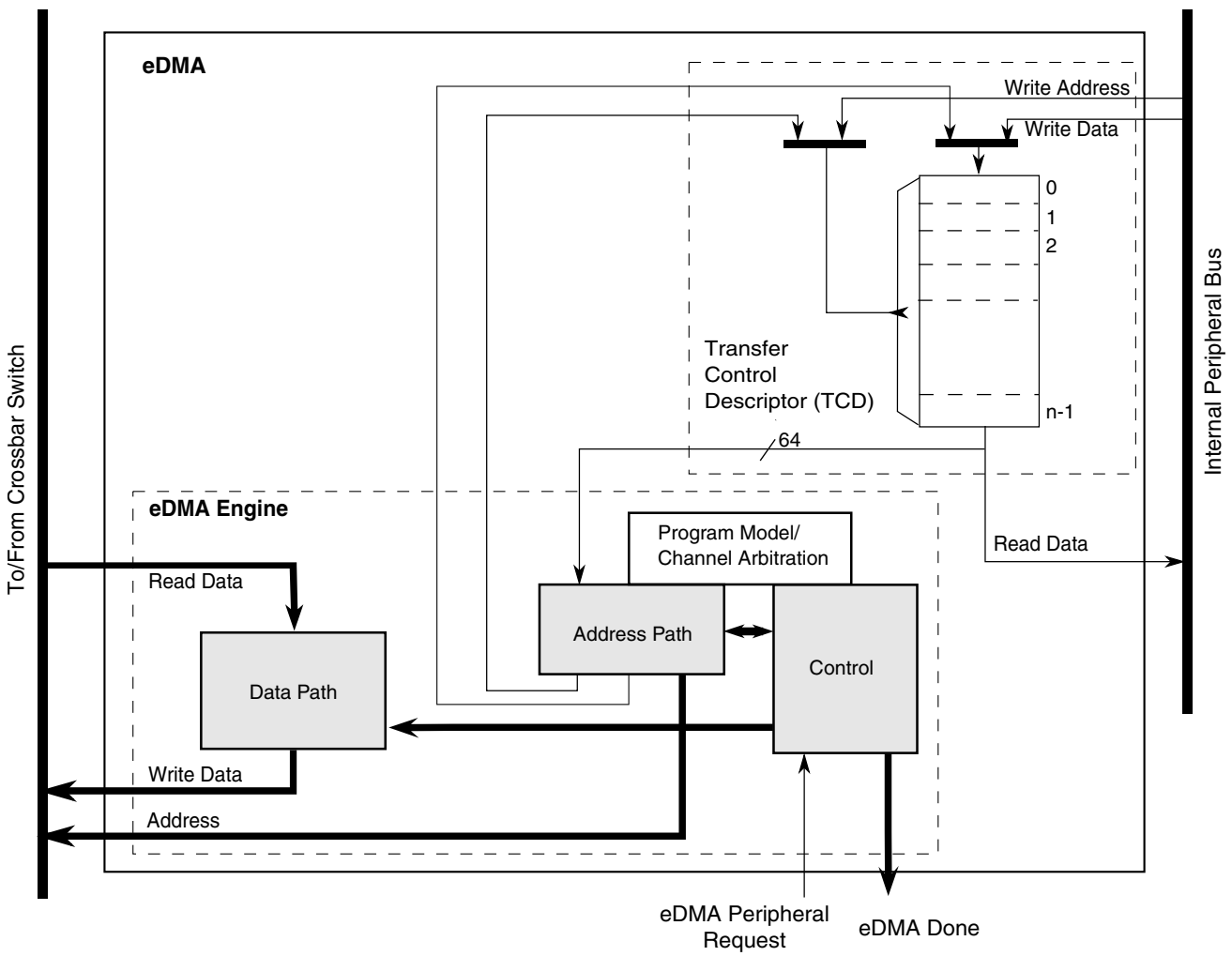



**Figure 22-290. eDMA operation, part 2**

The modules associated with the data transfer (address path, data path, and control) sequence through the required source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily stored in the data path block until it is gated onto the internal bus during the destination write. This source read/destination write processing continues until the minor byte count has transferred.

After the minor byte count has moved, the final phase of the basic data flow is performed. In this segment, the address path logic performs the required updates to certain fields in the appropriate TCD, e.g., SADDR, DADDR, CITER. If the major iteration count is exhausted, additional operations are performed. These include the final address adjustments and reloading of the BITER field into the CITER. Assertion of an optional interrupt request also occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor (if scatter/gather is enabled). The updates to the TCD memory and the assertion of an interrupt request are shown in the following diagram.

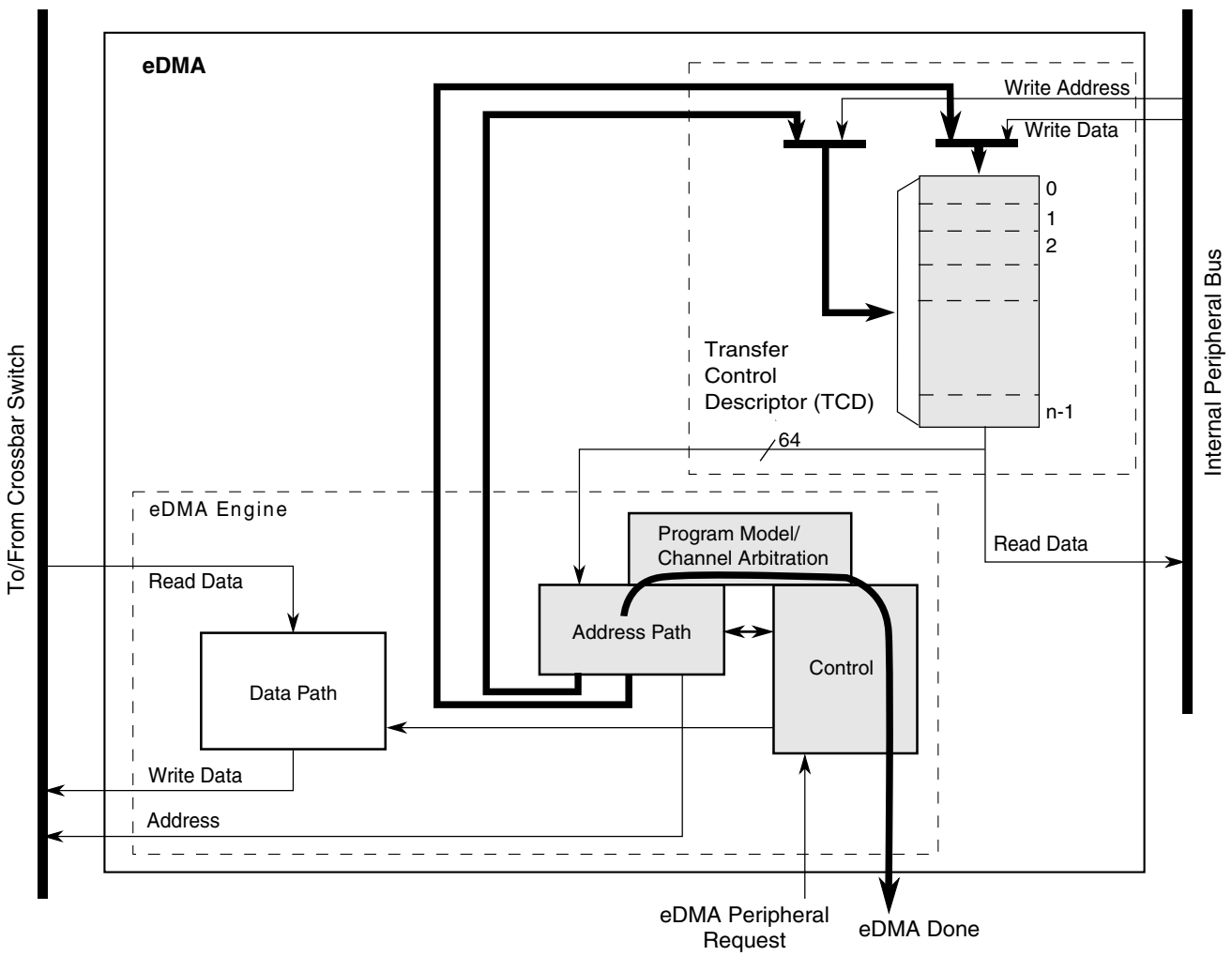



**Figure 22-291. eDMA operation, part 3**

### 22.4.2 Error reporting and handling

Channel errors are reported in the Error Status register (DMAx_ES) and can be caused by:

- A configuration error, which is an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed-Arbitration mode, or
- An error termination to a bus master read or write cycle

A configuration error is reported when the starting source or destination address, source or destination offsets, minor loop byte count, or the transfer size represent an inconsistent state. Each of these possible causes are detailed below:

- The addresses and offsets must be aligned on 0-modulo-transfer-size boundaries.
- The minor loop byte count must be a multiple of the source and destination transfer sizes.
- All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively.
- In fixed arbitration mode, a configuration error is caused by any two channel priorities being equal. All channel priority levels must be unique when fixed arbitration mode is enabled.
- If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary.
- If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCDn_CITER[E_LINK] bit does not equal the TCDn_BITER[E_LINK] bit.

If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, report as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and current iteration count at the point of the fault. When a system bus error occurs, the channel terminates after the next transfer. Due to pipeline effect, the next transfer is already in progress when the bus error is received by the eDMA. If a bus error occurs on the last read prior to beginning the write sequence, the write executes using the

data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.

A transfer may be cancelled by software with the CR[CX] bit. When a cancel transfer request is recognized, the DMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the Error Status register (DMAx_ES) is updated with the cancelled channel number and ECX is set. The TCD of a cancelled channel contains the source and destination addresses of the last transfer saved in the TCD. If the channel needs to be restarted, you must re-initialize the TCD because the aforementioned fields no longer represent the original parameters. When a transfer is cancelled by the error cancel transfer mechanism, the channel number is loaded into DMA_ES[ERRCHN] and ECX and VLD are set. In addition, an error interrupt may be generated if enabled.

The occurrence of any error causes the eDMA engine to stop normal processing of the active channel immediately (it goes to its error processing states and the transaction to the system bus still has peipeline effect), and the appropriate channel bit in the eDMA error register is asserted. At the same time, the details of the error condition are loaded into the Error Status register (DMAx_ES). The major loop complete indicators, setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request, are not affected when an error is detected. After the error status has been updated, the eDMA engine continues operating by servicing the next appropriate channel. A channel that experiences an error condition is not automatically disabled. If a channel is terminated by an error and then issues another service request before the error is fixed, that channel executes and terminates with the same error condition.

### 22.4.3 Channel preemption

Channel preemption is enabled on a per-channel basis by setting the DCHPRIn[ECP] bit. Channel preemption allows the executing channel's data transfers to temporarily suspend in favor of starting a higher priority channel. After the preempting channel has completed all its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel is suspended and the higher priority channel is serviced. Nested preemption, that is, attempting to preempt a preempting channel, is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is available only when fixed arbitration is selected.

A channel's ability to preempt another channel can be disabled by setting DCHPRIn[DPA]. When a channel's preempt ability is disabled, that channel cannot suspend a lower priority channel's data transfer, regardless of the lower priority channel's ECP setting. This allows for a pool of low priority, large data-moving channels to be defined. These low priority channels can be configured to not preempt each other, thus preventing a low priority channel from consuming the preempt slot normally available to a true, high priority channel.

## 22.4.4 Performance

This section addresses the performance of the eDMA module, focusing on two separate metrics:

- In the traditional data movement context, performance is best expressed as the peak data transfer rates achieved using the eDMA. In most implementations, this transfer rate is limited by the speed of the source and destination address spaces.
- In a second context where device-paced movement of single data values to/from peripherals is dominant, a measure of the requests that can be serviced in a fixed time is a more relevant metric. In this environment, the speed of the source and destination address spaces remains important. However, the microarchitecture of the eDMA also factors significantly into the resulting metric.

### 22.4.4.1 Peak transfer rates

The peak transfer rates for several different source and destination transfers are shown in the following tables. These tables assume:

- Internal SRAM can be accessed with zero wait-states when viewed from the system bus data phase
- All internal peripheral bus reads require two wait-states, and internal peripheral bus writes three wait-states, when viewed from the system bus data phase
- All internal peripheral bus accesses are 32-bits in size

This table compares peak transfer rates based n different possible system speeds. Specific chips/devices may not support all system speeds listed.

**Table 22-292. eDMA peak transfer rates (Mbytes/sec)**

| System Speed, Width | Internal SRAM-to-Internal SRAM | 32b internal peripheral bus-to-Internal SRAM | Internal SRAM-to-32b internal peripheral bus |
|---|---|---|---|
| 66.7 MHz, 32b | 133.3 | 66.7 | 53.3 |
| 83.3 MHz, 32b | 166.7 | 83.3 | 66.7 |
| 100.0 MHz, 32b | 200.0 | 100.0 | 80.0 |
| 133.3 MHz, 32b | 266.7 | 133.3 | 106.7 |
| 150.0 MHz, 32b | 300.0 | 150.0 | 120.0 |

Internal-SRAM-to-internal-SRAM transfers occur at the core's datapath width. For all transfers involving the internal peripheral bus, 32-bit transfer sizes are used. In all cases, the transfer rate includes the time to read the source plus the time to write the destination.

### 22.4.4.2 Peak request rates

The second performance metric is a measure of the number of DMA requests that can be serviced in a given amount of time. For this metric, assume that the peripheral request causes the channel to move a single internal peripheral bus-mapped operand to/from internal SRAM. The same timing assumptions used in the previous example apply to this calculation. In particular, this metric also reflects the time required to activate the channel.

The eDMA design supports the following hardware service request sequence. Note that the exact timing from Cycle 7 is a function of the response times for the channel's read and write accesses. In the case of an internal peripheral bus read and internal SRAM write, the combined data phase time is 4 cycles. For an SRAM read and internal peripheral bus write, it is 5 cycles.

**Table 22-293. Hardware service request process**

| Cycle | | Description |
|---|---|---|
| With internal peripheral bus read and internal SRAM write | With SRAM read and internal peripheral bus write | |
| 1 | | eDMA peripheral request is asserted. |
| 2 | | The eDMA peripheral request is registered locally in the eDMA module and qualified. TCD*n*_CSR[START] bit initiated requests start at this point with the registering of the user write to TCD*n* word 7. |
| 3 | | Channel arbitration begins. |
| 4 | | Channel arbitration completes. The transfer control descriptor local memory read is initiated. |

*Table continues on the next page...*

**Table 22-293. Hardware service request process (continued)**

| Cycle | | Description |
|---|---|---|
| With internal peripheral bus read and internal SRAM write | With SRAM read and internal peripheral bus write | |
| 5–6 | | The first two parts of the activated channel's TCD is read from the local memory. The memory width to the eDMA engine is 64 bits, so the entire descriptor can be accessed in four cycles |
| 7 | | The first system bus read cycle is initiated, as the third part of the channel's TCD is read from the local memory. Depending on the state of the crossbar switch, arbitration at the system bus may insert an additional cycle of delay here. |
| 8–11 | 8–12 | The last part of the TCD is read in. This cycle represents the first data phase for the read, and the address phase for the destination write. |
| 12 | 13 | This cycle represents the data phase of the last destination write. |
| 13 | 14 | The eDMA engine completes the execution of the inner minor loop and prepares to write back the required TCD*n* fields into the local memory. The TCD*n* word 7 is read and checked for channel linking or scatter/gather requests. |
| 14 | 15 | The appropriate fields in the first part of the TCD*n* are written back into the local memory. |
| 15 | 16 | The fields in the second part of the TCD*n* are written back into the local memory. This cycle coincides with the next channel arbitration cycle start. |
| 16 | 17 | The next channel to be activated performs the read of the first part of its TCD from the local memory. This is equivalent to Cycle 4 for the first channel's service request. |

Assuming zero wait states on the system bus, DMA requests can be processed every 9 cycles. Assuming an average of the access times associated with internal peripheral bus-to-SRAM (4 cycles) and SRAM-to-internal peripheral bus (5 cycles), DMA requests can be processed every 11.5 cycles (4 + (4+5)/2 + 3). This is the time from Cycle 4 to Cycle x +5. The resulting peak request rate, as a function of the system frequency, is shown in the following table.

**Table 22-294. eDMA peak request rate (MReq/sec)**

| System frequency (MHz) | Request rate with zero wait states | Request rate with wait states |
|---|---|---|
| 66.6 | 7.4 | 5.8 |
| 83.3 | 9.2 | 7.2 |
| 100.0 | 11.1 | 8.7 |
| 133.3 | 14.8 | 11.6 |
| 150.0 | 16.6 | 13.0 |

A general formula to compute the peak request rate with overlapping requests is:

PEAKreq = freq / [ entry + (1 + read_ws) + (1 + write_ws) + exit ]

where:

**Table 22-295. Peak request formula operands**

| Operand | Description |
|---|---|
| PEAKreq | Peak request rate |
| freq | System frequency |
| entry | Channel startup (4 cycles) |
| read_ws | Wait states seen during the system bus read data phase |
| write_ws | Wait states seen during the system bus write data phase |
| exit | Channel shutdown (3 cycles) |

### 22.4.4.3 eDMA performance example

Consider a system with the following characteristics:

- Internal SRAM can be accessed with one wait-state when viewed from the system bus data phase
- All internal peripheral bus reads require two wait-states, and internal peripheral bus writes three wait-states viewed from the system bus data phase
- System operates at 150 MHz

For an SRAM to internal peripheral bus transfer,

PEAKreq = 150 MHz / [ 4 + (1 + 1) + (1 + 3) + 3 ] cycles = 11.5 Mreq/sec

For an internal peripheral bus to SRAM transfer,

PEAKreq = 150 MHz / [ 4 + (1 + 2) + (1 + 1) + 3 ] cycles = 12.5 Mreq/sec

Assuming an even distribution of the two transfer types, the average peak request rate would be:

PEAKreq = (11.5 Mreq/sec + 12.5 Mreq/sec) / 2 = 12.0 Mreq/sec

The minimum number of cycles to perform a single read/write, zero wait states on the system bus, from a cold start where no channel is executing and eDMA is idle are:

- 11 cycles for a software, that is, a TCD*n*_CSR[START] bit, request
- 12 cycles for a hardware, that is, an eDMA peripheral request signal, request

Two cycles account for the arbitration pipeline and one extra cycle on the hardware request resulting from the internal registering of the eDMA peripheral request signals. For the peak request rate calculations above, the arbitration and request registering is absorbed in or overlaps the previous executing channel.

**Note**

When channel linking or scatter/gather is enabled, a two cycle delay is imposed on the next channel selection and startup. This allows the link channel or the scatter/gather channel to be eligible and considered in the arbitration pool for next channel selection.

## 22.5 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

### 22.5.1 eDMA initialization

To initialize the eDMA:

1. Write to the CR if a configuration other than the default is desired.
2. Write the channel priority levels to the DCHPRI*n* registers if a configuration other than the default is desired.
3. Enable error interrupts in the EEI register if so desired.
4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the ERQ register.
6. Request channel service via either:
   - Software: setting the TCD*n*_CSR[START]
   - Hardware: slave device asserting its eDMA peripheral request signal

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The eDMA engine reads the entire TCD, including the TCD control and status fields, as shown in the following table, for the selected channel into its internal address path module.

As the TCD is read, the first transfer is initiated on the internal bus, unless a configuration error is detected. Transfers from the source, as defined by TCD*n*_SADDR, to the destination, as defined by TCD*n*_DADDR, continue until the number of bytes specified by TCD*n*_NBYTES are transferred.

When the transfer is complete, the eDMA engine's local TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing executes, such as interrupts, major loop channel linking, and scatter/gather operations, if enabled.

**Table 22-296. TCD Control and Status fields**

| TCD*n*_CSR field name | Description |
|---|---|
| START | Control bit to start channel explicitly when using a software initiated DMA service (Automatically cleared by hardware) |
| ACTIVE | Status bit indicating the channel is currently in execution |
| DONE | Status bit indicating major loop completion (cleared by software when using a software initiated DMA service) |
| D_REQ | Control bit to disable DMA request at end of major loop completion when using a hardware initiated DMA service |
| BWC | Control bits for throttling bandwidth control of a channel |
| E_SG | Control bit to enable scatter-gather feature |
| INT_HALF | Control bit to enable interrupt when major loop is half complete |
| INT_MAJ | Control bit to enable interrupt when major loop completes |

The following figure shows how each DMA request initiates one minor-loop transfer, or iteration, without CPU intervention. DMA arbitration can occur after each minor loop, and one level of minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (BITER).

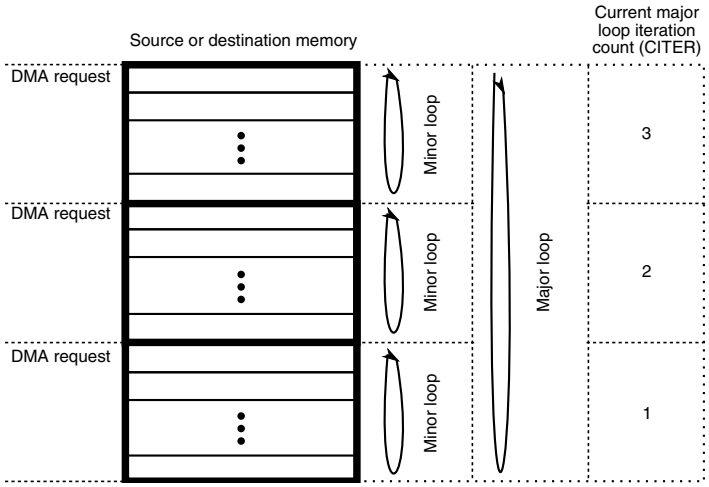



**Figure 22-292. Example of multiple loop iterations**

The following figure lists the memory array terms and how the TCD settings interrelate.

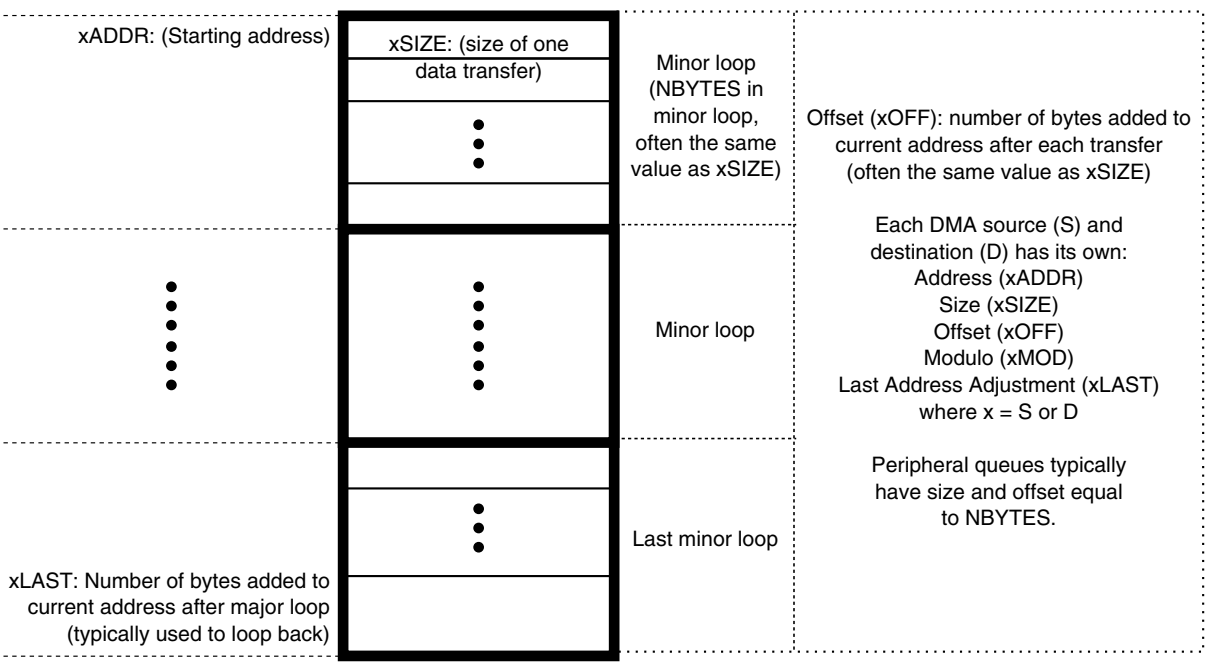



**Figure 22-293. Memory array terms**

## 22.5.2 Programming errors

The eDMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per channel basis with the exception of channel priority error (ES[CPE]).

For all error types other than channel priority error, the channel number causing the error is recorded in the Error Status register (DMAx_ES). If the error source is not removed before the next activation of the problem channel, the error is detected and recorded again.

If priority levels are not unique, when any channel requests service, a channel priority error is reported. The highest channel priority with an active request is selected, but the lowest numbered channel with that priority is selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts, and error reporting is associated with the selected channel.

## 22.5.3 Arbitration mode considerations

### 22.5.3.1 Fixed channel arbitration

In this mode, the channel service request from the highest priority channel is selected to execute.

### 22.5.3.2 Round-robin channel arbitration

Channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels.

## 22.5.4 Performing DMA transfers (examples)

### 22.5.4.1 Single request

To perform a simple transfer of n bytes of data with one activation, set the major loop to one (TCD*n*_CITER = TCD*n*_BITER = 1). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, the TCD*n*_CSR[DONE] bit is set and an interrupt generates if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a 32-bit port located at 0x2000. The address offsets are

programmed in increments to match the transfer size: one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```
TCDn_CITER = TCDn_BITER = 1
TCDn_NBYTES = 16
TCDn_SADDR = 0x1000
TCDn_SOFF = 1
TCDn_ATTR[SSIZE] = 0
TCDn_SLAST = -16
TCDn_DADDR = 0x2000
TCDn_DOFF = 4
TCDn_ATTR[DSIZE] = 2
TCDn_DLAST_SGA= -16
TCDn_CSR[INT_MAJ] = 1
TCDn_CSR[START] = 1 (Should be written last after all other fields have been initialized)
All other TCDn fields = 0
```

This generates the following event sequence:

1. User write to the TCD*n*_CSR[START] bit requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD data from local memory to internal register file.
5. The source-to-destination transfers are executed as follows:
   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b. Write 32-bits to location 0x2000 → first iteration of the minor loop.
   c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d. Write 32-bits to location 0x2004 → second iteration of the minor loop.
   e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
   f. Write 32-bits to location 0x2008 → third iteration of the minor loop.
   g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
   h. Write 32-bits to location 0x200C → last iteration of the minor loop → major loop complete.

6. The eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 1 (TCD*n*_BITER).
7. The eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[*n*] = 1.
8. The channel retires and the eDMA goes idle or services the next channel.

### 22.5.4.2 Multiple requests

The following example transfers 32 bytes via two hardware requests, but is otherwise the same as the previous example. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop transferring 16 bytes per iteration. After the channel's hardware requests are enabled in the ERQ register, the slave device initiates channel service requests.

```
TCDn_CITER = TCDn_BITER = 2
TCDn_SLAST = -32
TCDn_DLAST_SGA = -32
```

This would generate the following sequence of events:

1. First hardware, that is, eDMA peripheral, request for channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD*n* data from local memory to internal register file.
5. The source to destination transfers are executed as follows:
   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b. Write 32-bits to location 0x2000 → first iteration of the minor loop.
   c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d. Write 32-bits to location 0x2004 → second iteration of the minor loop.
   e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
   f. Write 32-bits to location 0x2008 → third iteration of the minor loop.

g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.

h. Write 32-bits to location 0x200C → last iteration of the minor loop.

6. eDMA engine writes: TCD*n*_SADDR = 0x1010, TCD*n*_DADDR = 0x2010, TCD*n*_CITER = 1.

7. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0.

8. The channel retires → one iteration of the major loop. The eDMA goes idle or services the next channel.

9. Second hardware, that is, eDMA peripheral, requests channel service.

10. The channel is selected by arbitration for servicing.

11. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.

12. eDMA engine reads: channel TCD data from local memory to internal register file.

13. The source to destination transfers are executed as follows:

a. Read byte from location 0x1010, read byte from location 0x1011, read byte from 0x1012, read byte from 0x1013.

b. Write 32-bits to location 0x2010 → first iteration of the minor loop.

c. Read byte from location 0x1014, read byte from location 0x1015, read byte from 0x1016, read byte from 0x1017.

d. Write 32-bits to location 0x2014 → second iteration of the minor loop.

e. Read byte from location 0x1018, read byte from location 0x1019, read byte from 0x101A, read byte from 0x101B.

f. Write 32-bits to location 0x2018 → third iteration of the minor loop.

g. Read byte from location 0x101C, read byte from location 0x101D, read byte from 0x101E, read byte from 0x101F.

h. Write 32-bits to location 0x201C → last iteration of the minor loop → major loop complete.

14. eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 2 (TCD*n*_BITER).

15. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[n] = 1.

16. The channel retires → major loop complete. The eDMA goes idle or services the next channel.

### 22.5.4.3 Using the modulo feature

The modulo feature of the eDMA provides the ability to implement a circular data queue in which the size of the queue is a power of 2. MOD is a 5-bit field for the source and destination in the TCD, and it specifies which lower address bits increment from their original value after the address+offset calculation. All upper address bits remain the same as in the original value. A setting of 0 for this field disables the modulo feature.

The following table shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x1234567*x*) retain their original value. In this example the source address is set to 0x12345670, the offset is set to 4 bytes and the MOD field is set to 4, allowing for a $2^4$ byte (16-byte) size queue.

**Table 22-297. Modulo example**

| Transfer Number | Address |
|---|---|
| 1 | 0x12345670 |
| 2 | 0x12345674 |
| 3 | 0x12345678 |
| 4 | 0x1234567C |
| 5 | 0x12345670 |
| 6 | 0x12345674 |

## 22.5.5 Monitoring transfer descriptor status

### 22.5.5.1 Testing for minor loop completion

There are two methods to test for minor loop completion when using software initiated service requests. The first is to read the TCD*n*_CITER field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test the TCD*n*_CSR[START] bit and the TCD*n*_CSR[ACTIVE] bit. The minor-loop-complete condition is indicated by both bits reading zero after the TCD*n*_CSR[START] was set. Polling the TCD*n*_CSR[ACTIVE] bit may be inconclusive, because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute the following sequence for a software activated channel:

| Stage | TCD*n*_CSR bits | | | State |
|---|---|---|---|---|
| | START | ACTIVE | DONE | |
| 1 | 1 | 0 | 0 | Channel service request via software |
| 2 | 0 | 1 | 0 | Channel is executing |
| 3a | 0 | 0 | 0 | Channel has completed the minor loop and is idle |
| 3b | 0 | 0 | 1 | Channel has completed the major loop and is idle |

The best method to test for minor-loop completion when using hardware, that is, peripheral, initiated service requests is to read the TCD*n*_CITER field and test for a change. The hardware request and acknowledge handshake signals are not visible in the programmer's model.

The TCD status bits execute the following sequence for a hardware-activated channel:

| Stage | TCD*n*_CSR bits | | | State |
|---|---|---|---|---|
| | START | ACTIVE | DONE | |
| 1 | 0 | 0 | 0 | Channel service request via hardware (peripheral request asserted) |
| 2 | 0 | 1 | 0 | Channel is executing |
| 3a | 0 | 0 | 0 | Channel has completed the minor loop and is idle |
| 3b | 0 | 0 | 1 | Channel has completed the major loop and is idle |

For both activation types, the major-loop-complete status is explicitly indicated via the TCD*n*_CSR[DONE] bit.

The TCD*n*_CSR[START] bit is cleared automatically when the channel begins execution regardless of how the channel activates.

### 22.5.5.2 Reading the transfer descriptors of active channels

The eDMA reads back the true TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_NBYTES values if read while a channel executes. The true values of the SADDR, DADDR, and NBYTES are the values the eDMA engine currently uses in its internal register file and not the values in the TCD local memory for that channel. The addresses, SADDR and DADDR, and NBYTES, which decrement to zero as the transfer progresses, can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

### 22.5.5.3 Checking channel preemption status

Preemption is available only when fixed arbitration is selected as the channel arbitration mode. A preemptive situation is one in which a preempt-enabled channel runs and a higher priority request becomes active. When the eDMA engine is not operating in fixed channel arbitration mode, the determination of the actively running relative priority outstanding requests become undefined. Channel priorities are treated as equal, that is, constantly rotating, when Round-Robin Arbitration mode is selected.

The TCD*n*_CSR[ACTIVE] bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one major loop iteration. If two TCD*n*_CSR[ACTIVE] bits are set simultaneously in the global TCD map, a higher priority channel is actively preempting a lower priority channel.

## 22.5.6 Channel Linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD*n*_CSR[START] bit of another channel (or itself), therefore initiating a service request for that channel. When properly enabled, the EDMA engine automatically performs this operation at the major or minor loop completion.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD*n*_CITER[E_LINK] field determines whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, the initial fields of:

```
TCDn_CITER[E_LINK] = 1
TCDn_CITER[LINKCH] = 0xC
TCDn_CITER[CITER] value = 0x4
TCDn_CSR[MAJOR_E_LINK] = 1
TCDn_CSR[MAJOR_LINKCH] = 0x7
```

executes as:

1. Minor loop done → set TCD12_CSR[START] bit
2. Minor loop done → set TCD12_CSR[START] bit
3. Minor loop done → set TCD12_CSR[START] bit
4. Minor loop done, major loop done→ set TCD7_CSR[START] bit

When minor loop linking is enabled (TCD*n*_CITER[E_LINK] = 1), the TCD*n*_CITER[CITER] field uses a nine bit vector to form the current iteration count. When minor loop linking is disabled (TCD*n*_CITER[E_LINK] = 0), the TCD*n*_CITER[CITER] field uses a 15-bit vector to form the current iteration count. The bits associated with the TCD*n*_CITER[LINKCH] field are concatenated onto the CITER value to increase the range of the CITER.

**Note**

The TCD*n*_CITER[E_LINK] bit and the TCD*n*_BITER[E_LINK] bit must equal or a configuration error is reported. The CITER and BITER vector widths must be equal to calculate the major loop, half-way done interrupt point.

The following table summarizes how a DMA channel can link to another DMA channel, i.e, use another channel's TCD, at the end of a loop.

**Table 22-298. Channel Linking Parameters**

| Desired Link Behavior | TCD Control Field Name | Description |
|---|---|---|
| Link at end of Minor Loop | CITER[E_LINK] | Enable channel-to-channel linking on minor loop completion (current iteration) |
| | CITER[LINKCH] | Link channel number when linking at end of minor loop (current iteration) |
| Link at end of Major Loop | CSR[MAJOR_E_LINK] | Enable channel-to-channel linking on major loop completion |
| | CSR[MAJOR_LINKCH] | Link channel number when linking at end of major loop |

## 22.5.7 Dynamic programming

### 22.5.7.1 Dynamically changing the channel priority

The following two options are recommended for dynamically changing channel priority levels:

1. Switch to Round-Robin Channel Arbitration mode, change the channel priorities, then switch back to Fixed Arbitration mode,
2. Disable all the channels, change the channel priorities, then enable the appropriate channels.

### 22.5.7.2 Dynamic channel linking

Dynamic channel linking is the process of setting the TCD.major.e_link bit during channel execution. This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic channel link by enabling the TCD.major.e_link bit at the same time the eDMA engine is retiring the channel. The TCD.major.e_link would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

The following coherency model is recommended when executing a dynamic channel link request.

1. Write 1 to the TCD.major.e_link bit.
2. Read back the TCD.major.e_link bit.
3. Test the TCD.major.e_link request status:
   - If TCD.major.e_link = 1, the dynamic link attempt was successful.
   - If TCD.major.e_link = 0, the attempted dynamic link did not succeed (the channel was already retiring).

For this request, the TCD local memory controller forces the TCD.major.e_link bit to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set, indicating the major loop is complete.

**NOTE**

The user must clear the TCD.done bit before writing the TCD.major.e_link bit. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

### 22.5.7.3 Dynamic scatter/gather

Scatter/gather is the process of automatically loading a new TCD into a channel. It allows a DMA channel to use multiple TCDs; this enables a DMA channel to scatter the DMA data to multiple destinations or gather it from multiple sources.When scatter/gather is enabled and the channel has finished its major loop, a new TCD is fetched from system memory and loaded into that channel's descriptor location in eDMA programmer's model, thus replacing the current descriptor.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic scatter/gather operation by enabling the TCD.e_sg bit at the same time the eDMA engine

is retiring the channel. The TCD.e_sg would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods for this coherency model are shown in the following subsections. Method 1 has the advantage of reading the major.linkch field and the e_sg bit with a single read. For both dynamic channel linking and scatter/gather requests, the TCD local memory controller forces the TCD.major.e_link and TCD.e_sg bits to zero on any writes to a channel's TCD.word7 if that channel's TCD.done bit is set indicating the major loop is complete.

**NOTE**

The user must clear the TCD.done bit before writing the TCD.major.e_link or TCD.e_sg bits. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

#### 22.5.7.3.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request.

When the TCD.major.e_link bit is zero, the TCD.major.linkch field is not used by the eDMA. In this case, the TCD.major.linkch bits may be used for other purposes. This method uses the TCD.major.linkch field as a TCD indentification (ID).

1. When the descriptors are built, write a unique TCD ID in the TCD.major.linkch field for each TCD associated with a channel using dynamic scatter/gather.
2. Write 1b to the TCD.d_req bit.

   Should a dynamic scatter/gather attempt fail, setting the TCD.d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offest value.

3. Write the TCD.dlast_sga field with the scatter/gather address.
4. Write 1b to the TCD.e_sg bit.
5. Read back the 16 bit TCD control/status field.
6. Test the TCD.e_sg request status and TCD.major.linkch value:

   If e_sg = 1b, the dynamic link attempt was successful.

If e_sg = 0b and the major.linkch (ID) did not change, the attempted dynamic link did not succeed (the channel was already retiring).

If e_sg = 0b and the major.linkch (ID) changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

#### 22.5.7.3.2 Method 2 (channel using major loop channel linking)

For a channel using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request. This method uses the TCD.dlast_sga field as a TCD indentification (ID).

1. Write 1b to the TCD.d_req bit.

   Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offest value.

2. Write theTCD.dlast_sga field with the scatter/gather address.
3. Write 1b to the TCD.e_sg bit.
4. Read back the TCD.e_sg bit.
5. Test the TCD.e_sg request status:

   If e_sg = 1b, the dynamic link attempt was successful.

   If e_sg = 0b, read the 32 bit TCD dlast_sga field.

   If e_sg = 0b and the dlast_sga did not change, the attempted dynamic link did not succeed (the channel was already retiring).

   If e_sg = 0b and the dlast_sga changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

# Chapter 23
# External Watchdog Monitor (EWM)

## 23.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The watchdog is generally used to monitor the flow and execution of embedded software within an MCU. The watchdog consists of a counter that if allowed to overflow, forces an internal reset (asynchronous) to all on-chip peripherals and optionally assert the RESET pin to reset external devices/circuits. The overflow of the watchdog counter must not occur if the software code works well and services the watchdog to re-start the actual counter.

For safety, a redundant watchdog system, External Watchdog Monitor (EWM), is designed to monitor external circuits, as well as the MCU software flow. This provides a back-up mechanism to the internal watchdog that resets the MCU's CPU and peripherals.

The EWM differs from the internal watchdog in that it does not reset the MCU's CPU and peripherals. The EWM if allowed to time-out, provides an independent EWM_out pin that when asserted resets or places an external circuit into a safe mode. The CPU resets the EWM counter that is logically ANDed with an external digital input pin. This pin allows an external circuit to influence the reset_out signal.

### 23.1.1 Features

Features of EWM module include:

- Independent LPO clock source
- Programmable time-out period specified in terms of number of EWM LPO clock cycles.

- Windowed refresh option
  - Provides robust check that program flow is faster than expected.
  - Programmable window.
  - Refresh outside window leads to assertion of $\overline{\text{EWM\_out}}$.
- Robust refresh mechanism
  - Write values of 0xB4 and 0x2C to EWM Refresh Register within 15 (*EWM_service_time*) peripheral bus clock cycles.
- One output port, $\overline{\text{EWM\_out}}$, when asserted is used to reset or place the external circuit into safe mode.
- One Input port, EWM_in, allows an external circuit to control the $\overline{\text{EWM\_out}}$ signal.

## 23.1.2 Modes of Operation

This section describes the module's operating modes.

### 23.1.2.1 Stop Mode

When the EWM is in stop mode, the CPU services to the EWM cannot occur. On entry to stop mode, the EWM's counter freezes.

There are two possible ways to exit from Stop mode:

- On exit from stop mode through a reset, the EWM remains disabled.
- On exit from stop mode by an interrupt, the EWM is re-enabled, and the counter continues to be clocked from the same value prior to entry to stop mode.

Note the following if the EWM enters the stop mode during CPU service mechanism: At the exit from stop mode by an interrupt, refresh mechanism state machine starts from the previous state which means, if first service command is written correctly and EWM enters the stop mode immediately, the next command has to be written within the next 15 (*EWM_service_time*) peripheral bus clocks after exiting from stop mode. User must mask all interrupts prior to executing EWM service instructions.

### 23.1.2.2 Wait Mode

The EWM module treats the stop and wait modes as the same. EWM functionality remains the same in both of these modes.

### 23.1.2.3 Debug Mode

Entry to debug mode has no effect on the EWM.

- If the EWM is enabled prior to entry of debug mode, it remains enabled.
- If the EWM is disabled prior to entry of debug mode, it remains disabled.

## 23.1.3 Block Diagram

This figure shows the EWM block diagram.

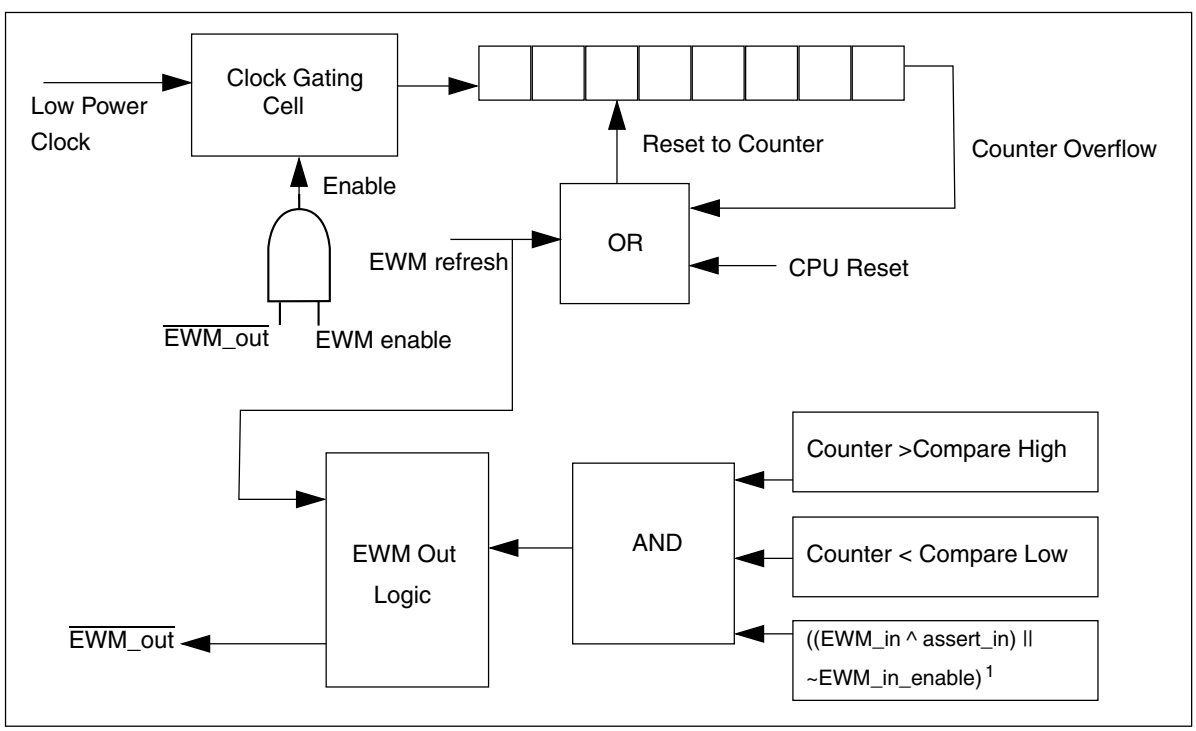



**Figure 23-1. EWM Block Diagram**

## 23.2 EWM Signal Descriptions

The EWM has two external signals, as shown in the following table.

**Table 23-1. EWM Signal Descriptions**

| Signal | Description | I/O |
|---|---|---|
| EWM_in | EWM input for safety status of external safety circuits. The polarity of EWM_in is programmable using the EWM_CTRL[ASSIN] bit. The default polarity is active-low. | I |
| $\overline{\text{EWM\_out}}$ | EWM reset out signal | O |

## 23.3 Memory Map/Register Definition

This section contains the module memory map and registers.

**EWM memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4006_1000 | Control Register (EWM_CTRL) | 8 | R/W | 00h | 23.3.1/506 |
| 4006_1001 | Service Register (EWM_SERV) | 8 | W (always reads 0) | 00h | 23.3.2/507 |
| 4006_1002 | Compare Low Register (EWM_CMPL) | 8 | R/W | 00h | 23.3.3/507 |
| 4006_1003 | Compare High Register (EWM_CMPH) | 8 | R/W | FFh | 23.3.4/508 |

### 23.3.1 Control Register (EWM_CTRL)

The CTRL register is cleared by any reset.

**NOTE**

INEN, ASSIN and EWMEN bits can be written once after a CPU reset. Modifying these bits more than once, generates a bus transfer error.

Address: 4006_1000h base + 0h offset = 4006_1000h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | | | | INTEN | INEN | ASSIN | EWMEN |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EWM_CTRL field descriptions**

| Field | Description |
|---|---|
| 7–4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3<br>INTEN | Interrupt Enable.<br><br>This bit when set and $\overline{\text{EWM\_out}}$ is asserted, an interrupt request is generated. To de-assert interrupt request, user should clear this bit by writing 0. |
| 2<br>INEN | Input Enable.<br><br>This bit when set, enables the EWM_in port. |
| 1<br>ASSIN | EWM_in's Assertion State Select.<br><br>Default assert state of the EWM_in signal is logic zero. Setting ASSIN bit inverts the assert state to a logic one. |
| 0<br>EWMEN | EWM enable.<br><br>This bit when set, enables the EWM module. This resets the EWM counter to zero and deasserts the $\overline{\text{EWM\_out}}$ signal. Clearing EWMEN bit disables the EWM, and therefore it cannot be enabled until a reset occurs, due to the write-once nature of this bit. |

## 23.3.2 Service Register (EWM_SERV)

The SERV register provides the interface from the CPU to the EWM module. It is write-only and reads of this register return zero.

Address: 4006_1000h base + 1h offset = 4006_1001h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | | | | | | | |
| Write | SERVICE | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EWM_SERV field descriptions**

| Field | Description |
|---|---|
| 7–0<br>SERVICE | The EWM service mechanism requires the CPU to write two values to the SERV register: a first data byte of 0xB4, followed by a second data byte of 0x2C. The EWM service is illegal if either of the following conditions is true.<br>• The first or second data byte is not written correctly.<br>• The second data byte is not written within a fixed number of peripheral bus cycles of the first data byte. This fixed number of cycles is called *EWM_service_time*. |

## 23.3.3 Compare Low Register (EWM_CMPL)

The CMPL register is reset to zero after a CPU reset. This provides no minimum time for the CPU to service the EWM counter.

**NOTE**

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

Address: 4006_1000h base + 2h offset = 4006_1002h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | COMPAREL | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EWM_CMPL field descriptions**

| Field | Description |
|---|---|
| 7–0<br>COMPAREL | To prevent runaway code from changing this field, software should write to this field after a CPU reset even if the (default) minimum service time is required. |

## 23.3.4 Compare High Register (EWM_CMPH)

The CMPH register is reset to 0xFF after a CPU reset. This provides a maximum of 256 clocks time, for the CPU to service the EWM counter.

**NOTE**

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

**NOTE**

The valid values for CMPH are up to 0xFE because the EWM counter never expires when CMPH = 0xFF. The expiration happens only if EWM counter is greater than CMPH.

Address: 4006_1000h base + 3h offset = 4006_1003h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | COMPAREH | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**EWM_CMPH field descriptions**

| Field | Description |
|---|---|
| 7–0<br>COMPAREH | To prevent runaway code from changing this field, software should write to this field after a CPU reset even if the (default) maximum service time is required. |

## 23.4 Functional Description

The following sections describe functional details of the EWM module.

### 23.4.1 The EWM_out Signal

The EWM_out is a digital output signal used to gate an external circuit (application specific) that controls critical safety functions. For example, the EWM_out could be connected to the high voltage transistors circuits that control an AC motor in a large appliance.

The EWM_out signal remains deasserted when the EWM is being regularly serviced by the CPU within the programmable service window, indicating that the application code is executed as expected.

The EWM_out signal is asserted in any of the following conditions:

- Servicing the EWM when the counter value is less than CMPL value.
- If the EWM counter value reaches the CMPH value, and no EWM service has occurred.
- Servicing the EWM when the counter value is more than CMPL and less than CMPH values and EWM_in signal is asserted.
- If functionality of EWM_in pin is enabled and EWM_in pin is asserted while servicing the EWM.
- After any reset (by the virtue of the external pull-down mechanism on the EWM_out pin)

On a normal reset, the EWM_out is asserted. To deassert the EWM_out, set EWMEN bit in the CTRL register to enable the EWM.

If the EWM_out signal shares its pad with a digital I/O pin, on reset this actual pad defers to being an input signal. It takes the EWM_out output condition only after you enable the EWM by the EWMEN bit in the CTRL register.

When the EWM_out pin is asserted, it can only be deasserted by forcing a MCU reset.

**Note**

EWM_out pad must be in pull down state when EWM functionality is used and when EWM is under Reset.

### 23.4.2 The EWM_in Signal

The EWM_in is a digital input signal that allows an external circuit to control the $\overline{\text{EWM\_out}}$ signal. For example, in the application, an external circuit monitors a critical safety function, and if there is fault with this circuit's behavior, it can then actively initiate the $\overline{\text{EWM\_out}}$ signal that controls the gating circuit.

The EWM_in signal is ignored if the EWM is disabled, or if INEN bit of CTRL register is cleared, as after any reset.

On enabling the EWM (setting the CTRL[EWMEN] bit) and enabling EWM_in functionality (setting the CTRL[INEN] bit), the EWM_in signal must be in the deasserted state prior to the CPU servicing the EWM. This ensures that the $\overline{\text{EWM\_out}}$ stays in the deasserted state; otherwise, the $\overline{\text{EWM\_out}}$ pin is asserted.

**Note**

You must update the CMPH and CMPL registers prior to enabling the EWM. After enabling the EWM, the counter resets to zero, therefore providing a reasonable time after a power-on reset for the external monitoring circuit to stabilize and ensure that the EWM_in pin is deasserted.

### 23.4.3 EWM Counter

It is an 8-bit ripple counter fed from a clock source that is independent of the peripheral bus clock source. As the preferred time-out is between 1 ms and 100 ms the actual clock source should be in the kHz range.

The counter is reset to zero, after a CPU reset, or a EWM refresh cycle. The counter value is not accessible to the CPU.

### 23.4.4 EWM Compare Registers

The compare registers CMPL and CMPH are write-once after a CPU reset and cannot be modified until another CPU reset occurs.

The EWM compare registers are used to create a service window, which is used by the CPU to service/refresh the EWM module.

- If the CPU services the EWM when the counter value lies between CMPL value and CMPH value, the counter is reset to zero. This is a legal service operation.
- If the CPU executes a EWM service/refresh action outside the legal service window, EWM_out is asserted.

It is illegal to program CMPL and CMPH with same value. In this case, as soon as counter reaches (CMPL + 1), EWM_out is asserted.

## 23.4.5 EWM Refresh Mechanism

Other than the initial configuration of the EWM, the CPU can only access the EWM by the EWM Service Register. The CPU must access the EWM service register with correct write of unique data within the windowed time frame as determined by the CMPL and CMPH registers. Therefore, three possible conditions can occur:

**Table 23-7. EWM Refresh Mechanisms**

| Condition | Mechanism |
|---|---|
| A unique EWM service occurs when CMPL < Counter < CMPH. | The software behaves as expected and the counter of the EWM is reset to zero, and EWM_out pin remains in the deasserted state.<br>**Note:** EWM_in pin is also assumed to be in the deasserted state. |
| A unique EWM service occurs when Counter < CMPL | The software services the EWM and therefore resets the counter to zero and asserts the EWM_out pin (irrespective of the EWM_in pin). The EWM_out pin is expected to gate critical safety circuits. |
| Counter value reaches CMPH prior to a unique EWM service | The counter value reaches the CMPH value and no service of the EWM resets the counter to zero and assert the EWM_out pin (irrespective of the EWM_in pin). The EWM_out pin is expected to gate critical safety circuits. |

Any illegal service on EWM has no effect on EWM_out.

## 23.4.6 EWM Interrupt

When EWM_out is asserted, an interrupt request is generated to indicate the assertion of the EWM reset out signal. This interrupt is enabled when CTRL[INTEN] is set. Clearing this bit clears the interrupt request but does not affect EWM_out. The EWM_out signal can be deasserted only by forcing a system reset.

# Chapter 24
# Watchdog Timer (WDOG)

## 24.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The Watchdog Timer (WDOG) keeps a watch on the system functioning and resets it in case of its failure. Reasons for failure include run-away software code and the stoppage of the system clock that in a safety critical system can lead to serious consequences. In such cases, the watchdog brings the system into a safe state of operation. The watchdog monitors the operation of the system by expecting periodic communication from the software, generally known as servicing or refreshing the watchdog. If this periodic refreshing does not occur, the watchdog resets the system.

## 24.2 Features

The features of the Watchdog Timer (WDOG) include:

- Clock source input independent from CPU/bus clock. Choice between two clock sources:
  - Low-power oscillator (LPO)
  - External system clock
- Unlock sequence for allowing updates to write-once WDOG control/configuration bits.
- All WDOG control/configuration bits are writable once only within 256 bus clock cycles of being unlocked.

    - You need to always update these bits after unlocking within 256 bus clock cycles. Failure to update these bits resets the system.
- Programmable time-out period specified in terms of number of WDOG clock cycles.
- Ability to test WDOG timer and reset with a flag indicating watchdog test.
    - Quick test—Small time-out value programmed for quick test.
    - Byte test—Individual bytes of timer tested one at a time.
    - Read-only access to the WDOG timer—Allows dynamic check that WDOG timer is operational.

**NOTE**

Reading the watchdog timer counter while running the watchdog on the bus clock might not give the accurate counter value.

- Windowed refresh option
    - Provides robust check that program flow is faster than expected.
    - Programmable window.
    - Refresh outside window leads to reset.
- Robust refresh mechanism
    - Write values of 0xA602 and 0xB480 to WDOG Refresh Register within 20 bus clock cycles.
- Count of WDOG resets as they occur.
- Configurable interrupt on time-out to provide debug breadcrumbs. This is followed by a reset after 256 bus clock cycles.

## 24.3 Functional overview

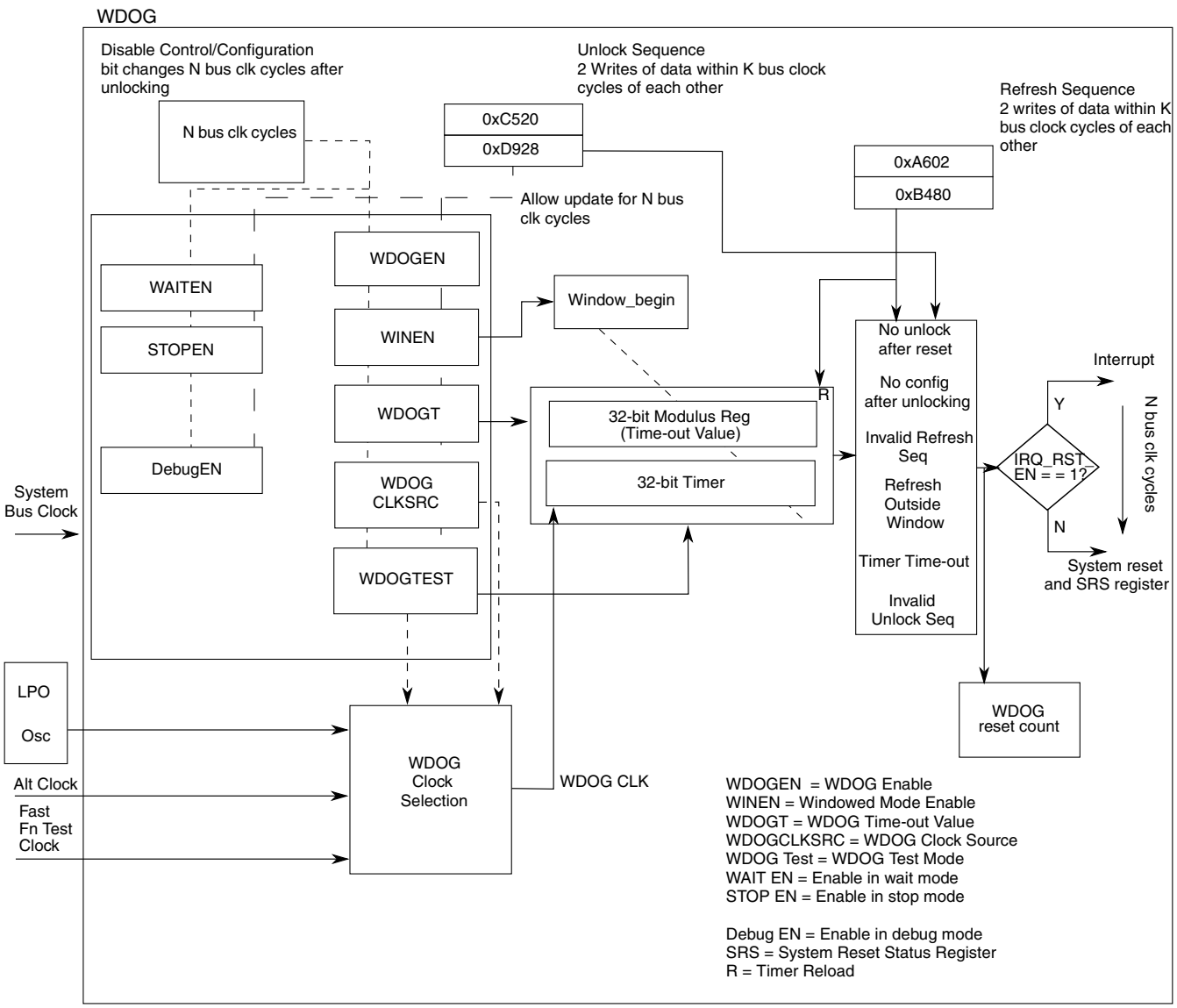



**Figure 24-1. WDOG operation**

The preceding figure shows the operation of the watchdog. The values for N and K are:

- N = 256
- K = 20

The watchdog is a fail safe mechanism that brings the system into a known initial state in case of its failure due to CPU clock stopping or a run-away condition in code execution. In its simplest form, the watchdog timer runs continuously off a clock source and expects

to be serviced periodically, failing which it resets the system. This ensures that the software is executing correctly and has not run away in an unintended direction. Software can adjust the period of servicing or the time-out value for the watchdog timer to meet the needs of the application.

You can select a windowed mode of operation that expects the servicing to be done only in a particular window of the time-out period. An attempted servicing of the watchdog outside this window results in a reset. By operating in this mode, you can get an indication of whether the code is running faster than expected. The window length is also user programmable.

If a system fails to update/refresh the watchdog due to an unknown and persistent cause, it will be caught in an endless cycle of resets from the watchdog. To analyze the cause of such conditions, you can program the watchdog to first issue an interrupt, followed by a reset. In the interrupt service routine, the software can analyze the system stack to aid debugging.

To enhance the independence of watchdog from the system, it runs off an independent LPO oscillator clock. You can also switch over to an alternate clock source if required, through a control register bit.

### 24.3.1 Unlocking and updating the watchdog

As long as ALLOW_UPDATE in the watchdog control register is set, you can unlock and modify the write-once-only control and configuration registers:

1. Write 0xC520 followed by 0xD928 within 20 bus clock cycles to a specific unlock register (WDOG_UNLOCK).
2. Wait one bus clock cycle. You cannot update registers on the bus clock cycle immediately following the write of the unlock sequence.
3. An update window equal in length to the watchdog configuration time (WCT) opens. Within this window, you can update the configuration and control register bits.

These register bits can be modified only once after unlocking.

If none of the configuration and control registers is updated within the update window, the watchdog issues a reset, that is, interrupt-then-reset, to the system. Trying to unlock the watchdog within the WCT after an initial unlock has no effect. During the update operation, the watchdog timer is not paused and continues running in the background. After the update window closes, the watchdog timer restarts and the watchdog functions according to the new configuration.

The update feature is useful for applications that have an initial, non-safety critical part, where the watchdog is kept disabled or with a conveniently long time-out period. This means the application coder does not have to frequently service the watchdog. After the critical part of the application begins, the watchdog can be reconfigured as needed.

The watchdog issues a reset, that is, interrupt-then-reset if enabled, to the system for any of these invalid unlock sequences:

- Write any value other than 0xC520 or 0xD928 to the unlock register.
- ALLOW_UPDATE is set and a gap of more than 20 bus clock cycles is inserted between the writing of the unlock sequence values.

An attempted refresh operation between the two writes of the unlock sequence and in the WCT time following a successful unlock, goes undetected. Also, see Watchdog Operation with 8-bit access for guidelines related to 8-bit accesses to the unlock register.

**Note**

A context switch during unlocking and refreshing may lead to a watchdog reset.

### 24.3.2 Watchdog configuration time (WCT)

To prevent unintended modification of the watchdog's control and configuration register bits, you are allowed to update them only within a period of 256 bus clock cycles after unlocking. This period is known as the watchdog configuration time (WCT). In addition, these register bits can be modified only once after unlocking them for editing, even after reset.

You must unlock the registers within WCT after system reset, failing which the WDOG issues a reset to the system. In other words, you must write at least the first word of the unlocking sequence within the WCT after reset. After this is done, you have a further 20 bus clock cycles, the maximum allowed gap between the words of the unlock sequence, to complete the unlocking operation. Thereafter, to make sure that you do not forget to configure the watchdog, the watchdog issues a reset if none of the WDOG control and configuration registers is updated in the WCT after unlock. After the close of this window or after the first write, these register bits are locked out from any further changes.

The watchdog timer keeps running according to its default configuration through unlocking and update operations that can extend up to a maximum total of 2xWCT + 20 bus clock cycles. Therefore, it must be ensured that the time-out value for the watchdog is always greater than 2xWCT time + 20 bus clock cycles.

Updates in the write-once registers take effect only after the WCT window closes with the following exceptions for which changes take effect immediately:

- Stop, Wait, and Debug mode enable
- IRQ_RST_EN

The operations of refreshing the watchdog goes undetected during the WCT.

### 24.3.3 Refreshing the watchdog

A robust refreshing mechanism has been chosen for the watchdog. A valid refresh is a write of 0xA602 followed by 0xB480 within 20 bus clock cycles to watchdog refresh register. If these two values are written more than 20 bus cycles apart or if something other than these two values is written to the register, a watchdog reset, or interrupt-then-reset if enabled, is issued to the system. A valid refresh makes the watchdog timer restart on the next bus clock. Also, an attempted unlock operation in between the two writes of the refresh sequence goes undetected. See Watchdog Operation with 8-bit access for guidelines related to 8-bit accesses to the refresh register.

### 24.3.4 Windowed mode of operation

In this mode of operation, a restriction is placed on the point in time within the time-out period at which the watchdog can be refreshed. The refresh is considered valid only when the watchdog timer increments beyond a certain count as specified by the watchdog window register. This is known as refreshing the watchdog within a window of the total time-out period. If a refresh is attempted before the timer reaches the window value, the watchdog generates a reset, or interrupt-then-reset if enabled. If there is no refresh at all, the watchdog times out and generates a reset or interrupt-then-reset if enabled.

### 24.3.5 Watchdog disabled mode of operation

When the watchdog is disabled through the WDOG_EN bit in the watchdog status and control register, the watchdog timer is reset to zero and is disabled from counting until you enable it or it is enabled again by the system reset. In this mode, the watchdog timer cannot be refreshed–there is no requirement to do so while the timer is disabled. However, the watchdog still generates a reset, or interrupt-then-reset if enabled, on a non-

time-out exception. See Generated Resets and Interrupts. You need to unlock the watchdog before enabling it. A system reset brings the watchdog out of the disabled mode.

### 24.3.6 Debug modes of operation

You can program the watchdog to disable in debug modes through DBG_EN in the watchdog control register. This results in the watchdog timer pausing for the duration of the mode. Register read/writes are still allowed, which means that operations like refresh, unlock, and so on are allowed. Upon exit from the mode, the timer resumes its operation from the point of pausing.

The entry of the system into the mode does not excuse it from compulsorily configuring the watchdog in the WCT time after unlock, unless the system bus clock is gated off, in which case the internal state machine pauses too. Failing to do so still results in a reset, or interrupt-then-reset, if enabled, to the system. Also, all of the exception conditions that result in a reset to the system, as described in Generated Resets and Interrupts, are still valid in mode. So, if an exception condition occurs and the system bus clock is on, a reset occurs, or interrupt-then-reset, if enabled.

The entry into Debug mode within WCT after reset is treated differently. The WDOG timer is kept reset to zero and there is no need to unlock and configure it within WCT. You must not try to refresh or unlock the WDOG in this state or unknown behavior may result. Upon exit from mode, the WDOG timer restarts and the WDOG has to be unlocked and configured within WCT.

## 24.4 Testing the watchdog

For IEC 60730 and other safety standards, the expectation is that anything that monitors a safety function must be tested, and this test is required to be fault tolerant. To test the watchdog, its main timer and its associated compare and reset logic must be tested. To this end, two tests are implemented for the watchdog, as described in Quick Test and Byte Test. A control bit is provided to put the watchdog into functional test mode. There is also an overriding test-disable control bit which allows the functional test mode to be disabled permanently. After it is set, this test-disable bit can only be cleared by a reset.

These two tests achieve the overall aim of testing the counter functioning and the compare and reset logic.

**Note**

Do not enable the watchdog interrupt during these tests. If required, you must ensure that the effective time-out value is greater than WCT time. See Generated Resets and Interrupts for more details.

To run a particular test:

1. Select either quick test or byte test..
2. Set a certain test mode bit to put the watchdog in the functional test mode. Setting this bit automatically switches the watchdog timer to a fast clock source. The switching of the clock source is done to achieve a faster time-out and hence a faster test.

In a successful test, the timer times out after reaching the programmed time-out value and generates a system reset.

**Note**

After emerging from a reset due to a watchdog test, unlock and configure the watchdog. The refresh and unlock operations and interrupt are not automatically disabled in the test mode.

### 24.4.1 Quick test

In this test, the time-out value of watchdog timer is programmed to a very low value to achieve quick time-out. The only difference between the quick test and the normal mode of the watchdog is that TESTWDOG is set for the quick test. This allows for a faster test of the watchdog reset mechanism.

### 24.4.2 Byte test

The byte test is a more thorough a test of the watchdog timer. In this test, the timer is split up into its constituent byte-wide stages that are run independently and tested for time-out against the corresponding byte of the time-out value register. The following figure explains the splitting concept:

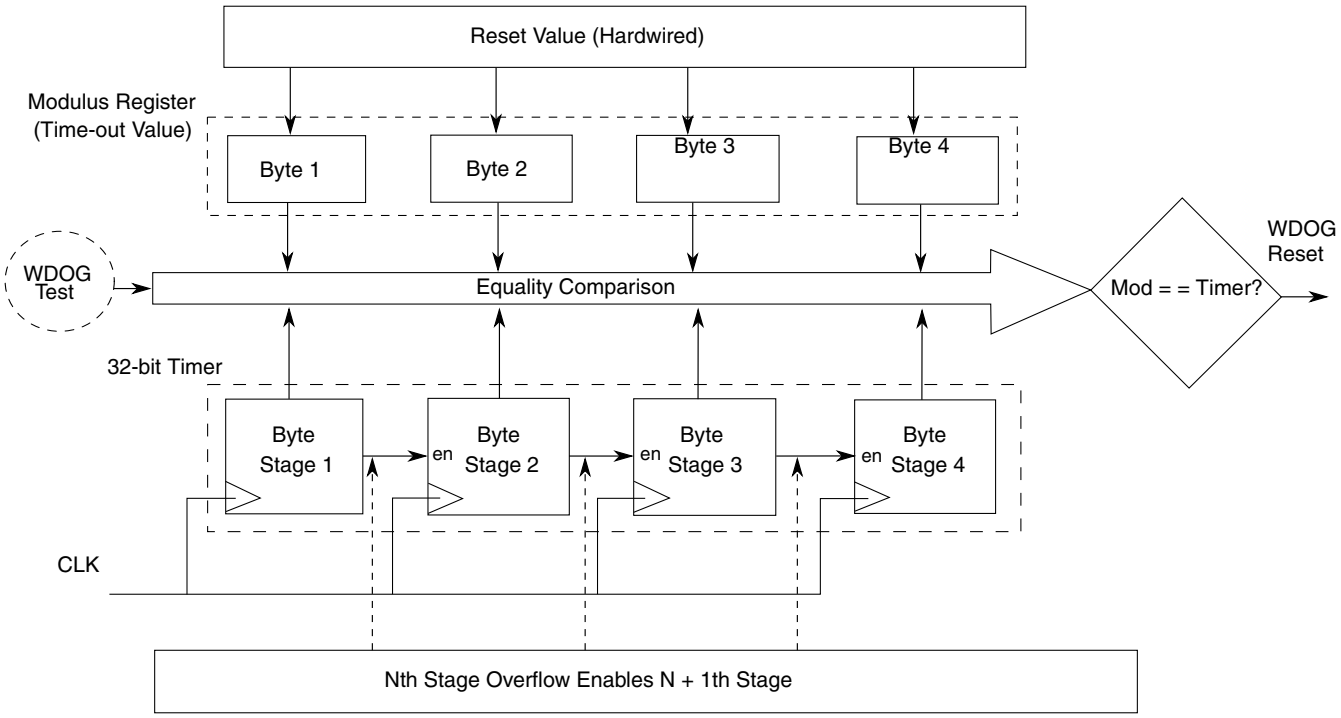



**Figure 24-2. Watchdog timer byte splitting**

Each stage is an 8-bit synchronous counter followed by combinational logic that generates an overflow signal. The overflow signal acts as an enable to the N + 1th stage.

In the test mode, when an individual byte, N, is tested, byte N – 1 is loaded forcefully with 0xFF, and both these bytes are allowed to run off the clock source. By doing so, the overflow signal from stage N – 1 is generated immediately, enabling counter stage N. The Nth stage runs and compares with the Nth byte of the time-out value register. In this way, the byte N is also tested along with the link between it and the preceding stage. No other stages, N – 2, N – 3... and N + 1, N + 2... are enabled for the test on byte N. These disabled stages, except the most significant stage of the counter, are loaded with a value of 0xFF.

## 24.5 Backup reset generator

The backup reset generator generates the final reset which goes out to the system. It has a backup mechanism which ensures that in case the bus clock stops and prevents the main state machine from generating a reset exception/interrupt, the watchdog timer's time-out is separately routed out as a reset to the system. Two successive timer time-outs without an intervening system reset result in the backup reset generator routing out the time-out signal as a reset to the system.

## 24.6 Generated resets and interrupts

The watchdog generates a reset in the following events, also referred to as exceptions:

- A watchdog time-out
- Failure to unlock the watchdog within WCT time after system reset deassertion
- No update of the control and configuration registers within the WCT window after unlocking. At least one of the following registers must be written to within the WCT window to avoid reset:
  - WDOG_ST_CTRL_H, WDOG_ST_CTRL_L
  - WDOG_TO_VAL_H, WDOG_TO_VAL_L
  - WDOG_WIN_H, WDOG_WIN_L
  - WDOG_PRESCALER
- A value other than the unlock sequence or the refresh sequence is written to the unlock and/or refresh registers, respectively.
- A gap of more than 20 bus cycles exists between the writes of two values of the unlock sequence.
- A gap of more than 20 bus cycles exists between the writes of two values of the refresh sequence.

The watchdog can also generate an interrupt. If IRQ_RST_EN is set, then on the above mentioned events WDOG_ST_CTRL_L[INT_FLG] is set, generating an interrupt. A watchdog reset is also generated WCT time later to ensure the watchdog is fault tolerant. The interrupt can be cleared by writing 1 to INT_FLG.

The gap of WCT between interrupt and reset means that the WDOG time-out value must be greater than WCT. Otherwise, if the interrupt was generated due to a time-out, a second consecutive time-out will occur in that WCT gap. This will trigger the backup reset generator to generate a reset to the system, prematurely ending the interrupt service routine execution. Also, jobs such as counting the number of watchdog resets would not be done.

## 24.7 Memory map and register definition

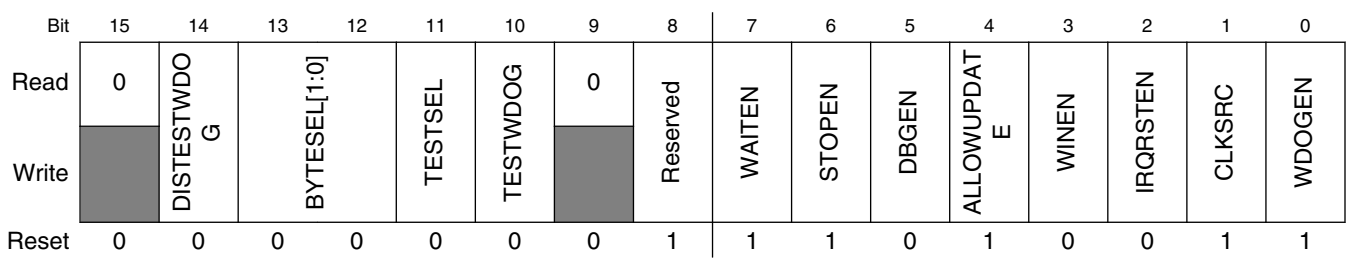
This section consists of the memory map and register descriptions.

### WDOG memory map

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4005_2000 | Watchdog Status and Control Register High (WDOG_STCTRLH) | 16 | R/W | 01D3h | 24.7.1/523 |
| 4005_2002 | Watchdog Status and Control Register Low (WDOG_STCTRLL) | 16 | R/W | 0001h | 24.7.2/525 |
| 4005_2004 | Watchdog Time-out Value Register High (WDOG_TOVALH) | 16 | R/W | 004Ch | 24.7.3/525 |
| 4005_2006 | Watchdog Time-out Value Register Low (WDOG_TOVALL) | 16 | R/W | 4B4Ch | 24.7.4/526 |
| 4005_2008 | Watchdog Window Register High (WDOG_WINH) | 16 | R/W | 0000h | 24.7.5/526 |
| 4005_200A | Watchdog Window Register Low (WDOG_WINL) | 16 | R/W | 0010h | 24.7.6/527 |
| 4005_200C | Watchdog Refresh register (WDOG_REFRESH) | 16 | R/W | B480h | 24.7.7/527 |
| 4005_200E | Watchdog Unlock register (WDOG_UNLOCK) | 16 | R/W | D928h | 24.7.8/527 |
| 4005_2010 | Watchdog Timer Output Register High (WDOG_TMROUTH) | 16 | R/W | 0000h | 24.7.9/528 |
| 4005_2012 | Watchdog Timer Output Register Low (WDOG_TMROUTL) | 16 | R/W | 0000h | 24.7.10/ 528 |
| 4005_2014 | Watchdog Reset Count register (WDOG_RSTCNT) | 16 | R/W | 0000h | 24.7.11/ 529 |
| 4005_2016 | Watchdog Prescaler register (WDOG_PRESC) | 16 | R/W | 0400h | 24.7.12/ 529 |

## 24.7.1 Watchdog Status and Control Register High (WDOG_STCTRLH)

Address: 4005_2000h base + 0h offset = 4005_2000h




### WDOG_STCTRLH field descriptions

| Field | Description |
|---|---|
| 15<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 14<br>DISTESTWDOG | Allows the WDOG's functional test mode to be disabled permanently. After it is set, it can only be cleared by a reset. It cannot be unlocked for editing after it is set.<br><br>0 WDOG functional test mode is not disabled.<br>1 WDOG functional test mode is disabled permanently until reset. |
| 13–12<br>BYTESEL[1:0] | This 2-bit field selects the byte to be tested when the watchdog is in the byte test mode. |

*Table continues on the next page...*

## WDOG_STCTRLH field descriptions (continued)

| Field | Description |
|---|---|
| | 00 Byte 0 selected<br>01 Byte 1 selected<br>10 Byte 2 selected<br>11 Byte 3 selected |
| 11<br>TESTSEL | Effective only if TESTWDOG is set. Selects the test to be run on the watchdog timer.<br><br>0 Quick test. The timer runs in normal operation. You can load a small time-out value to do a quick test.<br>1 Byte test. Puts the timer in the byte test mode where individual bytes of the timer are enabled for operation and are compared for time-out against the corresponding byte of the programmed time-out value. Select the byte through BYTESEL[1:0] for testing. |
| 10<br>TESTWDOG | Puts the watchdog in the functional test mode. In this mode, the watchdog timer and the associated compare and reset generation logic is tested for correct operation. The clock for the timer is switched from the main watchdog clock to the fast clock input for watchdog functional test. The TESTSEL bit selects the test to be run. |
| 9<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 8<br>Reserved | This field is reserved. |
| 7<br>WAITEN | Enables or disables WDOG in Wait mode.<br><br>0 WDOG is disabled in CPU Wait mode.<br>1 WDOG is enabled in CPU Wait mode. |
| 6<br>STOPEN | Enables or disables WDOG in Stop mode.<br><br>0 WDOG is disabled in CPU Stop mode.<br>1 WDOG is enabled in CPU Stop mode. |
| 5<br>DBGEN | Enables or disables WDOG in Debug mode.<br><br>0 WDOG is disabled in CPU Debug mode.<br>1 WDOG is enabled in CPU Debug mode. |
| 4<br>ALLOWUPDATE | Enables updates to watchdog write-once registers, after the reset-triggered initial configuration window (WCT) closes, through unlock sequence.<br><br>0 No further updates allowed to WDOG write-once registers.<br>1 WDOG write-once registers can be unlocked for updating. |
| 3<br>WINEN | Enables Windowing mode.<br><br>0 Windowing mode is disabled.<br>1 Windowing mode is enabled. |
| 2<br>IRQRSTEN | Used to enable the debug breadcrumbs feature. A change in this bit is updated immediately, as opposed to updating after WCT.<br><br>0 WDOG time-out generates reset only.<br>1 WDOG time-out initially generates an interrupt. After WCT, it generates a reset. |
| 1<br>CLKSRC | Selects clock source for the WDOG timer and other internal timing operations.<br><br>0 WDOG clock sourced from LPO .<br>1 WDOG clock sourced from alternate clock source. |

*Table continues on the next page...*

**WDOG_STCTRLH field descriptions (continued)**

| Field | Description |
|---|---|
| 0<br>WDOGEN | Enables or disables the WDOG's operation. In the disabled state, the watchdog timer is kept in the reset state, but the other exception conditions can still trigger a reset/interrupt. A change in the value of this bit must be held for more than one WDOG_CLK cycle for the WDOG to be enabled or disabled.<br><br>0 WDOG is disabled.<br>1 WDOG is enabled. |

## 24.7.2 Watchdog Status and Control Register Low (WDOG_STCTRLL)

Address: 4005_2000h base + 2h offset = 4005_2002h

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | INTFLG | Reserved | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | Reserved | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**WDOG_STCTRLL field descriptions**

| Field | Description |
|---|---|
| 15<br>INTFLG | Interrupt flag. It is set when an exception occurs. IRQRSTEN = 1 is a precondition to set this flag. INTFLG = 1 results in an interrupt being issued followed by a reset, WCT later. The interrupt can be cleared by writing 1 to this bit. It also gets cleared on a system reset. |
| 14–0<br>Reserved | This field is reserved.<br><br>**NOTE:** Do not modify this field value. |

## 24.7.3 Watchdog Time-out Value Register High (WDOG_TOVALH)

Address: 4005_2000h base + 4h offset = 4005_2004h

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read<br>Write | TOVALHIGH | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |

**WDOG_TOVALH field descriptions**

| Field | Description |
|---|---|
| 15–0<br>TOVALHIGH | Defines the upper 16 bits of the 32-bit time-out value for the watchdog timer. It is defined in terms of cycles of the watchdog clock. |

## 24.7.4 Watchdog Time-out Value Register Low (WDOG_TOVALL)

The time-out value of the watchdog must be set to a minimum of four watchdog clock cycles. This is to take into account the delay in new settings taking effect in the watchdog clock domain.

Address: 4005_2000h base + 6h offset = 4005_2006h

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read<br>Write | TOVALLOW | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |

### WDOG_TOVALL field descriptions

| Field | Description |
|---|---|
| 15–0<br>TOVALLOW | Defines the lower 16 bits of the 32-bit time-out value for the watchdog timer. It is defined in terms of cycles of the watchdog clock. |

## 24.7.5 Watchdog Window Register High (WDOG_WINH)

**NOTE**

You must set the Window Register value lower than the Time-out Value Register.

Address: 4005_2000h base + 8h offset = 4005_2008h

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read<br>Write | WINHIGH | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### WDOG_WINH field descriptions

| Field | Description |
|---|---|
| 15–0<br>WINHIGH | Defines the upper 16 bits of the 32-bit window for the windowed mode of operation of the watchdog. It is defined in terms of cycles of the watchdog clock. In this mode, the watchdog can be refreshed only when the timer has reached a value greater than or equal to this window length. A refresh outside this window resets the system or if IRQRSTEN is set, it interrupts and then resets the system. |

## 24.7.6 Watchdog Window Register Low (WDOG_WINL)

**NOTE**

You must set the Window Register value lower than the Time-out Value Register.

Address: 4005_2000h base + Ah offset = 4005_200Ah

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read<br>Write | WINLOW | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**WDOG_WINL field descriptions**

| Field | Description |
|---|---|
| 15–0<br>WINLOW | Defines the lower 16 bits of the 32-bit window for the windowed mode of operation of the watchdog. It is defined in terms of cycles of the pre-scaled watchdog clock. In this mode, the watchdog can be refreshed only when the timer reaches a value greater than or equal to this window length value. A refresh outside of this window resets the system or if IRQRSTEN is set, it interrupts and then resets the system. |

## 24.7.7 Watchdog Refresh register (WDOG_REFRESH)

Address: 4005_2000h base + Ch offset = 4005_200Ch

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read<br>Write | WDOGREFRESH | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**WDOG_REFRESH field descriptions**

| Field | Description |
|---|---|
| 15–0<br>WDOGREFRESH | Watchdog refresh register. A sequence of 0xA602 followed by 0xB480 within 20 bus clock cycles written to this register refreshes the WDOG and prevents it from resetting the system. Writing a value other than the above mentioned sequence or if the sequence is longer than 20 bus cycles, resets the system, or if IRQRSTEN is set, it interrupts and then resets the system. |

## 24.7.8 Watchdog Unlock register (WDOG_UNLOCK)

Address: 4005_2000h base + Eh offset = 4005_200Eh

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read<br>Write | WDOGUNLOCK | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |

### WDOG_UNLOCK field descriptions

| Field | Description |
|---|---|
| 15–0<br>WDOGUNLOCK | Writing the unlock sequence values to this register to makes the watchdog write-once registers writable again. The required unlock sequence is 0xC520 followed by 0xD928 within 20 bus clock cycles. A valid unlock sequence opens a window equal in length to the WCT within which you can update the registers. Writing a value other than the above mentioned sequence or if the sequence is longer than 20 bus cycles, resets the system or if IRQRSTEN is set, it interrupts and then resets the system. The unlock sequence is effective only if ALLOWUPDATE is set. |

## 24.7.9 Watchdog Timer Output Register High (WDOG_TMROUTH)

Address: 4005_2000h base + 10h offset = 4005_2010h

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read<br>Write | TIMEROUTHIGH | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### WDOG_TMROUTH field descriptions

| Field | Description |
|---|---|
| 15–0<br>TIMEROUTHIGH | Shows the value of the upper 16 bits of the watchdog timer. |

## 24.7.10 Watchdog Timer Output Register Low (WDOG_TMROUTL)

During Stop mode, the WDOG_TIMER_OUT will be caught at the pre-stop value of the watchdog timer. After exiting Stop mode, a maximum delay of 1 WDOG_CLK cycle + 3 bus clock cycles will occur before the WDOG_TIMER_OUT starts following the watchdog timer.

Address: 4005_2000h base + 12h offset = 4005_2012h

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read<br>Write | TIMEROUTLOW | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### WDOG_TMROUTL field descriptions

| Field | Description |
|---|---|
| 15–0<br>TIMEROUTLOW | Shows the value of the lower 16 bits of the watchdog timer. |

## 24.7.11 Watchdog Reset Count register (WDOG_RSTCNT)

Address: 4005_2000h base + 14h offset = 4005_2014h

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read<br>Write | RSTCNT | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### WDOG_RSTCNT field descriptions

| Field | Description |
|---|---|
| 15–0<br>RSTCNT | Counts the number of times the watchdog resets the system. This register is reset only on a POR. Writing 1 to the bit to be cleared enables you to clear the contents of this register. |

## 24.7.12 Watchdog Prescaler register (WDOG_PRESC)

Address: 4005_2000h base + 16h offset = 4005_2016h

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read | 0 | | | | | PRESCVAL | | | 0 | | | | | | | |
| Write | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### WDOG_PRESC field descriptions

| Field | Description |
|---|---|
| 15–11<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 10–8<br>PRESCVAL | 3-bit prescaler for the watchdog clock source. A value of zero indicates no division of the input WDOG clock. The watchdog clock is divided by (PRESCVAL + 1) to provide the prescaled WDOG_CLK. |
| 7–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

# 24.8 Watchdog operation with 8-bit access

## 24.8.1 General guideline

When performing 8-bit accesses to the watchdog's 16-bit registers where the intention is to access both the bytes of a register, place the two 8-bit accesses one after the other in your code.

## 24.8.2 Refresh and unlock operations with 8-bit access

One exception condition that generates a reset to the system is the write of any value other than those required for a legal refresh/update sequence to the respective refresh and unlock registers.

For an 8-bit access to these registers, writing a correct value requires at least two bus clock cycles, resulting in an invalid value in the registers for one cycle. Therefore, the system is reset even if the intention is to write a correct value to the refresh/unlock register. Keeping this in mind, the exception condition for 8-bit accesses is slightly modified.

Whereas the match for a correct value for a refresh/unlock sequence is as according to the original definition, the match for an incorrect value is done byte-wise on the refresh/unlock rather than for the whole 16-bit value. This means that if the high byte of the refresh/unlock register contains any value other than high bytes of the two values that make up the sequence, it is treated as an exception condition, leading to a reset or interrupt-then-reset. The same holds true for the lower byte of the refresh or unlock register. Take the refresh operation that expects a write of 0xA602 followed by 0xB480 to the refresh register, as an example.

**Table 24-14. Refresh for 8-bit access**

| | WDOG_REFRESH[15:8] | WDOG_REFRESH[7:0] | Sequence value1 or value2 match | Mismatch exception |
|---|---|---|---|---|
| **Current Value** | 0xB4 | 0x80 | Value2 match | No |
| **Write 1** | 0xB4 | 0x02 | No match | No |
| **Write 2** | 0xA6 | 0x02 | Value1 match | No |
| **Write 3** | 0xB4 | 0x02 | No match | No |
| **Write 4** | 0xB4 | 0x80 | Value2 match. Sequence complete. | No |
| **Write 5** | 0x02 | 0x80 | No match | Yes |

As shown in the preceding table, the refresh register holds its reset value initially. Thereafter, two 8-bit accesses are performed on the register to write the first value of the refresh sequence. No mismatch exception is registered on the intermediate write, Write1. The sequence is completed by performing two more 8-bit accesses, writing in the second value of the sequence for a successful refresh. It must be noted that the match of value2 takes place only when the complete 16-bit value is correctly written, write4. Hence, the requirement of writing value2 of the sequence within 20 bus clock cycles of value1 is checked by measuring the gap between write2 and write4.

It is reiterated that the condition for matching values 1 and 2 of the refresh or unlock sequence remains unchanged. The difference for 8-bit accesses is that the criterion for detecting a mismatch is less strict. Any 16-bit access still needs to adhere to the original guidelines, mentioned in the sections Refreshing the Watchdog.

## 24.9 Restrictions on watchdog operation

This section mentions some exceptions to the watchdog operation that may not be apparent to you.

- Restriction on unlock/refresh operations—In the period between the closure of the WCT window after unlock and the actual reload of the watchdog timer, unlock and refresh operations need not be attempted.
- The update and reload of the watchdog timer happens two to three watchdog clocks after WCT window closes, following a successful configuration on unlock.
- Clock Switching Delay—The watchdog uses glitch-free multiplexers at two places – one to choose between the LPO oscillator input and alternate clock input, and the other to choose between the watchdog functional clock and fast clock input for watchdog functional test. A maximum time period of ~2 clock A cycles plus ~2 clock B cycles elapses from the time a switch is requested to the occurrence of the actual clock switch, where clock A and B are the two input clocks to the clock mux.
- For the windowed mode, there is a two to three bus clock latency between the watchdog counter going past the window value and the same registering in the bus clock domain.
- For proper operation of the watchdog, the watchdog clock must be at least five times slower than the system bus clock at all times. An exception is when the watchdog clock is synchronous to the bus clock wherein the watchdog clock can be as fast as the bus clock.
- WCT must be equivalent to at least three watchdog clock cycles. If not ensured, this means that even after the close of the WCT window, you have to wait for the synchronized system reset to deassert in the watchdog clock domain, before expecting the configuration updates to take effect.
- The time-out value of the watchdog should be set to a minimum of four watchdog clock cycles. This is to take into account the delay in new settings taking effect in the watchdog clock domain.

- You must take care not only to refresh the watchdog within the watchdog timer's actual time-out period, but also provide enough allowance for the time it takes for the refresh sequence to be detected by the watchdog timer, on the watchdog clock.
- Updates cannot be made in the bus clock cycle immediately following the write of the unlock sequence, but one bus clock cycle later.
- It should be ensured that the time-out value for the watchdog is always greater than 2xWCT time + 20 bus clock cycles.
- An attempted refresh operation, in between the two writes of the unlock sequence and in the WCT time following a successful unlock, will go undetected.
- Trying to unlock the watchdog within the WCT time after an initial unlock has no effect.
- The refresh and unlock operations and interrupt are not automatically disabled in the watchdog functional test mode.
- After emerging from a reset due to a watchdog functional test, you are still expected to go through the mandatory steps of unlocking and configuring the watchdog. The watchdog continues to be in its functional test mode and therefore you should pull the watchdog out of the functional test mode within WCT time of reset.
- After emerging from a reset due to a watchdog functional test, you still need to go through the mandatory steps of unlocking and configuring the watchdog.
- You must ensure that both the clock inputs to the glitchless clock multiplexers are alive during the switching of clocks. Failure to do so results in a loss of clock at their outputs.
- There is a gap of two to three watchdog clock cycles from the point that stop mode is entered to the watchdog timer actually pausing, due to synchronization. The same holds true for an exit from the stop mode, this time resulting in a two to three watchdog clock cycle delay in the timer restarting. In case the duration of the stop mode is less than one watchdog clock cycle, the watchdog timer is not guaranteed to pause.
- Consider the case when the first refresh value is written, following which the system enters stop mode with system bus clk still on. If the second refresh value is not written within 20 bus cycles of the first value, the system is reset, or interrupt-then-reset if enabled.

# Chapter 25
# Multipurpose Clock Generator (MCG)

## 25.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The multipurpose clock generator (MCG) module provides several clock source choices for the MCU. The module contains a frequency-locked loop (FLL) and a phase-locked loop (PLL). The FLL is controllable by either an internal or an external reference clock. The PLL is controllable by the external reference clock. The module can select either of the FLL or PLL output clocks, or either of the internal or external reference clocks as a source for the MCU system clock. The MCG operates in conjuction with a crystal oscillator, which allows an external crystal, ceramic resonator, or another external clock source to produce the external reference clock.

### 25.1.1 Features

Key features of the MCG module are:

- Frequency-locked loop (FLL):
  - Digitally-controlled oscillator (DCO)
  - DCO frequency range is programmable for up to four different frequency ranges.
  - Option to program and maximize DCO output frequency for a low frequency external reference clock source.
  - Option to prevent FLL from resetting its current locked frequency when switching clock modes if FLL reference frequency is not changed.

- Internal or external reference clock can be used as the FLL source.
- Can be used as a clock source for other on-chip peripherals.

- Phase-locked loop (PLL):
  - Voltage-controlled oscillator (VCO)
  - External reference clock is used as the PLL source.
  - Modulo VCO frequency divider
  - Phase/Frequency detector
  - Integrated loop filter
  - Can be used as a clock source for other on-chip peripherals.
- Internal reference clock generator:
  - Slow clock with nine trim bits for accuracy
  - Fast clock with four trim bits
  - Can be used as source clock for the FLL. In FEI mode, only the slow Internal Reference Clock (IRC) can be used as the FLL source.
  - Either the slow or the fast clock can be selected as the clock source for the MCU.
  - Can be used as a clock source for other on-chip peripherals.
- 
  - Can be used as a second PLL source
  - Can be selected as the clock source for the MCU.
- Control signals for the MCG external reference low power oscillator clock generators are provided:
  - HGO0, RANGE0, EREFS0
- External clock from the Crystal Oscillator :
  - Can be used as a source for the FLL and/or the PLL.
  - Can be selected as the clock source for the MCU.
- External clock from the Real Time Counter (RTC):
  - Can only be used as a source for the FLL.
  - Can be selected as the clock source for the MCU.

- External clock monitor with reset and interrupt request capability to check for external clock failure when running in FBE, PEE, BLPE, or FEE modes
- Lock detector with interrupt request capability for use with the PLL
- Internal Reference Clocks Auto Trim Machine (ATM) capability using an external clock as a reference
- Reference dividers for both the FLL and the PLL are provided
- Reference dividers for the Fast Internal Reference Clock are provided
- MCG PLL Clock (MCGPLLCLK) is provided as a clock source for other on-chip peripherals
- 
- MCG FLL Clock (MCGFLLCLK) is provided as a clock source for other on-chip peripherals
- MCG Fixed Frequency Clock (MCGFFCLK) is provided as a clock source for other on-chip peripherals
- MCG Internal Reference Clock (MCGIRCLK) is provided as a clock source for other on-chip peripherals

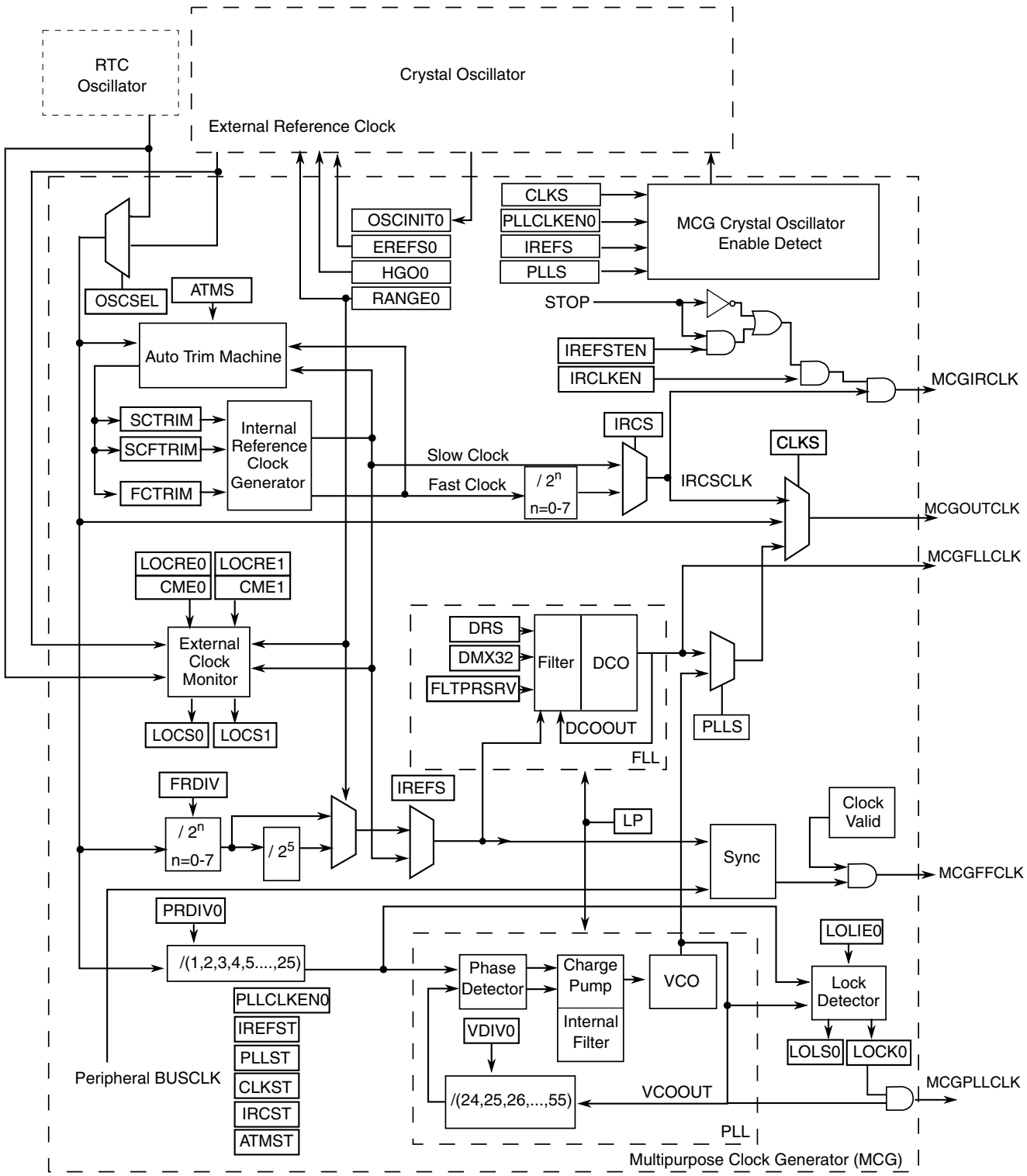



**Figure 25-1. Multipurpose Clock Generator (MCG) block diagram**

### 25.1.2 Modes of Operation

The MCG has the following modes of operation: FEI, FEE, FBI, FBE, PBE, PEE, BLPI, BLPE, and Stop. For details, see MCG modes of operation.

## 25.2 External Signal Description

There are no MCG signals that connect off chip.

## 25.3 Memory Map/Register Definition

This section includes the memory map and register definition.

The MCG registers can only be written when in supervisor mode. Write accesses when in user mode will result in a bus error. Read accesses may be performed in both supervisor and user mode.

**MCG memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4006_4000 | MCG Control 1 Register (MCG_C1) | 8 | R/W | 04h | 25.3.1/538 |
| 4006_4001 | MCG Control 2 Register (MCG_C2) | 8 | R/W | 80h | 25.3.2/539 |
| 4006_4002 | MCG Control 3 Register (MCG_C3) | 8 | R/W | Undefined | 25.3.3/540 |
| 4006_4003 | MCG Control 4 Register (MCG_C4) | 8 | R/W | Undefined | 25.3.4/541 |
| 4006_4004 | MCG Control 5 Register (MCG_C5) | 8 | R/W | 00h | 25.3.5/542 |
| 4006_4005 | MCG Control 6 Register (MCG_C6) | 8 | R/W | 00h | 25.3.6/543 |
| 4006_4006 | MCG Status Register (MCG_S) | 8 | R | 10h | 25.3.7/545 |
| 4006_4008 | MCG Status and Control Register (MCG_SC) | 8 | R/W | 02h | 25.3.8/546 |
| 4006_400A | MCG Auto Trim Compare Value High Register (MCG_ATCVH) | 8 | R/W | 00h | 25.3.9/548 |
| 4006_400B | MCG Auto Trim Compare Value Low Register (MCG_ATCVL) | 8 | R/W | 00h | 25.3.10/ 548 |
| 4006_400C | MCG Control 7 Register (MCG_C7) | 8 | R/W | 00h | 25.3.11/ 548 |
| 4006_400D | MCG Control 8 Register (MCG_C8) | 8 | R/W | 80h | 25.3.12/ 549 |

## 25.3.1 MCG Control 1 Register (MCG_C1)

Address: 4006_4000h base + 0h offset = 4006_4000h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | CLKS | | FRDIV | | | IREFS | IRCLKEN | IREFSTEN |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

### MCG_C1 field descriptions

| Field | Description |
|---|---|
| 7–6<br>CLKS | Clock Source Select<br><br>Selects the clock source for MCGOUTCLK .<br><br>00 Encoding 0 — Output of FLL or PLL is selected (depends on PLLS control bit).<br>01 Encoding 1 — Internal reference clock is selected.<br>10 Encoding 2 — External reference clock is selected.<br>11 Encoding 3 — Reserved. |
| 5–3<br>FRDIV | FLL External Reference Divider<br><br>Selects the amount to divide down the external reference clock for the FLL. The resulting frequency must be in the range 31.25 kHz to 39.0625 kHz (This is required when FLL/DCO is the clock source for MCGOUTCLK . In FBE mode, it is not required to meet this range, but it is recommended in the cases when trying to enter a FLL mode from FBE).<br><br>000 If RANGE 0 = 0 or OSCSEL=1 , Divide Factor is 1; for all other RANGE 0 values, Divide Factor is 32.<br>001 If RANGE 0 = 0 or OSCSEL=1 , Divide Factor is 2; for all other RANGE 0 values, Divide Factor is 64.<br>010 If RANGE 0 = 0 or OSCSEL=1 , Divide Factor is 4; for all other RANGE 0 values, Divide Factor is 128.<br>011 If RANGE 0 = 0 or OSCSEL=1 , Divide Factor is 8; for all other RANGE 0 values, Divide Factor is 256.<br>100 If RANGE 0 = 0 or OSCSEL=1 , Divide Factor is 16; for all other RANGE 0 values, Divide Factor is 512.<br>101 If RANGE 0 = 0 or OSCSEL=1 , Divide Factor is 32; for all other RANGE 0 values, Divide Factor is 1024.<br>110 If RANGE 0 = 0 or OSCSEL=1 , Divide Factor is 64; for all other RANGE 0 values, Divide Factor is 1280 .<br>111 If RANGE 0 = 0 or OSCSEL=1 , Divide Factor is 128; for all other RANGE 0 values, Divide Factor is 1536 . |
| 2<br>IREFS | Internal Reference Select<br><br>Selects the reference clock source for the FLL.<br><br>0 External reference clock is selected.<br>1 The slow internal reference clock is selected. |
| 1<br>IRCLKEN | Internal Reference Clock Enable<br><br>Enables the internal reference clock for use as MCGIRCLK. |

*Table continues on the next page...*

**MCG_C1 field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 MCGIRCLK inactive.<br>1 MCGIRCLK active. |
| 0<br>IREFSTEN | Internal Reference Stop Enable<br><br>Controls whether or not the internal reference clock remains enabled when the MCG enters Stop mode.<br><br>0 Internal reference clock is disabled in Stop mode.<br>1 Internal reference clock is enabled in Stop mode if IRCLKEN is set or if MCG is in FEI, FBI, or BLPI modes before entering Stop mode. |

## 25.3.2 MCG Control 2 Register (MCG_C2)

Address: 4006_4000h base + 1h offset = 4006_4001h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | LOCRE0 | 0 | RANGE0 | | HGO0 | EREFS0 | LP | IRCS |
| Write | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MCG_C2 field descriptions**

| Field | Description |
|---|---|
| 7<br>LOCRE0 | Loss of Clock Reset Enable<br><br>Determines whether an interrupt or a reset request is made following a loss of OSC0 external reference clock. The LOCRE0 only has an affect when CME0 is set.<br><br>0 Interrupt request is generated on a loss of OSC0 external reference clock.<br>1 Generate a reset request on a loss of OSC0 external reference clock. |
| 6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5–4<br>RANGE0 | Frequency Range Select<br><br>Selects the frequency range for the crystal oscillator or external clock source. See the Oscillator (OSC) chapter for more details and the device data sheet for the frequency ranges used.<br><br>00 Encoding 0 — Low frequency range selected for the crystal oscillator .<br>01 Encoding 1 — High frequency range selected for the crystal oscillator .<br>1X Encoding 2 — Very high frequency range selected for the crystal oscillator . |
| 3<br>HGO0 | High Gain Oscillator Select<br><br>Controls the crystal oscillator mode of operation. See the Oscillator (OSC) chapter for more details.<br><br>0 Configure crystal oscillator for low-power operation.<br>1 Configure crystal oscillator for high-gain operation. |
| 2<br>EREFS0 | External Reference Select<br><br>Selects the source for the external reference clock. See the Oscillator (OSC) chapter for more details. |

*Table continues on the next page...*

**MCG_C2 field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 External reference clock requested.<br>1 Oscillator requested. |
| 1<br>LP | Low Power Select<br><br>Controls whether the FLL or PLL is disabled in BLPI and BLPE modes. In FBE or PBE modes, setting this bit to 1 will transition the MCG into BLPE mode; in FBI mode, setting this bit to 1 will transition the MCG into BLPI mode. In any other MCG mode, LP bit has no affect.<br><br>0 FLL or PLL is not disabled in bypass modes.<br>1 FLL or PLL is disabled in bypass modes (lower power) |
| 0<br>IRCS | Internal Reference Clock Select<br><br>Selects between the fast or slow internal reference clock source.<br><br>0 Slow internal reference clock selected.<br>1 Fast internal reference clock selected. |

## 25.3.3 MCG Control 3 Register (MCG_C3)

Address: 4006_4000h base + 2h offset = 4006_4002h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | SCTRIM | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**MCG_C3 field descriptions**

| Field | Description |
|---|---|
| 7–0<br>SCTRIM | Slow Internal Reference Clock Trim Setting<br><br>SCTRIM [1] controls the slow internal reference clock frequency by controlling the slow internal reference clock period. The SCTRIM bits are binary weighted, that is, bit 1 adjusts twice as much as bit 0. Increasing the binary value increases the period, and decreasing the value decreases the period.<br><br>An additional fine trim bit is available in C4 register as the SCFTRIM bit. Upon reset, this value is loaded with a factory trim value.<br><br>If an SCTRIM value stored in nonvolatile memory is to be used, it is your responsibility to copy that value from the nonvolatile memory location to this register. |

1. A value for SCTRIM is loaded during reset from a factory programmed location.

## 25.3.4 MCG Control 4 Register (MCG_C4)

### NOTE

Reset values for DRST and DMX32 bits are 0.

Address: 4006_4000h base + 3h offset = 4006_4003h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | DMX32 | DRST_DRS | | | FCTRIM | | | SCFTRIM |
| Reset | 0 | 0 | 0 | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.
- A value for FCTRIM is loaded during reset from a factory programmed location. x = Undefined at reset.

### MCG_C4 field descriptions

| Field | Description |
|---|---|
| 7<br>DMX32 | DCO Maximum Frequency with 32.768 kHz Reference<br><br>The DMX32 bit controls whether the DCO frequency range is narrowed to its maximum frequency with a 32.768 kHz reference.<br><br>The following table identifies settings for the DCO frequency range.<br><br>**NOTE:** The system clocks derived from this source should not exceed their specified maximums.<br><br>0 DCO has a default range of 25%.<br>1 DCO is fine-tuned for maximum frequency with 32.768 kHz reference. |
| 6–5<br>DRST_DRS | DCO Range Select<br><br>The DRS bits select the frequency range for the FLL output, DCOOUT. When the LP bit is set, writes to the DRS bits are ignored. The DRST read field indicates the current frequency range for DCOOUT. The DRST field does not update immediately after a write to the DRS field due to internal synchronization between clock domains. See the DCO Frequency Range table for more details.<br><br>00 Encoding 0 — Low range (reset default). |

| DRST_DRS | DMX32 | Reference Range | FLL Factor | DCO Range |
|---|---|---|---|---|
| 00 | 0 | 31.25–39.0625 kHz | 640 | 20–25 MHz |
| | 1 | 32.768 kHz | 732 | 24 MHz |
| 01 | 0 | 31.25–39.0625 kHz | 1280 | 40–50 MHz |
| | 1 | 32.768 kHz | 1464 | 48 MHz |
| 10 | 0 | 31.25–39.0625 kHz | 1920 | 60–75 MHz |
| | 1 | 32.768 kHz | 2197 | 72 MHz |
| 11 | 0 | 31.25–39.0625 kHz | 2560 | 80–100 MHz |
| | 1 | 32.768 kHz | 2929 | 96 MHz |

*Table continues on the next page...*

**MCG_C4 field descriptions (continued)**

| Field | Description |
|---|---|
| | 01 Encoding 1 — Mid range.<br>10 Encoding 2 — Mid-high range.<br>11 Encoding 3 — High range. |
| 4–1<br>FCTRIM | Fast Internal Reference Clock Trim Setting<br>FCTRIM [1] controls the fast internal reference clock frequency by controlling the fast internal reference clock period. The FCTRIM bits are binary weighted, that is, bit 1 adjusts twice as much as bit 0. Increasing the binary value increases the period, and decreasing the value decreases the period.<br>If an FCTRIM[3:0] value stored in nonvolatile memory is to be used, it is your responsibility to copy that value from the nonvolatile memory location to this register. |
| 0<br>SCFTRIM | Slow Internal Reference Clock Fine Trim<br>SCFTRIM [2] controls the smallest adjustment of the slow internal reference clock frequency. Setting SCFTRIM increases the period and clearing SCFTRIM decreases the period by the smallest amount possible.<br>If an SCFTRIM value stored in nonvolatile memory is to be used, it is your responsibility to copy that value from the nonvolatile memory location to this bit. |

1. A value for FCTRIM is loaded during reset from a factory programmed location.
2. A value for SCFTRIM is loaded during reset from a factory programmed location .

## 25.3.5 MCG Control 5 Register (MCG_C5)

Address: 4006_4000h base + 4h offset = 4006_4004h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | PLLCLKEN0 | PLLSTEN0 | PRDIV0 | | | | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MCG_C5 field descriptions**

| Field | Description |
|---|---|
| 7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6<br>PLLCLKEN0 | PLL Clock Enable<br>Enables the PLL independent of PLLS and enables the PLL clock for use as MCGPLLCLK. (PRDIV 0 needs to be programmed to the correct divider to generate a PLL reference clock in the range of 2 - 4 MHz range prior to setting the PLLCLKEN 0 bit). Setting PLLCLKEN 0 will enable the external oscillator if not already enabled. Whenever the PLL is being enabled by means of the PLLCLKEN 0 bit, and the external oscillator is being used as the reference clock, the OSCINIT 0 bit should be checked to make sure it is set.<br>0 MCGPLLCLK is inactive.<br>1 MCGPLLCLK is active. |
| 5<br>PLLSTEN0 | PLL Stop Enable |

*Table continues on the next page...*

**MCG_C5 field descriptions (continued)**

| Field | Description |
|---|---|
| | Enables the PLL Clock during Normal Stop. In Low Power Stop mode, the PLL clock gets disabled even if PLLSTEN 0 =1. All other power modes, PLLSTEN 0 bit has no affect and does not enable the PLL Clock to run if it is written to 1.<br><br>0 MCGPLLCLK is disabled in any of the Stop modes.<br>1 MCGPLLCLK is enabled if system is in Normal Stop mode. |
| 4–0<br>PRDIV0 | PLL External Reference Divider<br><br>Selects the amount to divide down the external reference clock for the PLL. The resulting frequency must be in the range of 2 MHz to 4 MHz. After the PLL is enabled (by setting either PLLCLKEN 0 or PLLS), the PRDIV 0 value must not be changed when LOCK0 is zero. |

**Table 25-7. PLL External Reference Divide Factor**

| PRDIV 0 | Divide Factor | | PRDIV 0 | Divide Factor | | PRDIV 0 | Divide Factor | | PRDIV 0 | Divide Factor |
|---|---|---|---|---|---|---|---|---|---|---|
| 00000 | 1 | | 01000 | 9 | | 10000 | 17 | | 11000 | 25 |
| 00001 | 2 | | 01001 | 10 | | 10001 | 18 | | 11001 | Reserved |
| 00010 | 3 | | 01010 | 11 | | 10010 | 19 | | 11010 | Reserved |
| 00011 | 4 | | 01011 | 12 | | 10011 | 20 | | 11011 | Reserved |
| 00100 | 5 | | 01100 | 13 | | 10100 | 21 | | 11100 | Reserved |
| 00101 | 6 | | 01101 | 14 | | 10101 | 22 | | 11101 | Reserved |
| 00110 | 7 | | 01110 | 15 | | 10110 | 23 | | 11110 | Reserved |
| 00111 | 8 | | 01111 | 16 | | 10111 | 24 | | 11111 | Reserved |

## 25.3.6 MCG Control 6 Register (MCG_C6)

Address: 4006_4000h base + 5h offset = 4006_4005h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | LOLIE0 | PLLS | CME0 | VDIV0 | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MCG_C6 field descriptions**

| Field | Description |
|---|---|
| 7<br>LOLIE0 | Loss of Lock Interrrupt Enable |

*Table continues on the next page...*

## MCG_C6 field descriptions (continued)

| Field | Description |
|---|---|
| | Determines if an interrupt request is made following a loss of lock indication. This bit only has an effect when LOLS 0 is set.<br><br>0 No interrupt request is generated on loss of lock.<br>1 Generate an interrupt request on loss of lock. |
| 6<br>PLLS | PLL Select<br><br>Controls whether the PLL or FLL output is selected as the MCG source when CLKS[1:0]=00. If the PLLS bit is cleared and PLLCLKEN 0 is not set, the PLL is disabled in all modes. If the PLLS is set, the FLL is disabled in all modes.<br><br>0 FLL is selected.<br>1 PLL is selected (PRDIV 0 need to be programmed to the correct divider to generate a PLL reference clock in the range of 2–4 MHz prior to setting the PLLS bit). |
| 5<br>CME0 | Clock Monitor Enable<br><br>Enables the loss of clock monitoring circuit for the OSC0 external reference mux select. The LOCRE0 bit will determine if a interrupt or a reset request is generated following a loss of OSC0 indication. The CME0 bit must only be set to a logic 1 when the MCG is in an operational mode that uses the external clock (FEE, FBE, PEE, PBE, or BLPE) . Whenever the CME0 bit is set to a logic 1, the value of the RANGE0 bits in the C2 register should not be changed. CME0 bit should be set to a logic 0 before the MCG enters any Stop mode. Otherwise, a reset request may occur while in Stop mode. CME0 should also be set to a logic 0 before entering VLPR or VLPW power modes if the MCG is in BLPE mode.<br><br>0 External clock monitor is disabled for OSC0.<br>1 External clock monitor is enabled for OSC0. |
| 4–0<br>VDIV0 | VCO 0 Divider<br><br>Selects the amount to divide the VCO output of the PLL. The VDIV 0 bits establish the multiplication factor (M) applied to the reference clock frequency. After the PLL is enabled (by setting either PLLCLKEN 0 or PLLS), the VDIV 0 value must not be changed when LOCK 0 is zero. |

### Table 25-9. PLL VCO Divide Factor

| VDIV 0 | Multiply Factor | | VDIV 0 | Multiply Factor | | VDIV 0 | Multiply Factor | | VDIV 0 | Multiply Factor |
|---|---|---|---|---|---|---|---|---|---|---|
| 00000 | 24 | | 01000 | 32 | | 10000 | 40 | | 11000 | 48 |
| 00001 | 25 | | 01001 | 33 | | 10001 | 41 | | 11001 | 49 |
| 00010 | 26 | | 01010 | 34 | | 10010 | 42 | | 11010 | 50 |
| 00011 | 27 | | 01011 | 35 | | 10011 | 43 | | 11011 | 51 |
| 00100 | 28 | | 01100 | 36 | | 10100 | 44 | | 11100 | 52 |
| 00101 | 29 | | 01101 | 37 | | 10101 | 45 | | 11101 | 53 |
| 00110 | 30 | | 01110 | 38 | | 10110 | 46 | | 11110 | 54 |
| 00111 | 31 | | 01111 | 39 | | 10111 | 47 | | 11111 | 55 |

## 25.3.7 MCG Status Register (MCG_S)

Address: 4006_4000h base + 6h offset = 4006_4006h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | LOLS | LOCK0 | PLLST | IREFST | CLKST | | OSCINIT0 | IRCST |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

### MCG_S field descriptions

| Field | Description |
|---|---|
| 7<br>LOLS | Loss of Lock Status<br><br>This bit is a sticky bit indicating the lock status for the PLL. LOLS is set if after acquiring lock, the PLL output frequency has fallen outside the lock exit frequency tolerance, $D_{unl}$ . LOLIE determines whether an interrupt request is made when LOLS is set. LOLRE determines whether a reset request is made when LOLS is set. This bit is cleared by reset or by writing a logic 1 to it when set. Writing a logic 0 to this bit has no effect.<br><br>0 PLL has not lost lock since LOLS 0 was last cleared.<br>1 PLL has lost lock since LOLS 0 was last cleared. |
| 6<br>LOCK0 | Lock Status<br><br>This bit indicates whether the PLL has acquired lock. Lock detection is only enabled when the PLL is enabled (either through clock mode selection or PLLCLKEN0=1 setting). While the PLL clock is locking to the desired frequency, the MCG PLL clock (MCGPLLCLK) will be gated off until the LOCK bit gets asserted. If the lock status bit is set, changing the value of the PRDIV0 [4:0] bits in the C5 register or the VDIV0[4:0] bits in the C6 register causes the lock status bit to clear and stay cleared until the PLL has reacquired lock. Loss of PLL reference clock will also cause the LOCK0 bit to clear until the PLL has reacquired lock. Entry into LLS, VLPS, or regular Stop with PLLSTEN=0 also causes the lock status bit to clear and stay cleared until the Stop mode is exited and the PLL has reacquired lock. Any time the PLL is enabled and the LOCK0 bit is cleared, the MCGPLLCLK will be gated off until the LOCK0 bit is asserted again.<br><br>0 PLL is currently unlocked.<br>1 PLL is currently locked. |
| 5<br>PLLST | PLL Select Status<br><br>This bit indicates the clock source selected by PLLS . The PLLST bit does not update immediately after a write to the PLLS bit due to internal synchronization between clock domains.<br><br>0 Source of PLLS clock is FLL clock.<br>1 Source of PLLS clock is PLL output clock. |
| 4<br>IREFST | Internal Reference Status<br><br>This bit indicates the current source for the FLL reference clock. The IREFST bit does not update immediately after a write to the IREFS bit due to internal synchronization between clock domains.<br><br>0 Source of FLL reference clock is the external reference clock.<br>1 Source of FLL reference clock is the internal reference clock. |

*Table continues on the next page...*

**MCG_S field descriptions (continued)**

| Field | Description |
|---|---|
| 3–2<br>CLKST | Clock Mode Status<br><br>These bits indicate the current clock mode. The CLKST bits do not update immediately after a write to the CLKS bits due to internal synchronization between clock domains.<br><br>00 Encoding 0 — Output of the FLL is selected (reset default).<br>01 Encoding 1 — Internal reference clock is selected.<br>10 Encoding 2 — External reference clock is selected.<br>11 Encoding 3 — Output of the PLL is selected. |
| 1<br>OSCINIT0 | OSC Initialization<br><br>This bit, which resets to 0, is set to 1 after the initialization cycles of the crystal oscillator clock have completed. After being set, the bit is cleared to 0 if the OSC is subsequently disabled. See the OSC module's detailed description for more information. |
| 0<br>IRCST | Internal Reference Clock Status<br><br>The IRCST bit indicates the current source for the internal reference clock select clock (IRCSCLK). The IRCST bit does not update immediately after a write to the IRCS bit due to internal synchronization between clock domains. The IRCST bit will only be updated if the internal reference clock is enabled, either by the MCG being in a mode that uses the IRC or by setting the C1[IRCLKEN] bit .<br><br>0 Source of internal reference clock is the slow clock (32 kHz IRC).<br>1 Source of internal reference clock is the fast clock (4 MHz IRC). |

## 25.3.8 MCG Status and Control Register (MCG_SC)

Address: 4006_4000h base + 8h offset = 4006_4008h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | ATME | ATMS | ATMF | FLTPRSRV | FCRDIV | | | LOCS0 |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

**MCG_SC field descriptions**

| Field | Description |
|---|---|
| 7<br>ATME | Automatic Trim Machine Enable<br><br>Enables the Auto Trim Machine to start automatically trimming the selected Internal Reference Clock.<br><br>**NOTE:** ATME deasserts after the Auto Trim Machine has completed trimming all trim bits of the IRCS clock selected by the ATMS bit.<br>Writing to C1, C3, C4, and SC registers or entering Stop mode aborts the auto trim operation and clears this bit.<br><br>0 Auto Trim Machine disabled.<br>1 Auto Trim Machine enabled. |

*Table continues on the next page...*

## MCG_SC field descriptions (continued)

| Field | Description |
|---|---|
| 6<br>ATMS | Automatic Trim Machine Select<br><br>Selects the IRCS clock for Auto Trim Test.<br><br>0 32 kHz Internal Reference Clock selected.<br>1 4 MHz Internal Reference Clock selected. |
| 5<br>ATMF | Automatic Trim Machine Fail Flag<br><br>Fail flag for the Automatic Trim Machine (ATM). This bit asserts when the Automatic Trim Machine is enabled, ATME=1, and a write to the C1, C3, C4, and SC registers is detected or the MCG enters into any Stop mode. A write to ATMF clears the flag.<br><br>0 Automatic Trim Machine completed normally.<br>1 Automatic Trim Machine failed. |
| 4<br>FLTPRSRV | FLL Filter Preserve Enable<br><br>This bit will prevent the FLL filter values from resetting allowing the FLL output frequency to remain the same during clock mode changes where the FLL/DCO output is still valid. (Note: This requires that the FLL reference frequency to remain the same as what it was prior to the new clock mode switch. Otherwise FLL filter and frequency values will change.)<br><br>0 FLL filter and FLL frequency will reset on changes to currect clock mode.<br>1 Fll filter and FLL frequency retain their previous values during new clock mode change. |
| 3–1<br>FCRDIV | Fast Clock Internal Reference Divider<br><br>Selects the amount to divide down the fast internal reference clock. The resulting frequency will be in the range 31.25 kHz to 4 MHz (Note: Changing the divider when the Fast IRC is enabled is not supported).<br><br>000 Divide Factor is 1<br>001 Divide Factor is 2.<br>010 Divide Factor is 4.<br>011 Divide Factor is 8.<br>100 Divide Factor is 16<br>101 Divide Factor is 32<br>110 Divide Factor is 64<br>111 Divide Factor is 128. |
| 0<br>LOCS0 | OSC0 Loss of Clock Status<br><br>The LOCS0 indicates when a loss of OSC0 reference clock has occurred. The LOCS0 bit only has an effect when CME0 is set. This bit is cleared by writing a logic 1 to it when set.<br><br>0 Loss of OSC0 has not occurred.<br>1 Loss of OSC0 has occurred. |

## 25.3.9 MCG Auto Trim Compare Value High Register (MCG_ATCVH)

Address: 4006_4000h base + Ah offset = 4006_400Ah

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | ATCVH | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### MCG_ATCVH field descriptions

| Field | Description |
|---|---|
| 7–0<br>ATCVH | ATM Compare Value High<br><br>Values are used by Auto Trim Machine to compare and adjust Internal Reference trim values during ATM SAR conversion. |

## 25.3.10 MCG Auto Trim Compare Value Low Register (MCG_ATCVL)

Address: 4006_4000h base + Bh offset = 4006_400Bh

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | ATCVL | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### MCG_ATCVL field descriptions

| Field | Description |
|---|---|
| 7–0<br>ATCVL | ATM Compare Value Low<br><br>Values are used by Auto Trim Machine to compare and adjust Internal Reference trim values during ATM SAR conversion. |

## 25.3.11 MCG Control 7 Register (MCG_C7)

Address: 4006_4000h base + Ch offset = 4006_400Ch

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | | 0 | | | | | OSCSEL |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### MCG_C7 field descriptions

| Field | Description |
|---|---|
| 7–6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5–1<br>Reserved | Reserved<br><br>This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>OSCSEL | MCG OSC Clock Select<br><br>Selects the MCG FLL external reference clock<br><br>0 Selects System Oscillator (OSCCLK).<br>1 Selects 32 kHz RTC Oscillator. |

## 25.3.12 MCG Control 8 Register (MCG_C8)

Address: 4006_4000h base + Dh offset = 4006_400Dh

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | LOCRE1 | LOLRE | CME1 | 0 | | | | LOCS1 |
| Write | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### MCG_C8 field descriptions

| Field | Description |
|---|---|
| 7<br>LOCRE1 | Loss of Clock Reset Enable<br><br>Determines if a interrupt or a reset request is made following a loss of RTC external reference clock. The LOCRE1 only has an affect when CME1 is set.<br><br>0 Interrupt request is generated on a loss of RTC external reference clock.<br>1 Generate a reset request on a loss of RTC external reference clock |
| 6<br>LOLRE | PLL Loss of Lock Reset Enable<br><br>Determines if a interrupt or a reset request is made following a PLL loss of lock.<br><br>0 Interrupt request is generated on a PLL loss of lock indication. The PLL loss of lock interrupt enable bit must also be set to generate the interrupt request.<br>1 Generate a reset request on a PLL loss of lock indication. |
| 5<br>CME1 | Clock Monitor Enable1<br><br>Enables the loss of clock monitoring circuit for the output of the RTC external reference clock. The LOCRE1 bit will determine whether an interrupt or a reset request is generated following a loss of RTC clock indication. The CME1 bit should be set to a logic 1 when the MCG is in an operational mode that uses the RTC as its external reference clock or if the RTC is operational. CME1 bit must be set to a logic 0 before the MCG enters any Stop mode. Otherwise, a reset request may occur when in Stop mode. CME1 should also be set to a logic 0 before entering VLPR or VLPW power modes. |

*Table continues on the next page...*

**MCG_C8 field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 External clock monitor is disabled for RTC clock.<br>1 External clock monitor is enabled for RTC clock. |
| 4–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>LOCS1 | RTC Loss of Clock Status<br><br>This bit indicates when a loss of clock has occurred. This bit is cleared by writing a logic 1 to it when set.<br><br>0 Loss of RTC has not occur.<br>1 Loss of RTC has occur |

# 25.4 Functional Description

## 25.4.1 MCG mode state diagram

The nine states of the MCG are shown in the following figure and are described in Table 25-16. The arrows indicate the permitted MCG mode transitions.

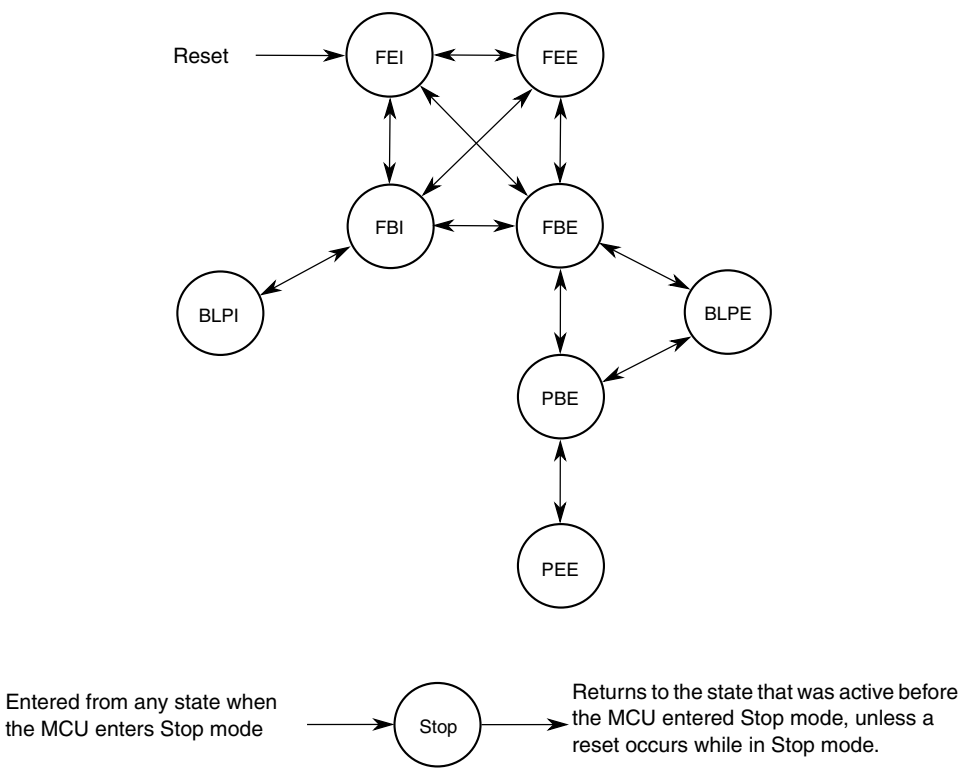



**Figure 25-14. MCG mode state diagram**

**NOTE**

- During exits from LLS or VLPS when the MCG is in PEE mode, the MCG will reset to PBE clock mode and the C1[CLKS] and S[CLKST] will automatically be set to 2'b10.
- If entering Normal Stop mode when the MCG is in PEE mode with PLLSTEN=0, the MCG will reset to PBE clock mode and C1[CLKS] and S[CLKST] will automatically be set to 2'b10.

#### 25.4.1.1 MCG modes of operation

The MCG operates in one of the following modes.

**Note**

The MCG restricts transitions between modes. For the permitted transitions, see Figure 25-14.

**Table 25-16. MCG modes of operation**

| Mode | Description |
|---|---|
| FLL Engaged Internal (FEI) | FLL engaged internal (FEI) is the default mode of operation and is entered when all the following condtions occur:<br>• C1[CLKS] bits are written to 00<br>• C1[IREFS] bit is written to 1<br>• C6[PLLS] bit is written to 0<br><br>In FEI mode, MCGOUTCLK is derived from the FLL clock (DCOCLK) that is controlled by the 32 kHz Internal Reference Clock (IRC). The FLL loop will lock the DCO frequency to the FLL factor, as selected by C4[DRST_DRS] and C4[DMX32] bits, times the internal reference frequency. See the C4[DMX32] bit description for more details. In FEI mode, the PLL is disabled in a low-power state unless C5[PLLCLKEN] is set . |

*Table continues on the next page...*

## Table 25-16. MCG modes of operation (continued)

| Mode | Description |
|---|---|
| FLL Engaged External (FEE) | FLL engaged external (FEE) mode is entered when all the following conditions occur:<br>• C1[CLKS] bits are written to 00<br>• C1[IREFS] bit is written to 0<br>• C1[FRDIV] must be written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz<br>• C6[PLLS] bit is written to 0<br><br>In FEE mode, MCGOUTCLK is derived from the FLL clock (DCOCLK) that is controlled by the external reference clock. The FLL loop will lock the DCO frequency to the FLL factor, as selected by C4[DRST_DRS] and C4[DMX32] bits, times the external reference frequency, as specified by C1[FRDIV] and C2[RANGE0]. See the C4[DMX32] bit description for more details. In FEE mode, the PLL is disabled in a low-power state unless C5[PLLCLKEN] is set . |
| FLL Bypassed Internal (FBI) | FLL bypassed internal (FBI) mode is entered when all the following conditions occur:<br>• C1[CLKS] bits are written to 01<br>• C1[IREFS] bit is written to 1<br>• C6[PLLS] is written to 0<br>• C2[LP] is written to 0<br><br>In FBI mode, the MCGOUTCLK is derived either from the slow (32 kHz IRC) or fast (4 MHz IRC) internal reference clock, as selected by the C2[IRCS] bit. The FLL is operational but its output is not used. This mode is useful to allow the FLL to acquire its target frequency while the MCGOUTCLK is driven from the C2[IRCS] selected internal reference clock. The FLL clock (DCOCLK) is controlled by the slow internal reference clock, and the DCO clock frequency locks to a multiplication factor, as selected by C4[DRST_DRS] and C4[DMX32] bits, times the internal reference frequency. See the C4[DMX32] bit description for more details. In FBI mode, the PLL is disabled in a low-power state unless C5[PLLCLKEN] is set . |
| FLL Bypassed External (FBE) | FLL bypassed external (FBE) mode is entered when all the following conditions occur:<br>• C1[CLKS] bits are written to 10<br>• C1[IREFS] bit is written to 0<br>• C1[FRDIV] must be written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz.<br>• C6[PLLS] bit is written to 0<br>• C2[LP] is written to 0<br><br>In FBE mode, the MCGOUTCLK is derived from the OSCSEL external reference clock. The FLL is operational but its output is not used. This mode is useful to allow the FLL to acquire its target frequency while the MCGOUTCLK is driven from the external reference clock. The FLL clock (DCOCLK) is controlled by the external reference clock, and the DCO clock frequency locks to a multiplication factor, as selected by C4[DRST_DRS] and C4[DMX32] bits, times the divided external reference frequency. See the C4[DMX32] bit description for more details. In FBI mode, the PLL is disabled in a low-power state unless C5[PLLCLKEN] is set . |

*Table continues on the next page...*

## Table 25-16. MCG modes of operation (continued)

| Mode | Description |
|---|---|
| PLL Engaged External (PEE) | PLL Engaged External (PEE) mode is entered when all the following conditions occur:<br>• C1[CLKS] bits are written to 00<br>• C1[IREFS] bit is written to 0<br>• C6[PLLS] bit is written to 1<br><br>In PEE mode, the MCGOUTCLK is derived from the output of PLL which is controlled by a external reference clock. The PLL clock frequency locks to a multiplication factor, as specified by its corresponding VDIV, times the selected PLL reference frequency, as specified by its corresponding PRDIV. The PLL's programmable reference divider must be configured to produce a valid PLL reference clock. The FLL is disabled in a low-power state. |
| PLL Bypassed External (PBE) | PLL Bypassed External (PBE) mode is entered when all the following conditions occur:<br>• C1[CLKS] bits are written to 10<br>• C1[IREFS] bit is written to 0<br>• C6[PLLS] bit is written to 1<br>• C2[LP] bit is written to 0<br><br>In PBE mode, MCGOUTCLK is derived from the OSCSEL external reference clock; the PLL is operational, but its output clock is not used. This mode is useful to allow the PLL to acquire its target frequency while MCGOUTCLK is driven from the external reference clock. The PLL clock frequency locks to a multiplication factor, as specified by its [VDIV], times the PLL reference frequency, as specified by its [PRDIV]. In preparation for transition to PEE, the PLL's programmable reference divider must be configured to produce a valid PLL reference clock. The FLL is disabled in a low-power state. |
| Bypassed Low Power Internal (BLPI)[1] | Bypassed Low Power Internal (BLPI) mode is entered when all the following conditions occur:<br>• C1[CLKS] bits are written to 01<br>• C1[IREFS] bit is written to 1<br>• C6[PLLS] bit is written to 0<br>• C2[LP] bit is written to 1<br><br>In BLPI mode, MCGOUTCLK is derived from the internal reference clock. The FLL is disabled and PLL is disabled even if the C5[PLLCLKEN0] is set to 1. |
| Bypassed Low Power External (BLPE) | Bypassed Low Power External (BLPE) mode is entered when all the following conditions occur:<br>• C1[CLKS] bits are written to 10<br>• C1[IREFS] bit is written to 0<br>• C2[LP] bit is written to 1<br><br>In BLPE mode, MCGOUTCLK is derived from the OSCSEL external reference clock. The FLL is disabled and PLL is disabled even if the C5[PLLCLKEN0] is set to 1. |

*Table continues on the next page...*

**Table 25-16. MCG modes of operation (continued)**

| Mode | Description |
|---|---|
| Stop | Entered whenever the MCU enters a Stop state. The power modes are chip specific. For power mode assignments, see the chapter that describes how modules are configured and MCG behavior during Stop recovery. Entering Stop mode, the FLL is disabled, and all MCG clock signals are static except in the following case:<br>MCGPLL0CLK is active in Normal Stop mode when PLLSTEN0=1<br>MCGIRCLK is active in Normal Stop mode when all the following conditions become true:<br>• C1[IRCLKEN] = 1<br>• C1[IREFSTEN] = 1<br>**NOTE:** • In VLPS Stop Mode, the MCGIRCLK can be programmed to stay enabled and continue running if C1[IRCLKEN] = 1, C1[IREFSTEN]=1, and Fast IRC clock is selected (C2[IRCS] = 1)<br>**NOTE:** • When entering Low Power Stop modes (LLS or VLPS) from PEE mode, on exit the MCG clock mode is forced to PBE clock mode. C1[CLKS] and S[CLKST] will be configured to 2'b10if entering from PEE mode or to 2'b01 if entering from PEI mode, C5[PLLSTEN0] will be force to 1'b0 and S[LOCK0] bit will be cleared without setting S[LOLS0].<br>• When entering Normal Stop mode from PEE mode and if C5[PLLSTEN0]=0, on exit the MCG clock mode is forced to PBE mode, the C1[CLKS] and S[CLKST] will be configured to 2'b10 and S[LOCK0] bit will clear without setting S[LOLS0]. If C5[PLLSTEN0]=1, the S[LOCK0] bit will not get cleared and on exit the MCG will continue to run in PEE mode. |

1. If entering VLPR mode, MCG has to be configured and enter BLPE mode or BLPI mode with the Fast IRC clock selected (C2[IRCS]=1). After it enters VLPR mode, writes to any of the MCG control registers that can cause an MCG clock mode switch to a non low power clock mode must be avoided.

**NOTE**

For the chip-specific modes of operation, see the power management chapter of this MCU.

### 25.4.1.2 MCG mode switching

The C1[IREFS] bit can be changed at any time, but the actual switch to the newly selected reference clocks is shown by the S[IREFST] bit. When switching between engaged internal and engaged external modes, the FLL will begin locking again after the switch is completed.

The C1[CLKS] bits can also be changed at any time, but the actual switch to the newly selected clock is shown by the S[CLKST] bits. If the newly selected clock is not available, the previous clock will remain selected.

The C4[DRST_DRS] write bits can be changed at any time except when C2[LP] bit is 1. If the C4[DRST_DRS] write bits are changed while in FLL engaged internal (FEI) or FLL engaged external (FEE), the MCGOUTCLK will switch to the new selected DCO range within three clocks of the selected DCO clock. After switching to the new DCO, the FLL remains unlocked for several reference cycles. DCO startup time is equal to the FLL acquisition time. After the selected DCO startup time is over, the FLL is locked. The completion of the switch is shown by the C4[DRST_DRS] read bits.

## 25.4.2 Low Power Bit Usage

The C2[LP] bit is provided to allow the FLL or PLL to be disabled and thus conserve power when these systems are not being used. The C4[DRST_DRS] can not be written while C2[LP] bit is 1. However, in some applications, it may be desirable to enable the FLL or PLL and allow it to lock for maximum accuracy before switching to an engaged mode. Do this by writing C2[LP] to 0.

## 25.4.3 MCG Internal Reference Clocks

This module supports two internal reference clocks with nominal frequencies of 32 kHz (slow IRC) and 4 MHz (fast IRC). The fast IRC frequency can be divided down by programming of the FCRDIV to produce a frequency range of 32 kHz to 4 MHz.

### 25.4.3.1 MCG Internal Reference Clock

The MCG Internal Reference Clock (MCGIRCLK) provides a clock source for other on-chip peripherals and is enabled when C1[IRCLKEN]=1. When enabled, MCGIRCLK is driven by either the fast internal reference clock (4 MHz IRC which can be divided down by the FRDIV factors) or the slow internal reference clock (32 kHz IRC). The IRCS clock frequency can be re-targeted by trimming the period of its IRCS selected internal reference clock. This can be done by writing a new trim value to the C3[SCTRIM]:C4[SCFTRIM] bits when the slow IRC clock is selected or by writing a new trim value to the C4[FCTRIM] bits when the fast IRC clock is selected. The internal reference clock period is proportional to the trim value written.
C3[SCTRIM]:C4[SCFTRIM] (if C2[IRCS]=0) and C4[FCTRIM] (if C2[IRCS]=1) bits affect the MCGOUTCLK frequency if the MCG is in FBI or BLPI modes.
C3[SCTRIM]:C4[SCFTRIM] (if C2[IRCS]=0) bits also affect the MCGOUTCLK frequency if the MCG is in FEI mode.

Additionally, this clock can be enabled in Stop mode by setting C1[IRCLKEN] and C1[IREFSTEN], otherwise this clock is disabled in Stop mode.

### 25.4.4 External Reference Clock

The MCG module can support an external reference clock in all modes. See the device datasheet for external reference frequency range. When C1[IREFS] is set, the external reference clock will not be used by the FLL or PLL. In these modes, the frequency can be equal to the maximum frequency the chip-level timing specifications will support.

If any of the CME bits are asserted the slow internal reference clock is enabled along with the enabled external clock monitor. For the case when C6[CME0]=1, a loss of clock is detected if the OSC0 external reference falls below a minimum frequency ($f_{loc\_high}$ or $f_{loc\_low}$ depending on C2[RANGE0]). For the case when C8[CME1]=1, a loss of clock is detected if the RTC external reference falls below a minimum frequency ($f_{loc\_low}$).

**NOTE**

All clock monitors must be disabled before entering these low-power modes: Stop, VLPS, VLPR, VLPW, LLS, and VLLSx.

Upon detect of a loss of clock event, the MCU generates a system reset if the respective LOCRE bit is set. Otherwise the MCG sets the respective LOCS bit and the MCG generates a LOCS interrupt request. In the case where a OSC loss of clock is detected, the PLL LOCK status bit is cleared.

### 25.4.5 MCG Fixed frequency clock

The MCG Fixed Frequency Clock (MCGFFCLK) provides a fixed frequency clock source for other on-chip peripherals; see the block diagram. This clock is driven by either the slow clock from the internal reference clock generator or the external reference clock from the Crystal Oscillator, divided by the FLL reference clock divider. The source of MCGFFCLK is selected by C1[IREFS].

This clock is synchronized to the peripheral bus clock and is valid only when its frequency is not more than 1/8 of the MCGOUTCLK frequency. When it is not valid, it is disabled and held high. The MCGFFCLK is not available when the MCG is in BLPI mode. This clock is also disabled in Stop mode. The FLL reference clock must be set within the valid frequency range for the MCGFFCLK.

### 25.4.6 MCG PLL clock

The MCG PLL Clock (MCGPLLCLK) is available depending on the device's configuration of the MCG module. For more details, see the clock distribution chapter of this MCU. The MCGPLLCLK is prevented from coming out of the MCG until it is enabled and S[LOCK0] is set.

### 25.4.7 MCG Auto TRIM (ATM)

The MCG Auto Trim (ATM) is a MCG feature that when enabled, it configures the MCG hardware to automatically trim the MCG Internal Reference Clocks using an external clock as a reference. The selection between which MCG IRC clock gets tested and enabled is controlled by the ATC[ATMS] control bit (ATC[ATMS]=0 selects the 32 kHz IRC and ATC[ATMS]=1 selects the 4 MHz IRC). If 4 MHz IRC is selected for the ATM, a divide by 128 is enabled to divide down the 4 MHz IRC to a range of 31.250 kHz.

When MCG ATM is enabled by writing ATC[ATME] bit to 1, The ATM machine will start auto trimming the selected IRC clock. During the autotrim process, ATC[ATME] will remain asserted and will deassert after ATM is completed or an abort occurs. The MCG ATM is aborted if a write to any of the following control registers is detected : C1, C3, C4, or ATC or if Stop mode is entered. If an abort occurs, ATC[ATMF] fail flag is asserted.

The ATM machine uses the bus clock as the external reference clock to perform the IRC auto-trim. Therefore, it is required that the MCG is configured in a clock mode where the reference clock used to generate the system clock is the external reference clock such as FBE clock mode. The MCG must not be configured in a clock mode where selected IRC ATM clock is used to generate the system clock. The bus clock is also required to be running with in the range of 8–16 MHz.

To perform the ATM on the selected IRC, the ATM machine uses the successive approximation technique to adjust the IRC trim bits to generate the desired IRC trimmed frequency. The ATM SARs each of the ATM IRC trim bits starting with the MSB. For each trim bit test, the ATM uses a pulse that is generated by the ATM selected IRC clock to enable a counter that counts number of ATM external clocks. At end of each trim bit, the ATM external counter value is compared to the ATCV[15:0] register value. Based on the comparison result, the ATM trim bit under test will get cleared or stay asserted. This is done until all trim bits have been tested by ATM SAR machine.

Before the ATM can be enabled, the ATM expected count needs to be derived and stored into the ATCV register. The ATCV expected count is derived based on the required target Internal Reference Clock (IRC) frequency, and the frequency of the external reference clock using the following formula:

$$\text{ATCV Expected Count Value} = 21*(Fe/Fr)$$

- Fr = Target Internal Reference Clock (IRC) Trimmed Frequency
- Fe = External Clock Frequency

If the auto trim is being performed on the 4 MHz IRC, the calculated expected count value must be multiplied by 128 before storing it in the ATCV register. Therefore, the ATCV Expected Count Value for trimming the 4 MHz IRC is calculated using the following formula.

$$\text{Expected Count Value} = (Fe/Fr)*21*(128)$$

## 25.5 Initialization / Application information

This section describes how to initialize and configure the MCG module in an application. The following sections include examples on how to initialize the MCG and properly switch between the various available modes.

### 25.5.1 MCG module initialization sequence

The MCG comes out of reset configured for FEI mode. The internal reference will stabilize in $t_{irefsts}$ microseconds before the FLL can acquire lock. As soon as the internal reference is stable, the FLL will acquire lock in $t_{fll\_acquire}$ milliseconds.

#### 25.5.1.1 Initializing the MCG

Because the MCG comes out of reset in FEI mode, the only MCG modes that can be directly switched to upon reset are FEE, FBE, and FBI modes (see Figure 25-14). Reaching any of the other modes requires first configuring the MCG for one of these three intermediate modes. Care must be taken to check relevant status bits in the MCG status register reflecting all configuration changes within each mode.

To change from FEI mode to FEE or FBE modes, follow this procedure:

1. Enable the external clock source by setting the appropriate bits in C2 register.

2. Write to C1 register to select the clock mode.
   - If entering FEE mode, set C1[FRDIV] appropriately, clear the C1[IREFS] bit to switch to the external reference, and leave the C1[CLKS] bits at 2'b00 so that the output of the FLL is selected as the system clock source.
   - If entering FBE, clear the C1[IREFS] bit to switch to the external reference and change the C1[CLKS] bits to 2'b10 so that the external reference clock is selected as the system clock source. The C1[FRDIV] bits should also be set appropriately here according to the external reference frequency to keep the FLL reference clock in the range of 31.25 kHz to 39.0625 kHz. Although the FLL is bypassed, it is still on in FBE mode.
   - The internal reference can optionally be kept running by setting the C1[IRCLKEN] bit. This is useful if the application will switch back and forth between internal and external modes. For minimum power consumption, leave the internal reference disabled while in an external clock mode.
3. Once the proper configuration bits have been set, wait for the affected bits in the MCG status register to be changed appropriately, reflecting that the MCG has moved into the proper mode.
   - If the MCG is in FEE, FBE, PEE, PBE, or BLPE mode, and C2[EREFS0] was also set in step 1, wait here for S[OSCINIT0] bit to become set indicating that the external clock source has finished its initialization cycles and stabilized.
   - If in FEE mode, check to make sure the S[IREFST] bit is cleared before moving on.
   - If in FBE mode, check to make sure the S[IREFST] bit is cleared and S[CLKST] bits have changed to 2'b10 indicating the external reference clock has been appropriately selected. Although the FLL is bypassed, it is still on in FBE mode.
4. Write to the C4 register to determine the DCO output (MCGFLLCLK) frequency range.
   - By default, with C4[DMX32] cleared to 0, the FLL multiplier for the DCO output is 640. For greater flexibility, if a mid-low-range FLL multiplier of 1280 is desired instead, set C4[DRST_DRS] bits to 2'b01 for a DCO output frequency of 40 MHz. If a mid high-range FLL multiplier of 1920 is desired instead, set the C4[DRST_DRS] bits to 2'b10 for a DCO output frequency of 60 MHz. If a high-range FLL multiplier of 2560 is desired instead, set the C4[DRST_DRS] bits to 2'b11 for a DCO output frequency of 80 MHz.

- When using a 32.768 kHz external reference, if the maximum low-range DCO frequency that can be achieved with a 32.768 kHz reference is desired, set C4[DRST_DRS] bits to 2'b00 and set C4[DMX32] bit to 1. The resulting DCO output (MCGOUTCLK) frequency with the new multiplier of 732 will be 24 MHz.
- When using a 32.768 kHz external reference, if the maximum mid-range DCO frequency that can be achieved with a 32.768 kHz reference is desired, set C4[DRST_DRS] bits to 2'b01 and set C4[DMX32] bit to 1. The resulting DCO output (MCGOUTCLK) frequency with the new multiplier of 1464 will be 48 MHz.
- When using a 32.768 kHz external reference, if the maximum mid high-range DCO frequency that can be achieved with a 32.768 kHz reference is desired, set C4[DRST_DRS] bits to 2'b10 and set C4[DMX32] bit to 1. The resulting DCO output (MCGOUTCLK) frequency with the new multiplier of 2197 will be 72 MHz.
- When using a 32.768 kHz external reference, if the maximum high-range DCO frequency that can be achieved with a 32.768 kHz reference is desired, set C4[DRST_DRS] bits to 2'b11 and set C4[DMX32] bit to 1. The resulting DCO output (MCGOUTCLK) frequency with the new multiplier of 2929 will be 96 MHz.

5. Wait for the FLL lock time to guarantee FLL is running at new C4[DRST_DRS] and C4[DMX32] programmed frequency.

To change from FEI clock mode to FBI clock mode, follow this procedure:

1. Change C1[CLKS] bits in C1 register to 2'b01 so that the internal reference clock is selected as the system clock source.
2. Wait for S[CLKST] bits in the MCG status register to change to 2'b01, indicating that the internal reference clock has been appropriately selected.
3. Write to the C2 register to determine the IRCS output (IRCSCLK) frequency range.
   - By default, with C2[IRCS] cleared to 0, the IRCS selected output clock is the slow internal reference clock (32 kHz IRC). If the faster IRC is desired, set C2[IRCS] bit to 1 for a IRCS clock derived from the 4 MHz IRC source.

## 25.5.2 Using a 32.768 kHz reference

In FEE and FBE modes, if using a 32.768 kHz external reference, at the default FLL multiplication factor of 640, the DCO output (MCGFLLCLK) frequency is 20.97 MHz at low-range. If C4[DRST_DRS] bits are set to 2'b01, the multiplication factor is doubled to 1280, and the resulting DCO output frequency is 41.94 MHz at mid-low-range. If C4[DRST_DRS] bits are set to 2'b10, the multiplication factor is set to 1920, and the resulting DCO output frequency is 62.91 MHz at mid high-range. If C4[DRST_DRS] bits are set to 2'b11, the multiplication factor is set to 2560, and the resulting DCO output frequency is 83.89 MHz at high-range.

In FBI and FEI modes, setting C4[DMX32] bit is not recommended. If the internal reference is trimmed to a frequency above 32.768 kHz, the greater FLL multiplication factor could potentially push the microcontroller system clock out of specification and damage the part.

## 25.5.3 MCG mode switching

When switching between operational modes of the MCG, certain configuration bits must be changed in order to properly move from one mode to another. Each time any of these bits are changed (C6[PLLS], C1[IREFS], C1[CLKS], C2[IRCS], or C2[EREFS0]), the corresponding bits in the MCG status register (PLLST, IREFST, CLKST, IRCST, or OSCINIT) must be checked before moving on in the application software.

Additionally, care must be taken to ensure that the reference clock divider (C1[FRDIV] and C5[PRDIV0]) is set properly for the mode being switched to. For instance, in PEE mode, if using a 4 MHz crystal, C5[PRDIV0] must be set to 5'b000 (divide-by-1) or 5'b001 (divide-by-2) to divide the external reference down to the required frequency between 2 and 4 MHz.

In FBE, FEE, FBI, and FEI modes, at any time, the application can switch the FLL multiplication factor between 640, 1280, 1920, and 2560 with C4[DRST_DRS] bits. Writes to C4[DRST_DRS] bits will be ignored if C2[LP]=1.

The table below shows MCGOUTCLK frequency calculations using C1[FRDIV], C5[PRDIV0], and C6[VDIV0] settings for each clock mode.

**Table 25-17. MCGOUTCLK Frequency Calculation Options**

| Clock Mode | $f_{MCGOUTCLK}$[1] | Note |
|---|---|---|
| FEI (FLL engaged internal) | ($f_{int}$ * F) | Typical $f_{MCGOUTCLK}$ = 21 MHz immediately after reset. |
| FEE (FLL engaged external) | ($f_{ext}$ / FLL_R) *F | $f_{ext}$ / FLL_R must be in the range of 31.25 kHz to 39.0625 kHz |

*Table continues on the next page...*

**Table 25-17. MCGOUTCLK Frequency Calculation Options (continued)**

| Clock Mode | $f_{MCGOUTCLK}$[1] | Note |
|---|---|---|
| FBE (FLL bypassed external) | OSCCLK | OSCCLK / FLL_R must be in the range of 31.25 kHz to 39.0625 kHz |
| FBI (FLL bypassed internal) | MCGIRCLK | Selectable between slow and fast IRC |
| PEE (PLL engaged external) | (OSCCLK / PLL_R) * M | OSCCLK / PLL_R must be in the range of 2 – 4 MHz |
| PBE (PLL bypassed external) | OSCCLK | OSCCLK / PLL_R must be in the range of 2 – 4 MHz |
| BLPI (Bypassed low power internal) | MCGIRCLK | Selectable between slow and fast IRC |
| BLPE (Bypassed low power external) | OSCCLK | |

1. FLL_R is the reference divider selected by the C1[FRDIV] bits, PLL_R is the reference divider selected by C5[PRDIV0] bits, F is the FLL factor selected by C4[DRST_DRS] and C4[DMX32] bits, and M is the multiplier selected by C6[VDIV0] bits.

This section will include three mode switching examples using an 4 MHz external crystal. If using an external clock source less than 2 MHz, the MCG must not be configured for any of the PLL modes (PEE and PBE).

### 25.5.3.1 Example 1: Moving from FEI to PEE mode: External Crystal = 4 MHz, MCGOUTCLK frequency = 48 MHz

In this example, the MCG will move through the proper operational modes from FEI to PEE to achieve 48 MHz MCGOUTCLK frequency from 4 MHz external crystal reference. First, the code sequence will be described. Then there is a flowchart that illustrates the sequence.

1. First, FEI must transition to FBE mode:
   a. C2 = 0x2C
      - C2[RANGE0] set to 2'b01 because the frequency of 4 MHz is within the high frequency range.
      - C2[HGO0] set to 1 to configure the crystal oscillator for high gain operation.
      - C2[EREFS0] set to 1, because a crystal is being used.
   b. C1 = 0x90
      - C1[CLKS] set to 2'b10 to select external reference clock as system clock source

- C1[FRDIV] set to 3'b010, or divide-by-128 because 4 MHz / 128 = 31.25 kHz which is in the 31.25 kHz to 39.0625 kHz range required by the FLL
- C1[IREFS] cleared to 0, selecting the external reference clock and enabling the external oscillator.

c. Loop until S[OSCINIT0] is 1, indicating the crystal selected by C2[EREFS0] has been initialized.

d. Loop until S[IREFST] is 0, indicating the external reference is the current source for the reference clock.

e. Loop until S[CLKST] is 2'b10, indicating that the external reference clock is selected to feed MCGOUTCLK.

2. Then configure C5[PRDIV0] to generate correct PLL reference frequency.

a. C5 = 0x01

- C5[PRDIV0] set to 5'b001, or divide-by-2 resulting in a pll reference frequency of 4 MHz/2 = 2 MHz.

3. Then, FBE must transition either directly to PBE mode or first through BLPE mode and then to PBE mode:

a. BLPE: If a transition through BLPE mode is desired, first set C2[LP] to 1.

b. BLPE/PBE: C6 = 0x40

- C6[PLLS] set to 1, selects the PLL. At this time, with a C1[PRDIV] value of 2'b001, the PLL reference divider is 2 (see PLL External Reference Divide Factor table), resulting in a reference frequency of 4 MHz/ 2 = 2 MHz. In BLPE mode, changing the C6[PLLS] bit only prepares the MCG for PLL usage in PBE mode.
- C6[VDIV0] set to 5'b00000, or multiply-by-24 because 2 MHz reference * 24 = 48 MHz. In BLPE mode, the configuration of the VDIV bits does not matter because the PLL is disabled. Changing them only sets up the multiply value for PLL usage in PBE mode.

c. BLPE: If transitioning through BLPE mode, clear C2[LP] to 0 here to switch to PBE mode.

d. PBE: Loop until S[PLLST] is set, indicating that the current source for the PLLS clock is the PLL.

e. PBE: Then loop until S[LOCK0] is set, indicating that the PLL has acquired lock.

4. Lastly, PBE mode transitions into PEE mode:
    a. C1 = 0x10
        - C1[CLKS] set to 2'b00 to select the output of the PLL as the system clock source.
    b. Loop until S[CLKST] are 2'b11, indicating that the PLL output is selected to feed MCGOUTCLK in the current clock mode.
        - Now, with PRDIV0 of divide-by-2, and C6[VDIV0] of multiply-by-24, MCGOUTCLK = [(4 MHz / 2) * 24] = 48 MHz.

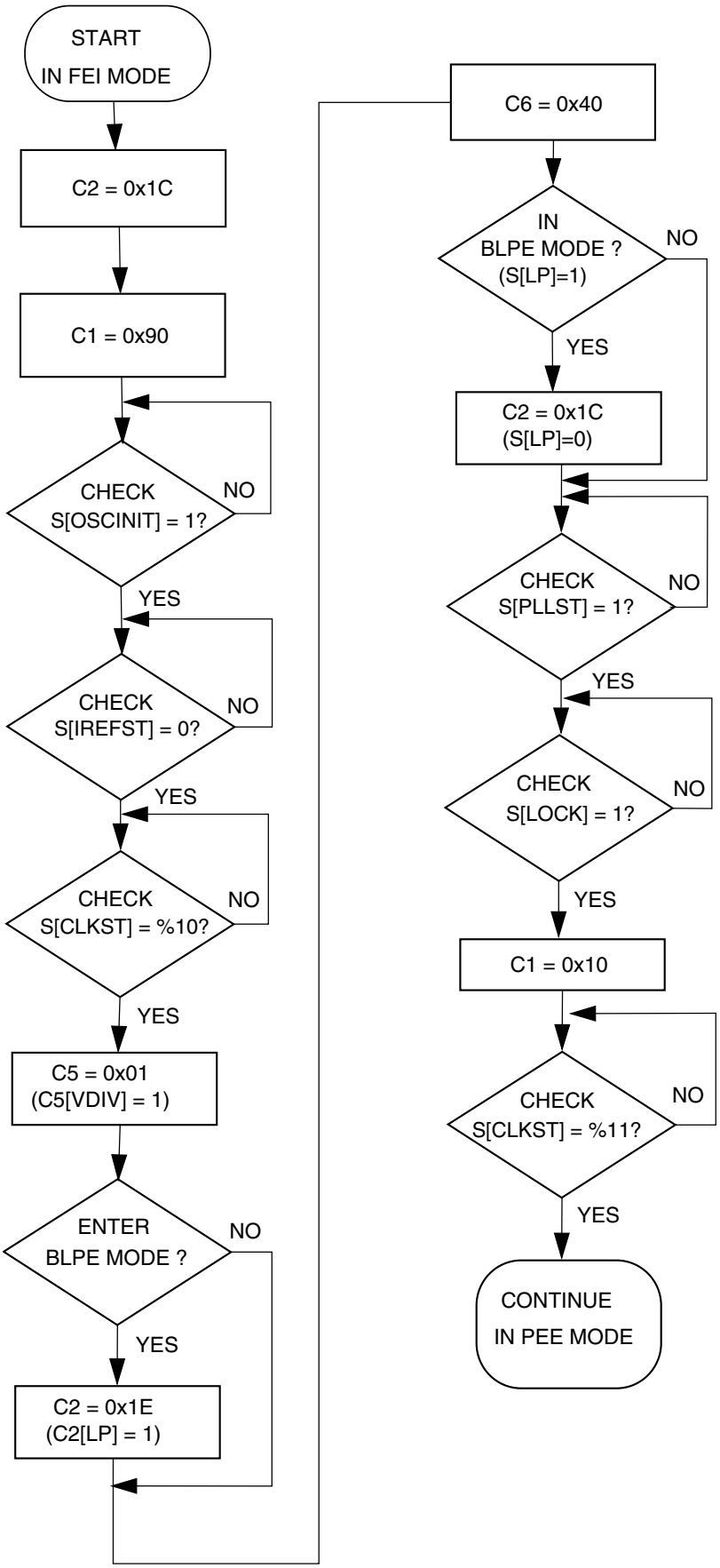



**Figure 25-15. Flowchart of FEI to PEE mode transition using an 4 MHz crystal**

### 25.5.3.2 Example 2: Moving from PEE to BLPI mode: MCGOUTCLK frequency =32 kHz

In this example, the MCG will move through the proper operational modes from PEE mode with a 4 MHz crystal configured for a 48 MHz MCGOUTCLK frequency (see previous example) to BLPI mode with a 32 kHz MCGOUTCLK frequency. First, the code sequence will be described. Then there is a flowchart that illustrates the sequence.

1. First, PEE must transition to PBE mode:
   a. C1 = 0x90
      - C1[CLKS] set to 2'b10 to switch the system clock source to the external reference clock.
   b. Loop until S[CLKST] are 2'b10, indicating that the external reference clock is selected to feed MCGOUTCLK.
2. Then, PBE must transition either directly to FBE mode or first through BLPE mode and then to FBE mode:
   a. BLPE: If a transition through BLPE mode is desired, first set C2[LP] to 1.
   b. BLPE/FBE: C6 = 0x00
      - C6[PLLS] clear to 0 to select the FLL. At this time, with C1[FRDIV] value of 3'b010, the FLL divider is set to 128, resulting in a reference frequency of 4 MHz / 128 = 31.25 kHz. If C1[FRDIV] was not previously set to 3'b010 (necessary to achieve required 31.25–39.06 kHz FLL reference frequency with an 4 MHz external source frequency), it must be changed prior to clearing C6[PLLS] bit. In BLPE mode,changing this bit only prepares the MCG for FLL usage in FBE mode. With C6[PLLS] = 0, the C6[VDIV0] value does not matter.
   c. BLPE: If transitioning through BLPE mode, clear C2[LP] to 0 here to switch to FBE mode.
   d. FBE: Loop until S[PLLST] is cleared, indicating that the current source for the PLLS clock is the FLL.
3. Next, FBE mode transitions into FBI mode:
   a. C1 = 0x54
      - C1[CLKS] set to 2'b01 to switch the system clock to the internal reference clock.

- C1[IREFS] set to 1 to select the internal reference clock as the reference clock source.
- C1[FRDIV] remain unchanged because the reference divider does not affect the internal reference.

b. Loop until S[IREFST] is 1, indicating the internal reference clock has been selected as the reference clock source.

c. Loop until S[CLKST] are 2'b01, indicating that the internal reference clock is selected to feed MCGOUTCLK.

4. Lastly, FBI transitions into BLPI mode.

a. C2 = 0x02

- C2[LP] is 1
- C2[RANGE0], C2[HGO0], C2[EREFS0], C1[IRCLKEN], and C1[IREFSTEN] bits are ignored when the C1[IREFS] bit is set. They can remain set, or be cleared at this point.

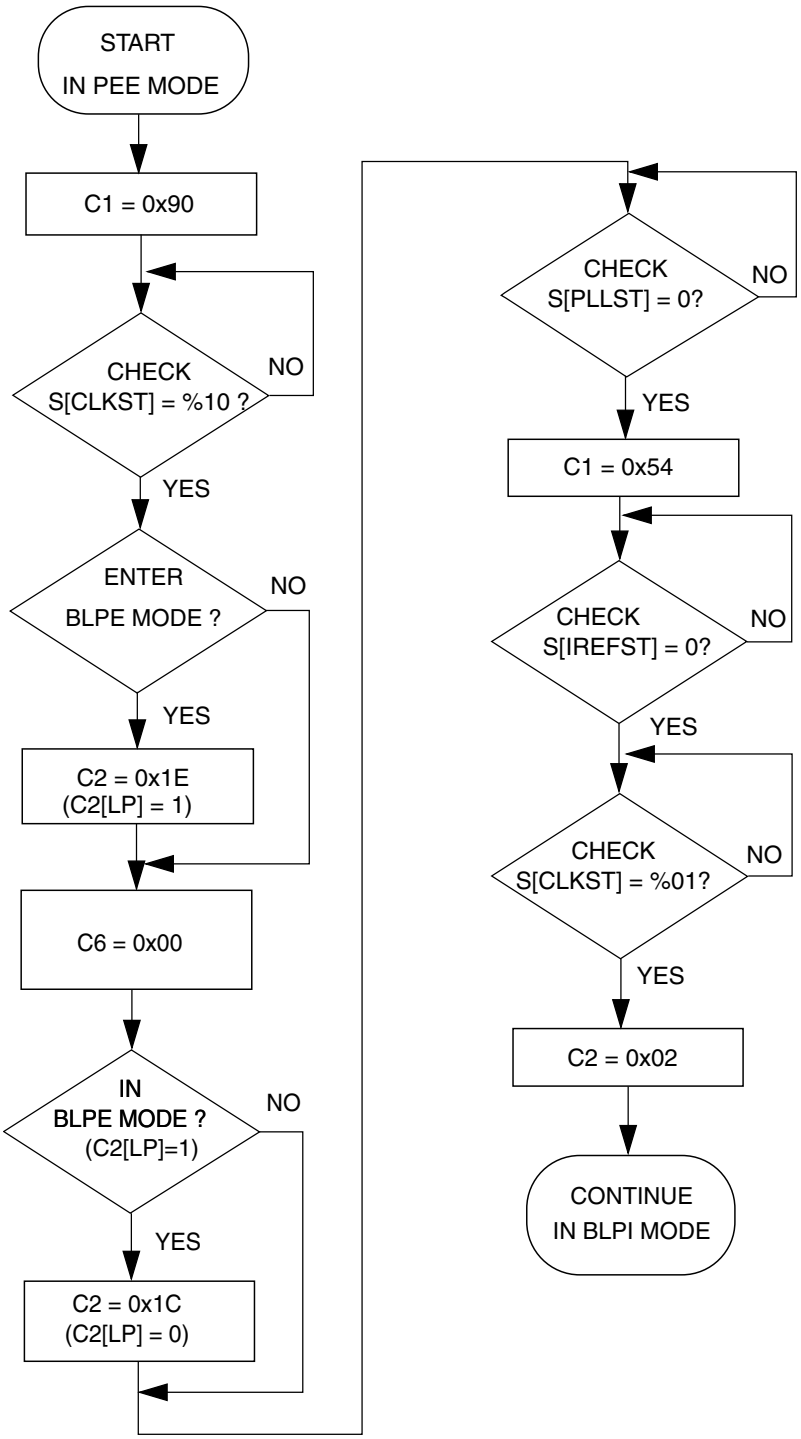



**Figure 25-16. Flowchart of PEE to BLPI mode transition using an 4 MHz crystal**

### 25.5.3.3 Example 3: Moving from BLPI to FEE mode

In this example, the MCG will move through the proper operational modes from BLPI mode at a 32 kHz MCGOUTCLK frequency running off the internal reference clock (see previous example) to FEE mode using a 4 MHz crystal configured for a 20 MHz MCGOUTCLK frequency. First, the code sequence will be described. Then there is a flowchart that illustrates the sequence.

1. First, BLPI must transition to FBI mode.
   a. C2 = 0x00
      - C2[LP] is 0
2. Next, FBI will transition to FEE mode.
   a. C2 = 0x1C
      - C2[RANGE0] set to 2'b01 because the frequency of 4 MHz is within the high frequency range.
      - C2[HGO0] set to 1 to configure the crystal oscillator for high gain operation.
      - C2[EREFS0] set to 1, because a crystal is being used.
   b. C1 = 0x10
      - C1[CLKS] set to 2'b00 to select the output of the FLL as system clock source.
      - C1[FRDIV] remain at 3'b010, or divide-by-128 for a reference of 4 MHz / 128 = 31.25 kHz.
      - C1[IREFS] cleared to 0, selecting the external reference clock.
   c. Loop until S[OSCINIT0] is 1, indicating the crystal selected by the C2[EREFS0] bit has been initialized.
   d. Loop until S[IREFST] is 0, indicating the external reference clock is the current source for the reference clock.
   e. Loop until S[CLKST] are 2'b00, indicating that the output of the FLL is selected to feed MCGOUTCLK.
   f. Now, with a 31.25 kHz reference frequency, a fixed DCO multiplier of 640, MCGOUTCLK = 31.25 kHz * 640 / 1 = 20 MHz.
   g. At this point, by default, the C4[DRST_DRS] bits are set to 2'b00 and C4[DMX32] is cleared to 0. If the MCGOUTCLK frequency of 40 MHz is desired instead, set the C4[DRST_DRS] bits to 0x01 to switch the FLL

multiplication factor from 640 to 1280. To return the MCGOUTCLK frequency to 20 MHz, set C4[DRST_DRS] bits to 2'b00 again, and the FLL multiplication factor will switch back to 640.

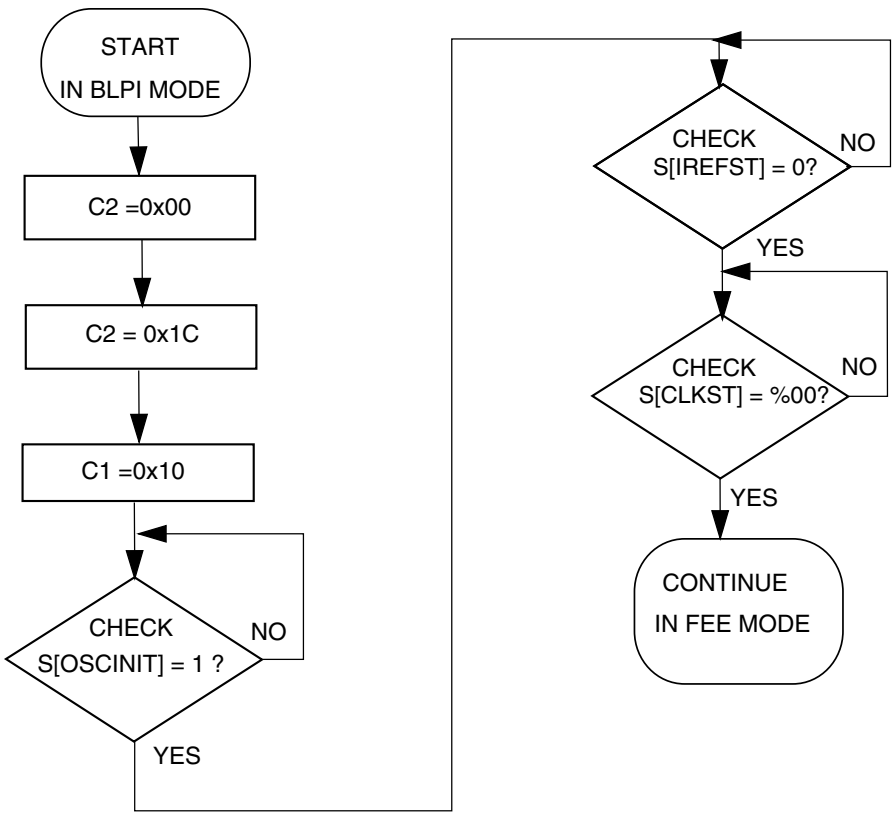



**Figure 25-17. Flowchart of BLPI to FEE mode transition using an 4 MHz crystal**

# Chapter 26
# Oscillator (OSC)

## 26.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The OSC module is a crystal oscillator. The module, in conjunction with an external crystal or resonator, generates a reference clock for the MCU.

## 26.2 Features and Modes

Key features of the module are:

- Supports 32 kHz crystals (Low Range mode)
- Supports 3–8 MHz, 8–32 MHz crystals and resonators (High Range mode)
- Automatic Gain Control (AGC) to optimize power consumption in high frequency ranges 3–8 MHz, 8–32 MHz using low-power mode
- High gain option in frequency ranges: 32 kHz, 3–8 MHz, and 8–32 MHz
- Voltage and frequency filtering to guarantee clock frequency and stability
- Optionally external input bypass clock from EXTAL signal directly
- One clock for MCU clock system
- Two clocks for on-chip peripherals that can work in Stop modes

Functional Description describes the module's operation in more detail.

## 26.3 Block Diagram

The OSC module uses a crystal or resonator to generate three filtered oscillator clock signals. Three clocks are output from OSC module: OSCCLK for MCU system, OSCERCLK for on-chip peripherals, and OSC32KCLK. The OSCCLK can only work in run mode. OSCERCLK and OSC32KCLK can work in low power modes. For the clock source assignments, refer to the clock distribution information of this MCU.

Refer to the chip configuration chapter for the external reference clock source in this MCU.

The following figure shows the block diagram of the OSC module.

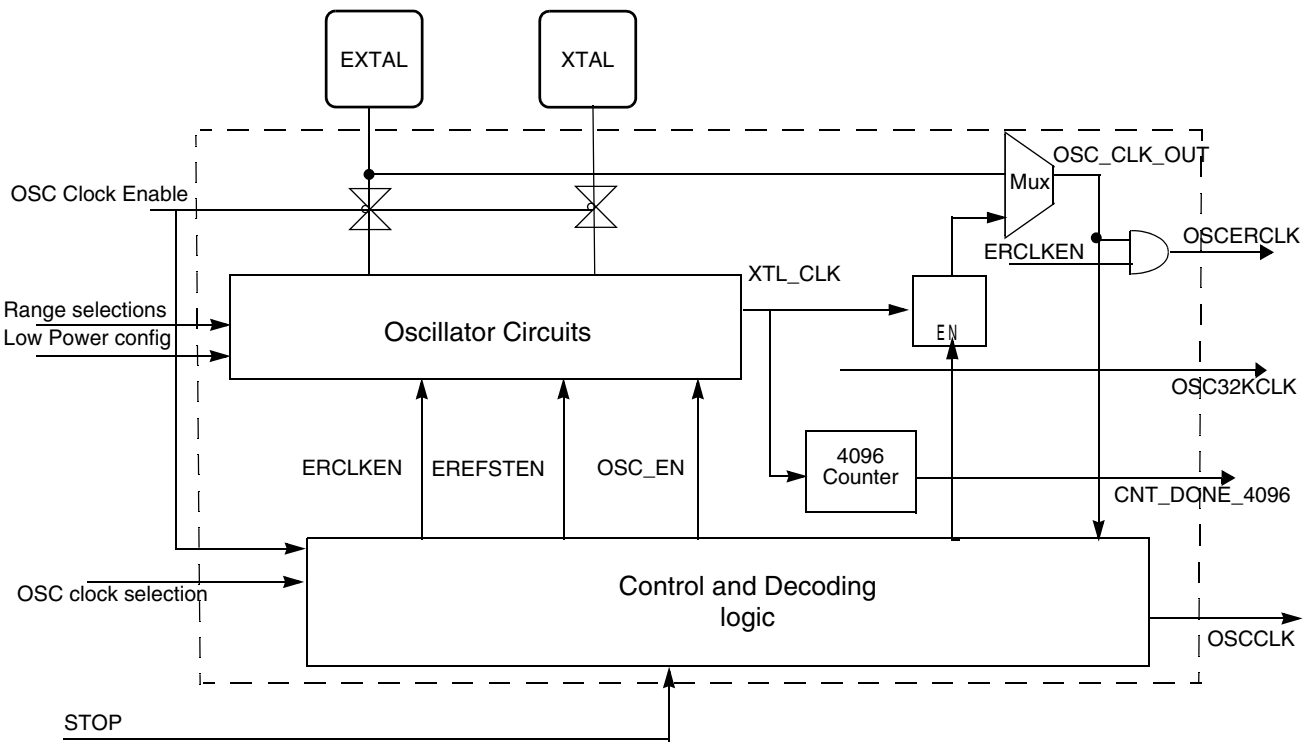



**Figure 26-1. OSC Module Block Diagram**

## 26.4 OSC Signal Descriptions

The following table shows the user-accessible signals available for the OSC module. Refer to signal multiplexing information for this MCU for more details.

**Table 26-1. OSC Signal Descriptions**

| Signal | Description | I/O |
|---|---|---|
| EXTAL | External clock/Oscillator input | I |
| XTAL | Oscillator output | O |

## 26.5 External Crystal / Resonator Connections

The connections for a crystal/resonator frequency reference are shown in the following figures. When using low-frequency, low-power mode, the only external component is the crystal or ceramic resonator itself. In the other oscillator modes, load capacitors ($C_x$, $C_y$) and feedback resistor ($R_F$) are required. The following table shows all possible connections.

**Table 26-2. External Caystal/Resonator Connections**

| Oscillator Mode | Connections |
|---|---|
| Low-frequency (32 kHz), low-power | Connection 1 |
| Low-frequency (32 kHz), high-gain | Connection 2/Connection 3[1] |
| High-frequency (3~32 MHz), low-power | Connection 1/Connection 3[2,2] |
| High-frequency (3~32 MHz), high-gain | Connection 2/Connection 3[2] |

1. When the load capacitors (Cx, Cy) are greater than 30 pF, use Connection 3.
2. With the low-power mode, the oscillator has the internal feedback resistor $R_F$. Therefore, the feedback resistor must not be externally with the Connection 3.

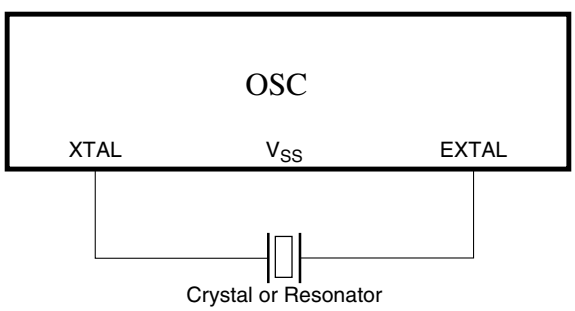



**Figure 26-2. Crystal/Ceramic Resonator Connections - Connection 1**

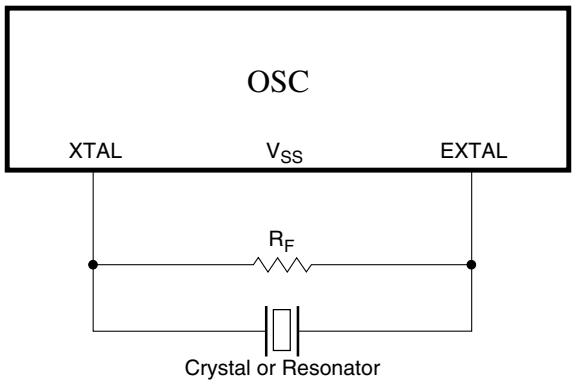



**Figure 26-3. Crystal/Ceramic Resonator Connections - Connection 2**

**NOTE**

Connection 1 and Connection 2 should use internal capacitors as the load of the oscillator by configuring the CR[SCxP] bits.

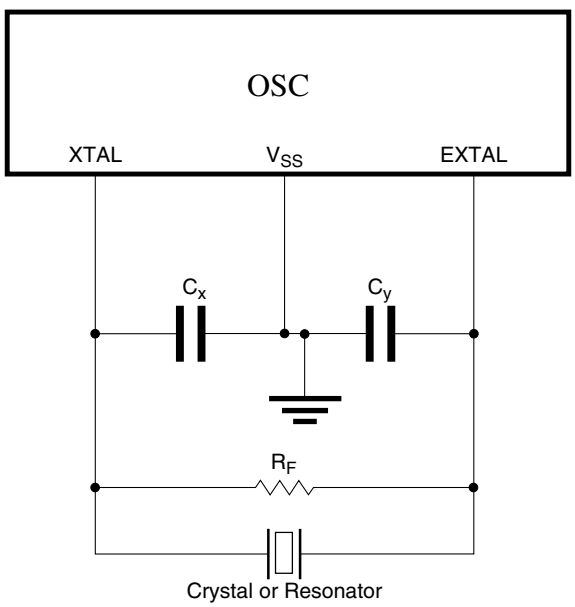



**Figure 26-4. Crystal/Ceramic Resonator Connections - Connection 3**

## 26.6 External Clock Connections

In external clock mode, the pins can be connected as shown below.

**NOTE**

XTAL can be used as a GPIO when the GPIO alternate function is configured for it.

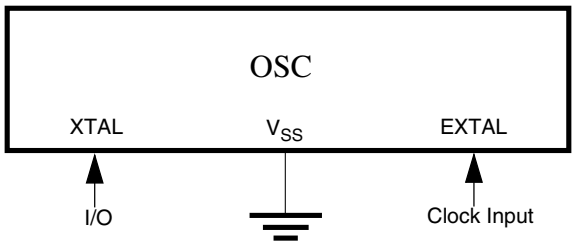



**Figure 26-5. External Clock Connections**

## 26.7 Memory Map/Register Definitions

Some oscillator module register bits are typically incorporated into other peripherals such as MCG or SIM.

### 26.7.1 OSC Memory Map/Register Definition

**OSC memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4006_5000 | OSC Control Register (OSC_CR) | 8 | R/W | 00h | 26.71.1/ 575 |

### 26.71.1 OSC Control Register (OSC_CR)

**NOTE**

After OSC is enabled and starts generating the clocks, the configurations such as low power and frequency range, must not be changed.

Address: 4006_5000h base + 0h offset = 4006_5000h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | ERCLKEN | 0 | EREFSTEN | 0 | SC2P | SC4P | SC8P | SC16P |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**OSC_CR field descriptions**

| Field | Description |
|---|---|
| 7<br>ERCLKEN | External Reference Enable<br><br>Enables external reference clock (OSCERCLK).<br><br>0 External reference clock is inactive.<br>1 External reference clock is enabled. |
| 6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5<br>EREFSTEN | External Reference Stop Enable<br><br>Controls whether or not the external reference clock (OSCERCLK) remains enabled when MCU enters Stop mode.<br><br>0 External reference clock is disabled in Stop mode.<br>1 External reference clock stays enabled in Stop mode if ERCLKEN is set before entering Stop mode. |
| 4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3<br>SC2P | Oscillator 2 pF Capacitor Load Configure<br><br>Configures the oscillator load.<br><br>0 Disable the selection.<br>1 Add 2 pF capacitor to the oscillator load. |
| 2<br>SC4P | Oscillator 4 pF Capacitor Load Configure<br><br>Configures the oscillator load.<br><br>0 Disable the selection.<br>1 Add 4 pF capacitor to the oscillator load. |
| 1<br>SC8P | Oscillator 8 pF Capacitor Load Configure<br><br>Configures the oscillator load.<br><br>0 Disable the selection.<br>1 Add 8 pF capacitor to the oscillator load. |
| 0<br>SC16P | Oscillator 16 pF Capacitor Load Configure<br><br>Configures the oscillator load.<br><br>0 Disable the selection.<br>1 Add 16 pF capacitor to the oscillator load. |

## 26.8 Functional Description

This following sections provide functional details of the module.

## 26.8.1 OSC Module States

The states of the OSC module are shown in the following figure. The states and their transitions between each other are described in this section.

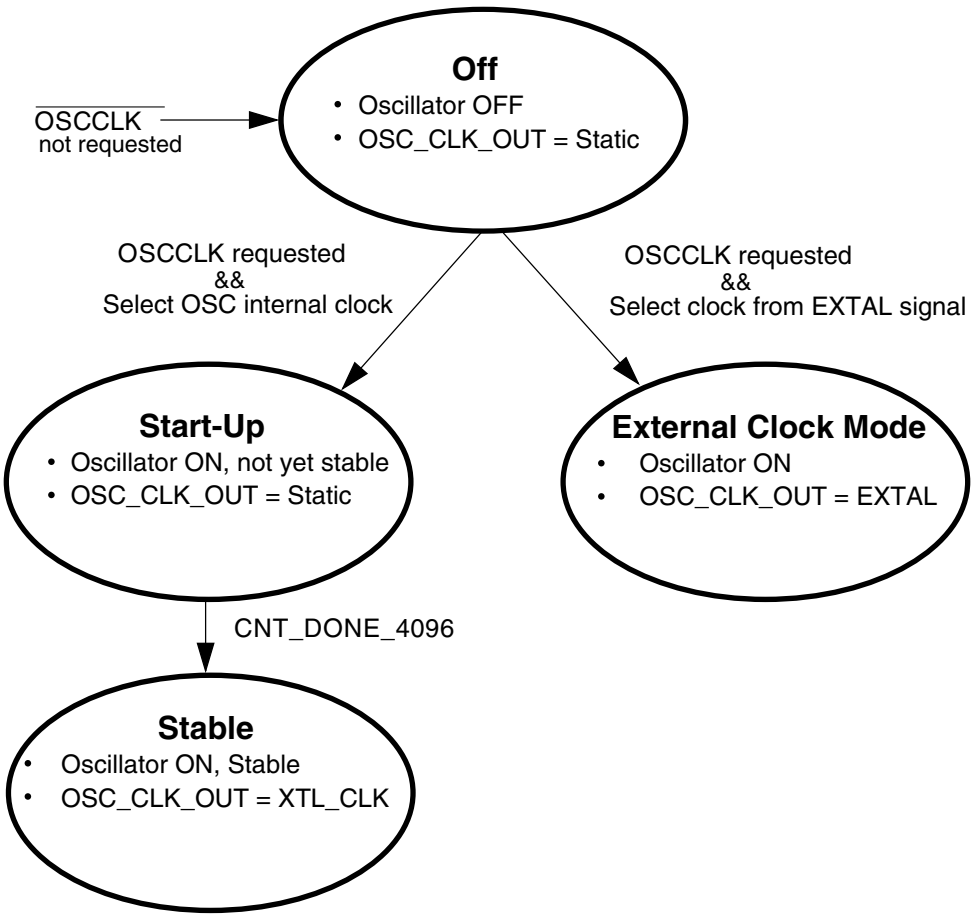



**Figure 26-7. OSC Module State Diagram**

### NOTE

XTL_CLK is the clock generated internally from OSC circuits.

### 26.8.1.1 Off

The OSC enters the Off state when the system does not require OSC clocks. Upon entering this state, XTL_CLK is static unless OSC is configured to select the clock from the EXTAL pad by clearing the external reference clock selection bit. For details regarding the external reference clock source in this MCU, refer to the chip configuration chapter. The EXTAL and XTAL pins are also decoupled from all other oscillator circuitry in this state. The OSC module circuitry is configured to draw minimal current.

#### 26.8.1.2 Oscillator Start-Up

The OSC enters start-up state when it is configured to generate clocks (internally the OSC_EN transitions high) using the internal oscillator circuits by setting the external reference clock selection bit. In this state, the OSC module is enabled and oscillations are starting up, but have not yet stabilized. When the oscillation amplitude becomes large enough to pass through the input buffer, XTL_CLK begins clocking the counter. When the counter reaches 4096 cycles of XTL_CLK, the oscillator is considered stable and XTL_CLK is passed to the output clock OSC_CLK_OUT.

#### 26.8.1.3 Oscillator Stable

The OSC enters stable state when it is configured to generate clocks (internally the OSC_EN transitions high) using the internal oscillator circuits by setting the external reference clock selection bit and the counter reaches 4096 cycles of XTL_CLK (when CNT_DONE_4096 is high). In this state, the OSC module is producing a stable output clock on OSC_CLK_OUT. Its frequency is determined by the external components being used.

#### 26.8.1.4 External Clock Mode

The OSC enters external clock state when it is enabled and external reference clock selection bit is cleared. For details regarding external reference clock source in this MCU, refer to the chip configuration chapter. In this state, the OSC module is set to buffer (with hysteresis) a clock from EXTAL onto the OSC_CLK_OUT. Its frequency is determined by the external clock being supplied.

### 26.8.2 OSC Module Modes

The OSC is a Pierce-type oscillator that supports external crystals or resonators operating over the frequency ranges shown in Table 26-5. These modes assume the following conditions: OSC is enabled to generate clocks (OSC_EN=1), configured to generate clocks internally (MCG_C2[EREFS] = 1), and some or one of the other peripherals (MCG, Timer, and so on) is configured to use the oscillator output clock (OSC_CLK_OUT).

**Table 26-5. Oscillator Modes**

| Mode | Frequency Range |
|---|---|
| Low-frequency, high-gain | $f_{osc\_lo}$ (1 kHz) up to $f_{osc\_lo}$ (32.768 kHz) |
| High-frequency mode1, high-gain | $f_{osc\_hi\_1}$ (3 MHz) up to $f_{osc\_hi\_1}$ (8 MHz) |
| High-frequency mode1, low-power | |
| High-frequency mode2, high-gain | $f_{osc\_hi\_2}$ (8 MHz) up to $f_{osc\_hi\_2}$ (32 MHz) |
| High-frequency mode2, low-power | |

**NOTE**

For information about low power modes of operation used in this chip and their alignment with some OSC modes, refer to the chip's Power Management details.

### 26.8.2.1 Low-Frequency, High-Gain Mode

In Low-frequency, high-gain mode, the oscillator uses a simple inverter-style amplifier. The gain is set to achieve rail-to-rail oscillation amplitudes.

The oscillator input buffer in this mode is single-ended. It provides low pass frequency filtering as well as hysteresis for voltage filtering and converts the output to logic levels. In this mode, the internal capacitors could be used.

### 26.8.2.2 Low-Frequency, Low-Power Mode

In low-frequency, low-power mode, the oscillator uses a gain control loop to minimize power consumption. As the oscillation amplitude increases, the amplifier current is reduced. This continues until a desired amplitude is achieved at steady-state. This mode provides low pass frequency filtering as well as hysteresis for voltage filtering and converts the output to logic levels. In this mode, the internal capacitors could be used, the internal feedback resistor is connected, and no external resistor should be used.

In this mode, the amplifier inputs, gain-control input, and input buffer input are all capacitively coupled for leakage tolerance (not sensitive to the DC level of EXTAL).

Also in this mode, all external components except for the resonator itself are integrated, which includes the load capacitors and feeback resistor that biases EXTAL.

#### 26.8.2.3 High-Frequency, High-Gain Mode

In high-frequency, high-gain mode, the oscillator uses a simple inverter-style amplifier. The gain is set to achieve rail-to-rail oscillation amplitudes. This mode provides low pass frequency filtering as well as hysteresis for voltage filtering and converts the output to logic levels. In this mode, the internal capacitors could be used.

#### 26.8.2.4 High-Frequency, Low-Power Mode

In high-frequency, low-power mode, the oscillator uses a gain control loop to minimize power consumption. As the oscillation amplitude increases, the amplifier current is reduced. This continues until a desired amplitude is achieved at steady-state. In this mode, the internal capacitors could be used, the internal feedback resistor is connected, and no external resistor should be used.

The oscillator input buffer in this mode is differential. It provides low pass frequency filtering as well as hysteresis for voltage filtering and converts the output to logic levels.

### 26.8.3 Counter

The oscillator output clock (OSC_CLK_OUT) is gated off until the counter has detected 4096 cycles of its input clock (XTL_CLK). After 4096 cycles are completed, the counter passes XTL_CLK onto OSC_CLK_OUT. This counting time-out is used to guarantee output clock stability.

### 26.8.4 Reference Clock Pin Requirements

The OSC module requires use of both the EXTAL and XTAL pins to generate an output clock in Oscillator mode, but requires only the EXTAL pin in External clock mode. The EXTAL and XTAL pins are available for I/O. For the implementation of these pins on this device, refer to the Signal Multiplexing chapter.

## 26.9 Reset

There is no reset state associated with the OSC module. The counter logic is reset when the OSC is not configured to generate clocks.

There are no sources of reset requests for the OSC module.

## 26.10 Low Power Modes Operation

When the MCU enters Stop modes, the OSC is functional depending on ERCLKEN and EREFSETN bit settings. If both these bits are set, the OSC is in operation. In Low Leakage Stop (LLS) modes, the OSC holds all register settings. If ERCLKEN and EREFSTEN bits are set before entry to Low Leakage Stop modes, the OSC is still functional in these modes. After waking up from Very Low Leakage Stop (VLLSx) modes, all OSC register bits are reset and initialization is required through software.

## 26.11 Interrupts

The OSC module does not generate any interrupts.

# Chapter 27
# RTC Oscillator (OSC32K)

## 27.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The RTC oscillator module provides the clock source for the RTC. The RTC oscillator module, in conjunction with an external crystal, generates a reference clock for the RTC.

### 27.1.1 Features and Modes

The key features of the RTC oscillator are as follows:

- Supports 32 kHz crystals with very low power
- Consists of internal feed back resistor
- Consists of internal programmable capacitors as the $C_{load}$ of the oscillator
- Automatic Gain Control (AGC) to optimize power consumption

The RTC oscillator operations are described in detail in Functional Description .

### 27.1.2 Block Diagram

The following is the block diagram of the RTC oscillator.

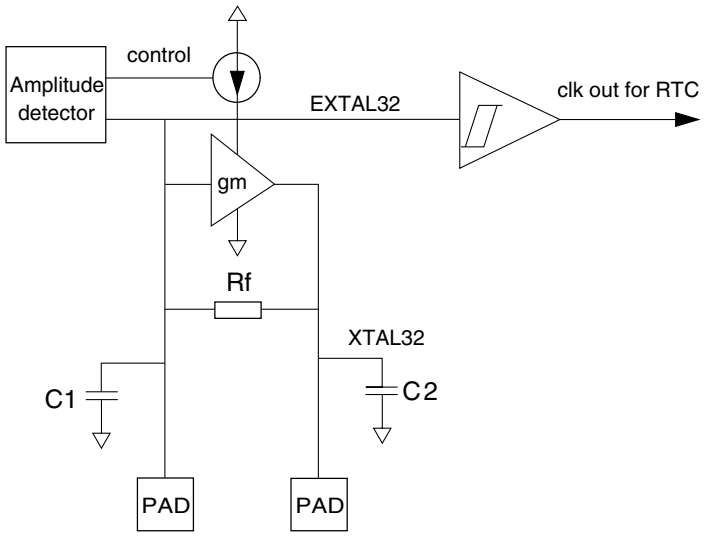



**Figure 27-1. RTC Oscillator Block Diagram**

## 27.2 RTC Signal Descriptions

The following table shows the user-accessible signals available for the RTC oscillator. See the chip-level specification to find out which signals are actually connected to the external pins.

**Table 27-1. RTC Signal Descriptions**

| Signal | Description | I/O |
|---|---|---|
| EXTAL32 | Oscillator Input | I |
| XTAL32 | Oscillator Output | O |

### 27.2.1 EXTAL32 — Oscillator Input

This signal is the analog input of the RTC oscillator.

### 27.2.2 XTAL32 — Oscillator Output

This signal is the analog output of the RTC oscillator module.

## 27.3 External Crystal Connections

The connections with a crystal is shown in the following figure. External load capacitors and feedback resistor are not required.

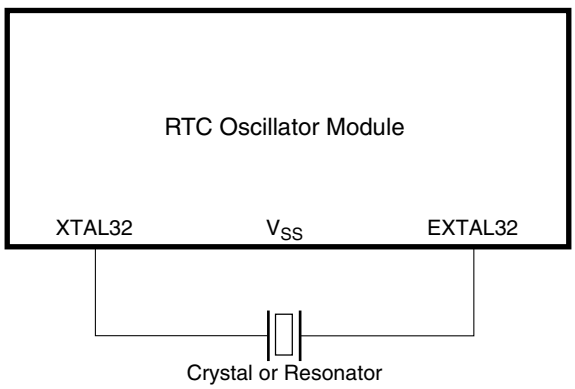



**Figure 27-2. Crystal Connections**

## 27.4 Memory Map/Register Descriptions

RTC oscillator control bits are part of the RTC registers. Refer to RTC Control register for more details.

## 27.5 Functional Description

As shown in Figure 27-1, the module includes an amplifier which supplies the negative resistor for the RTC oscillator. The gain of the amplifier is controlled by the amplitude detector, which optimizes the power consumption. A schmitt trigger is used to translate the sine-wave generated by this oscillator to a pulse clock out, which is a reference clock for the RTC digital core.

The oscillator includes an internal feedback resistor of approximately 100 MΩ between EXTAL32 and XTAL32.

In addition, there are two programmable capacitors with this oscillator, which can be used as the Cload of the oscillator. The programmable range is from 0pF to 30pF.

## 27.6 Reset Overview

There is no reset state associated with the RTC oscillator.

## 27.7 Interrupts

The RTC oscillator does not generate any interrupts.

# Chapter 28
# Flash Memory Controller (FMC)

## 28.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The Flash Memory Controller (FMC) is a memory acceleration unit that provides:

- an interface between the device and the dual-bank nonvolatile memory. Bank 0 consists of program flash memory, and bank 1 consists of FlexNVM.
- buffers that can accelerate flash memory and FlexNVM data transfers.

### 28.1.1 Overview

The Flash Memory Controller manages the interface between the device and the dual-bank flash memory. The FMC receives status information detailing the configuration of the memory and uses this information to ensure a proper interface. The following table shows the supported read/write operations.

| Flash memory type | Read | Write |
|---|---|---|
| Program flash memory | 8-bit, 16-bit, and 32-bit reads | —[1] |
| FlexNVM used as Data flash memory | 8-bit, 16-bit, and 32-bit reads | —[1] |
| FlexNVM and FlexRAM used as EEPROM | 8-bit, 16-bit, and 32-bit reads | 8-bit, 16-bit, and 32-bit writes |

1. A write operation to program flash memory or to FlexNVM used as data flash memory results in a bus error.

In addition, for bank 0 and bank 1, the FMC provides three separate mechanisms for accelerating the interface between the device and the flash memory. A 128-bit speculation buffer can prefetch the next 128-bit flash memory location, and both a 4-way, 4-set cache and a single-entry 128-bit buffer can store previously accessed flash memory or FlexNVM data for quick access times.

### 28.1.2 Features

The FMC's features include:

- Interface between the device and the dual-bank flash memory and FlexMemory:
  - 8-bit, 16-bit, and 32-bit read operations to program flash memory and FlexNVM used as data flash memory.
  - 8-bit, 16-bit, and 32-bit read and write operations to FlexNVM and FlexRAM used as EEPROM.
  - For bank 0 and bank 1: Read accesses to consecutive 32-bit spaces in memory return the second, third, and fourth read data with no wait states. The memory returns 128 bits via the 32-bit bus access.
  - Crossbar master access protection for setting no access, read-only access, write-only access, or read/write access for each crossbar master.
- For bank 0 and bank 1: Acceleration of data transfer from program flash memory and FlexMemory to the device:
  - 128-bit prefetch speculation buffer with controls for instruction/data access per master and bank
  - 4-way, 4-set, 128-bit line size cache for a total of sixteen 128-bit entries with controls for replacement algorithm and lock per way for each bank
  - Single-entry buffer with enable per bank
  - Invalidation control for the speculation buffer and the single-entry buffer

## 28.2 Modes of operation

The FMC only operates when the device accesses the flash memory or FlexMemory.

In terms of device power modes, the FMC only operates in run and wait modes, including VLPR and VLPW modes.

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

## 28.3 External signal description

The FMC has no external signals.

## 28.4 Memory map and register descriptions

The programming model consists of the FMC control registers and the program visible cache (data and tag/valid entries).

**NOTE**

Program the registers only while the flash controller is idle (for example, execute from RAM). Changing configuration settings while a flash access is in progress can lead to non-deterministic behavior.

**Table 28-2. FMC register access**

| Registers | Read access | | Write access | |
|---|---|---|---|---|
| | Mode | Length | Mode | Length |
| Control registers: PFAPR, PFB0CR, PFB1CR | Supervisor (privileged) mode or user mode | 32 bits | Supervisor (privileged) mode only | 32 bits |
| Cache registers | Supervisor (privileged) mode or user mode | 32 bits | Supervisor (privileged) mode only | 32 bits |

**NOTE**

Accesses to unimplemented registers within the FMC's 4 KB address space return a bus error.

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

**NOTE**

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

The cache is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. The following table elaborates on the tag/valid and data entries.

## Table 28-3. Program visible cache registers

| Cache storage | Based at offset | Contents of 32-bit read | Nomenclature | Nomenclature example |
|---|---|---|---|---|
| Tag | 100h | 12'h0, tag[19:6], 5'h0, valid | In TAGVDWxSy, x denotes the way and y denotes the set. | TAGVDW2S0 is the 14-bit tag and 1-bit valid for cache entry way 2, set 0. |
| Data | 200h | One of the four longwords in a 128-bit cache entry | In DATAWxSyUM, DATAWxSyMU, DATAWxSyML, and DATAWxSyLM, x denotes the way, y denotes the set, and the final two letters identify the word: UM (uppermost), MU (mid-upper), ML (mid-lower), and LM (lowermost). | For data entry way 1, set 3, DATAW1S3UM represents bits [127:96], DATAW1S3MU represents bits [95:64], DATAW1S3ML represents bits [63:32], and DATAW1S3LM represents bits [31:0]. |

## FMC memory map

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4001_F000 | Flash Access Protection Register (FMC_PFAPR) | 32 | R/W | 00F8_003Fh | 28.4.1/594 |
| 4001_F004 | Flash Bank 0 Control Register (FMC_PFB0CR) | 32 | R/W | 3004_001Fh | 28.4.2/597 |
| 4001_F008 | Flash Bank 1 Control Register (FMC_PFB1CR) | 32 | R/W | 3004_001Fh | 28.4.3/600 |
| 4001_F100 | Cache Tag Storage (FMC_TAGVDW0S0) | 32 | R/W | 0000_0000h | 28.4.4/602 |
| 4001_F104 | Cache Tag Storage (FMC_TAGVDW0S1) | 32 | R/W | 0000_0000h | 28.4.4/602 |
| 4001_F108 | Cache Tag Storage (FMC_TAGVDW0S2) | 32 | R/W | 0000_0000h | 28.4.4/602 |
| 4001_F10C | Cache Tag Storage (FMC_TAGVDW0S3) | 32 | R/W | 0000_0000h | 28.4.4/602 |
| 4001_F110 | Cache Tag Storage (FMC_TAGVDW1S0) | 32 | R/W | 0000_0000h | 28.4.5/603 |
| 4001_F114 | Cache Tag Storage (FMC_TAGVDW1S1) | 32 | R/W | 0000_0000h | 28.4.5/603 |
| 4001_F118 | Cache Tag Storage (FMC_TAGVDW1S2) | 32 | R/W | 0000_0000h | 28.4.5/603 |
| 4001_F11C | Cache Tag Storage (FMC_TAGVDW1S3) | 32 | R/W | 0000_0000h | 28.4.5/603 |
| 4001_F120 | Cache Tag Storage (FMC_TAGVDW2S0) | 32 | R/W | 0000_0000h | 28.4.6/604 |
| 4001_F124 | Cache Tag Storage (FMC_TAGVDW2S1) | 32 | R/W | 0000_0000h | 28.4.6/604 |
| 4001_F128 | Cache Tag Storage (FMC_TAGVDW2S2) | 32 | R/W | 0000_0000h | 28.4.6/604 |
| 4001_F12C | Cache Tag Storage (FMC_TAGVDW2S3) | 32 | R/W | 0000_0000h | 28.4.6/604 |
| 4001_F130 | Cache Tag Storage (FMC_TAGVDW3S0) | 32 | R/W | 0000_0000h | 28.4.7/605 |
| 4001_F134 | Cache Tag Storage (FMC_TAGVDW3S1) | 32 | R/W | 0000_0000h | 28.4.7/605 |
| 4001_F138 | Cache Tag Storage (FMC_TAGVDW3S2) | 32 | R/W | 0000_0000h | 28.4.7/605 |
| 4001_F13C | Cache Tag Storage (FMC_TAGVDW3S3) | 32 | R/W | 0000_0000h | 28.4.7/605 |
| 4001_F200 | Cache Data Storage (uppermost word) (FMC_DATAW0S0UM) | 32 | R/W | 0000_0000h | 28.4.8/606 |
| 4001_F204 | Cache Data Storage (mid-upper word) (FMC_DATAW0S0MU) | 32 | R/W | 0000_0000h | 28.4.9/606 |
| 4001_F208 | Cache Data Storage (mid-lower word) (FMC_DATAW0S0ML) | 32 | R/W | 0000_0000h | 28.4.10/ 607 |

*Table continues on the next page...*

## FMC memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4001_F20C | Cache Data Storage (lowermost word) (FMC_DATAW0S0LM) | 32 | R/W | 0000_0000h | 28.4.11/ 607 |
| 4001_F210 | Cache Data Storage (uppermost word) (FMC_DATAW0S1UM) | 32 | R/W | 0000_0000h | 28.4.8/606 |
| 4001_F214 | Cache Data Storage (mid-upper word) (FMC_DATAW0S1MU) | 32 | R/W | 0000_0000h | 28.4.9/606 |
| 4001_F218 | Cache Data Storage (mid-lower word) (FMC_DATAW0S1ML) | 32 | R/W | 0000_0000h | 28.4.10/ 607 |
| 4001_F21C | Cache Data Storage (lowermost word) (FMC_DATAW0S1LM) | 32 | R/W | 0000_0000h | 28.4.11/ 607 |
| 4001_F220 | Cache Data Storage (uppermost word) (FMC_DATAW0S2UM) | 32 | R/W | 0000_0000h | 28.4.8/606 |
| 4001_F224 | Cache Data Storage (mid-upper word) (FMC_DATAW0S2MU) | 32 | R/W | 0000_0000h | 28.4.9/606 |
| 4001_F228 | Cache Data Storage (mid-lower word) (FMC_DATAW0S2ML) | 32 | R/W | 0000_0000h | 28.4.10/ 607 |
| 4001_F22C | Cache Data Storage (lowermost word) (FMC_DATAW0S2LM) | 32 | R/W | 0000_0000h | 28.4.11/ 607 |
| 4001_F230 | Cache Data Storage (uppermost word) (FMC_DATAW0S3UM) | 32 | R/W | 0000_0000h | 28.4.8/606 |
| 4001_F234 | Cache Data Storage (mid-upper word) (FMC_DATAW0S3MU) | 32 | R/W | 0000_0000h | 28.4.9/606 |
| 4001_F238 | Cache Data Storage (mid-lower word) (FMC_DATAW0S3ML) | 32 | R/W | 0000_0000h | 28.4.10/ 607 |
| 4001_F23C | Cache Data Storage (lowermost word) (FMC_DATAW0S3LM) | 32 | R/W | 0000_0000h | 28.4.11/ 607 |
| 4001_F240 | Cache Data Storage (uppermost word) (FMC_DATAW1S0UM) | 32 | R/W | 0000_0000h | 28.4.12/ 608 |
| 4001_F244 | Cache Data Storage (mid-upper word) (FMC_DATAW1S0MU) | 32 | R/W | 0000_0000h | 28.4.13/ 608 |
| 4001_F248 | Cache Data Storage (mid-lower word) (FMC_DATAW1S0ML) | 32 | R/W | 0000_0000h | 28.4.14/ 609 |
| 4001_F24C | Cache Data Storage (lowermost word) (FMC_DATAW1S0LM) | 32 | R/W | 0000_0000h | 28.4.15/ 609 |
| 4001_F250 | Cache Data Storage (uppermost word) (FMC_DATAW1S1UM) | 32 | R/W | 0000_0000h | 28.4.12/ 608 |
| 4001_F254 | Cache Data Storage (mid-upper word) (FMC_DATAW1S1MU) | 32 | R/W | 0000_0000h | 28.4.13/ 608 |
| 4001_F258 | Cache Data Storage (mid-lower word) (FMC_DATAW1S1ML) | 32 | R/W | 0000_0000h | 28.4.14/ 609 |
| 4001_F25C | Cache Data Storage (lowermost word) (FMC_DATAW1S1LM) | 32 | R/W | 0000_0000h | 28.4.15/ 609 |
| 4001_F260 | Cache Data Storage (uppermost word) (FMC_DATAW1S2UM) | 32 | R/W | 0000_0000h | 28.4.12/ 608 |

*Table continues on the next page...*

## FMC memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4001_F264 | Cache Data Storage (mid-upper word) (FMC_DATAW1S2MU) | 32 | R/W | 0000_0000h | 28.4.13/ 608 |
| 4001_F268 | Cache Data Storage (mid-lower word) (FMC_DATAW1S2ML) | 32 | R/W | 0000_0000h | 28.4.14/ 609 |
| 4001_F26C | Cache Data Storage (lowermost word) (FMC_DATAW1S2LM) | 32 | R/W | 0000_0000h | 28.4.15/ 609 |
| 4001_F270 | Cache Data Storage (uppermost word) (FMC_DATAW1S3UM) | 32 | R/W | 0000_0000h | 28.4.12/ 608 |
| 4001_F274 | Cache Data Storage (mid-upper word) (FMC_DATAW1S3MU) | 32 | R/W | 0000_0000h | 28.4.13/ 608 |
| 4001_F278 | Cache Data Storage (mid-lower word) (FMC_DATAW1S3ML) | 32 | R/W | 0000_0000h | 28.4.14/ 609 |
| 4001_F27C | Cache Data Storage (lowermost word) (FMC_DATAW1S3LM) | 32 | R/W | 0000_0000h | 28.4.15/ 609 |
| 4001_F280 | Cache Data Storage (uppermost word) (FMC_DATAW2S0UM) | 32 | R/W | 0000_0000h | 28.4.16/ 610 |
| 4001_F284 | Cache Data Storage (mid-upper word) (FMC_DATAW2S0MU) | 32 | R/W | 0000_0000h | 28.4.17/ 610 |
| 4001_F288 | Cache Data Storage (mid-lower word) (FMC_DATAW2S0ML) | 32 | R/W | 0000_0000h | 28.4.18/ 611 |
| 4001_F28C | Cache Data Storage (lowermost word) (FMC_DATAW2S0LM) | 32 | R/W | 0000_0000h | 28.4.19/ 611 |
| 4001_F290 | Cache Data Storage (uppermost word) (FMC_DATAW2S1UM) | 32 | R/W | 0000_0000h | 28.4.16/ 610 |
| 4001_F294 | Cache Data Storage (mid-upper word) (FMC_DATAW2S1MU) | 32 | R/W | 0000_0000h | 28.4.17/ 610 |
| 4001_F298 | Cache Data Storage (mid-lower word) (FMC_DATAW2S1ML) | 32 | R/W | 0000_0000h | 28.4.18/ 611 |
| 4001_F29C | Cache Data Storage (lowermost word) (FMC_DATAW2S1LM) | 32 | R/W | 0000_0000h | 28.4.19/ 611 |
| 4001_F2A0 | Cache Data Storage (uppermost word) (FMC_DATAW2S2UM) | 32 | R/W | 0000_0000h | 28.4.16/ 610 |
| 4001_F2A4 | Cache Data Storage (mid-upper word) (FMC_DATAW2S2MU) | 32 | R/W | 0000_0000h | 28.4.17/ 610 |
| 4001_F2A8 | Cache Data Storage (mid-lower word) (FMC_DATAW2S2ML) | 32 | R/W | 0000_0000h | 28.4.18/ 611 |
| 4001_F2AC | Cache Data Storage (lowermost word) (FMC_DATAW2S2LM) | 32 | R/W | 0000_0000h | 28.4.19/ 611 |
| 4001_F2B0 | Cache Data Storage (uppermost word) (FMC_DATAW2S3UM) | 32 | R/W | 0000_0000h | 28.4.16/ 610 |
| 4001_F2B4 | Cache Data Storage (mid-upper word) (FMC_DATAW2S3MU) | 32 | R/W | 0000_0000h | 28.4.17/ 610 |
| 4001_F2B8 | Cache Data Storage (mid-lower word) (FMC_DATAW2S3ML) | 32 | R/W | 0000_0000h | 28.4.18/ 611 |

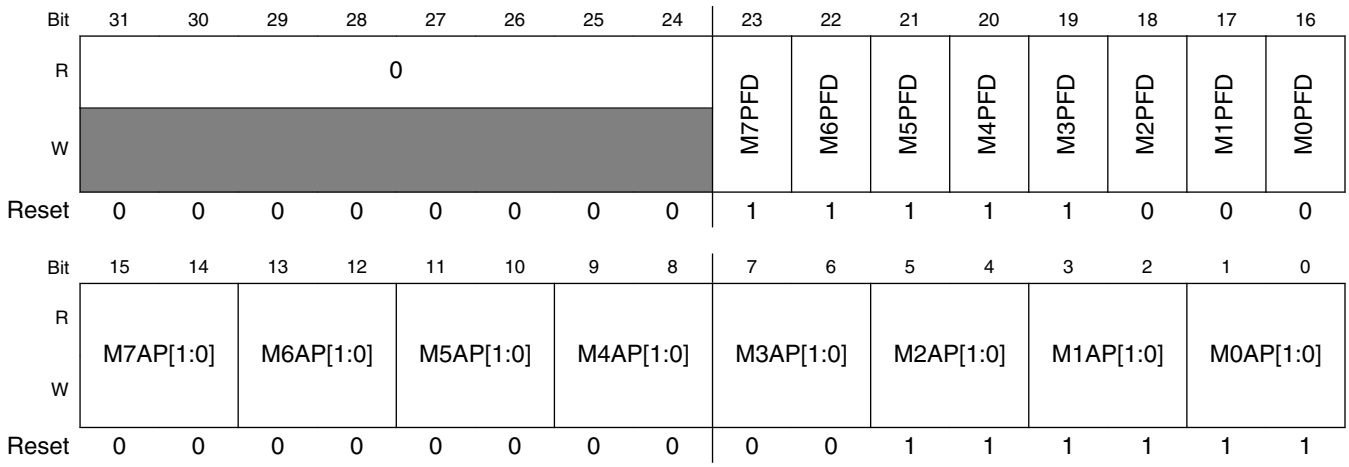
*Table continues on the next page...*

## FMC memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4001_F2BC | Cache Data Storage (lowermost word) (FMC_DATAW2S3LM) | 32 | R/W | 0000_0000h | 28.4.19/ 611 |
| 4001_F2C0 | Cache Data Storage (uppermost word) (FMC_DATAW3S0UM) | 32 | R/W | 0000_0000h | 28.4.20/ 612 |
| 4001_F2C4 | Cache Data Storage (mid-upper word) (FMC_DATAW3S0MU) | 32 | R/W | 0000_0000h | 28.4.21/ 612 |
| 4001_F2C8 | Cache Data Storage (mid-lower word) (FMC_DATAW3S0ML) | 32 | R/W | 0000_0000h | 28.4.22/ 613 |
| 4001_F2CC | Cache Data Storage (lowermost word) (FMC_DATAW3S0LM) | 32 | R/W | 0000_0000h | 28.4.23/ 613 |
| 4001_F2D0 | Cache Data Storage (uppermost word) (FMC_DATAW3S1UM) | 32 | R/W | 0000_0000h | 28.4.20/ 612 |
| 4001_F2D4 | Cache Data Storage (mid-upper word) (FMC_DATAW3S1MU) | 32 | R/W | 0000_0000h | 28.4.21/ 612 |
| 4001_F2D8 | Cache Data Storage (mid-lower word) (FMC_DATAW3S1ML) | 32 | R/W | 0000_0000h | 28.4.22/ 613 |
| 4001_F2DC | Cache Data Storage (lowermost word) (FMC_DATAW3S1LM) | 32 | R/W | 0000_0000h | 28.4.23/ 613 |
| 4001_F2E0 | Cache Data Storage (uppermost word) (FMC_DATAW3S2UM) | 32 | R/W | 0000_0000h | 28.4.20/ 612 |
| 4001_F2E4 | Cache Data Storage (mid-upper word) (FMC_DATAW3S2MU) | 32 | R/W | 0000_0000h | 28.4.21/ 612 |
| 4001_F2E8 | Cache Data Storage (mid-lower word) (FMC_DATAW3S2ML) | 32 | R/W | 0000_0000h | 28.4.22/ 613 |
| 4001_F2EC | Cache Data Storage (lowermost word) (FMC_DATAW3S2LM) | 32 | R/W | 0000_0000h | 28.4.23/ 613 |
| 4001_F2F0 | Cache Data Storage (uppermost word) (FMC_DATAW3S3UM) | 32 | R/W | 0000_0000h | 28.4.20/ 612 |
| 4001_F2F4 | Cache Data Storage (mid-upper word) (FMC_DATAW3S3MU) | 32 | R/W | 0000_0000h | 28.4.21/ 612 |
| 4001_F2F8 | Cache Data Storage (mid-lower word) (FMC_DATAW3S3ML) | 32 | R/W | 0000_0000h | 28.4.22/ 613 |
| 4001_F2FC | Cache Data Storage (lowermost word) (FMC_DATAW3S3LM) | 32 | R/W | 0000_0000h | 28.4.23/ 613 |

## 28.4.1 Flash Access Protection Register (FMC_PFAPR)

Address: 4001_F000h base + 0h offset = 4001_F000h




### FMC_PFAPR field descriptions

| Field | Description |
|---|---|
| 31–24<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 23<br>M7PFD | Master 7 Prefetch Disable<br><br>These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits.<br><br>0 Prefetching for this master is enabled.<br>1 Prefetching for this master is disabled. |
| 22<br>M6PFD | Master 6 Prefetch Disable<br><br>These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits.<br><br>0 Prefetching for this master is enabled.<br>1 Prefetching for this master is disabled. |
| 21<br>M5PFD | Master 5 Prefetch Disable<br><br>These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits.<br><br>0 Prefetching for this master is enabled.<br>1 Prefetching for this master is disabled. |
| 20<br>M4PFD | Master 4 Prefetch Disable<br><br>These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits.<br><br>0 Prefetching for this master is enabled.<br>1 Prefetching for this master is disabled. |

*Table continues on the next page...*

## FMC_PFAPR field descriptions (continued)

| Field | Description |
|---|---|
| 19<br>M3PFD | Master 3 Prefetch Disable<br><br>These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits.<br><br>0 Prefetching for this master is enabled.<br>1 Prefetching for this master is disabled. |
| 18<br>M2PFD | Master 2 Prefetch Disable<br><br>These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits.<br><br>0 Prefetching for this master is enabled.<br>1 Prefetching for this master is disabled. |
| 17<br>M1PFD | Master 1 Prefetch Disable<br><br>These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits.<br><br>0 Prefetching for this master is enabled.<br>1 Prefetching for this master is disabled. |
| 16<br>M0PFD | Master 0 Prefetch Disable<br><br>These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits.<br><br>0 Prefetching for this master is enabled.<br>1 Prefetching for this master is disabled. |
| 15–14<br>M7AP[1:0] | Master 7 Access Protection<br><br>This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master.<br><br>00 No access may be performed by this master.<br>01 Only read accesses may be performed by this master.<br>10 Only write accesses may be performed by this master.<br>11 Both read and write accesses may be performed by this master. |
| 13–12<br>M6AP[1:0] | Master 6 Access Protection<br><br>This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master.<br><br>00 No access may be performed by this master<br>01 Only read accesses may be performed by this master<br>10 Only write accesses may be performed by this master<br>11 Both read and write accesses may be performed by this master |
| 11–10<br>M5AP[1:0] | Master 5 Access Protection<br><br>This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master.<br><br>00 No access may be performed by this master |

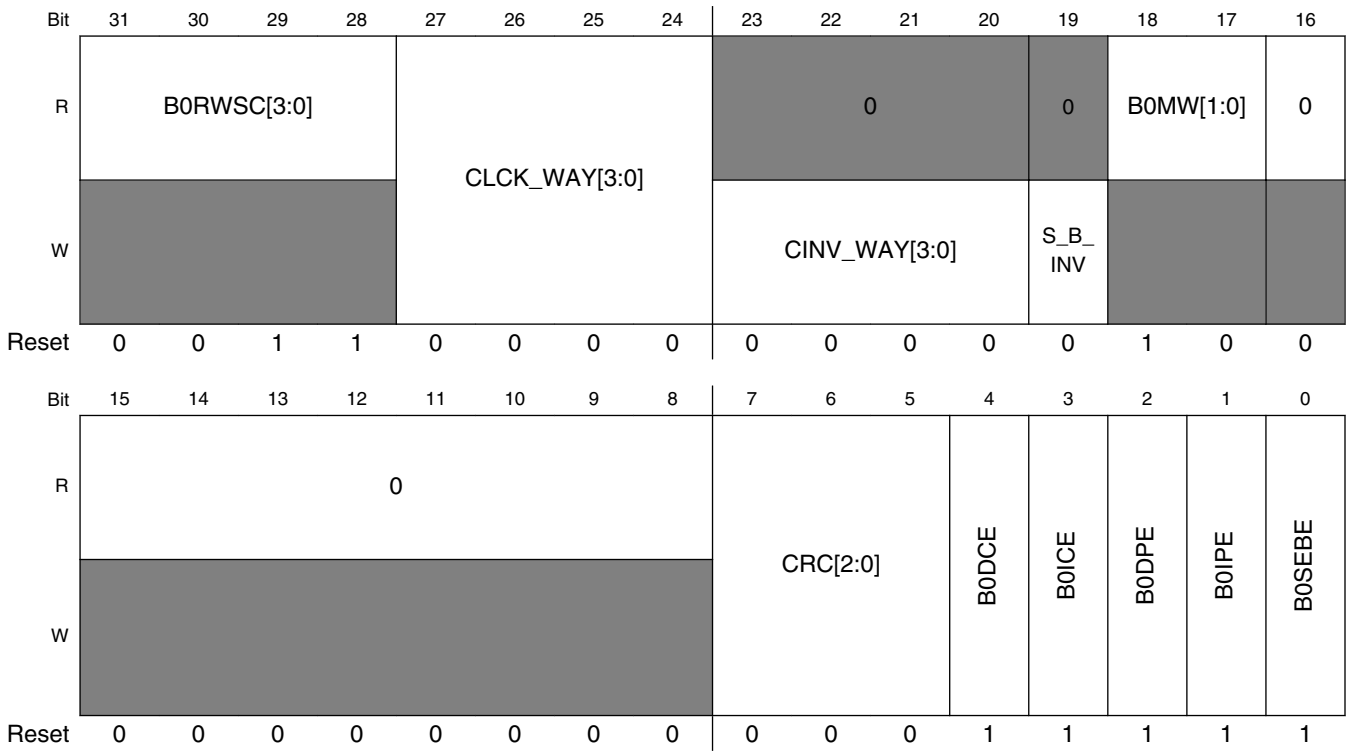
*Table continues on the next page...*

## FMC_PFAPR field descriptions (continued)

| Field | Description |
|---|---|
| | 01 Only read accesses may be performed by this master<br>10 Only write accesses may be performed by this master<br>11 Both read and write accesses may be performed by this master |
| 9–8<br>M4AP[1:0] | Master 4 Access Protection<br><br>This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master.<br><br>00 No access may be performed by this master<br>01 Only read accesses may be performed by this master<br>10 Only write accesses may be performed by this master<br>11 Both read and write accesses may be performed by this master |
| 7–6<br>M3AP[1:0] | Master 3 Access Protection<br><br>This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master.<br><br>00 No access may be performed by this master<br>01 Only read accesses may be performed by this master<br>10 Only write accesses may be performed by this master<br>11 Both read and write accesses may be performed by this master |
| 5–4<br>M2AP[1:0] | Master 2 Access Protection<br><br>This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master.<br><br>00 No access may be performed by this master<br>01 Only read accesses may be performed by this master<br>10 Only write accesses may be performed by this master<br>11 Both read and write accesses may be performed by this master |
| 3–2<br>M1AP[1:0] | Master 1 Access Protection<br><br>This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master.<br><br>00 No access may be performed by this master<br>01 Only read accesses may be performed by this master<br>10 Only write accesses may be performed by this master<br>11 Both read and write accesses may be performed by this master |
| 1–0<br>M0AP[1:0] | Master 0 Access Protection<br><br>This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master.<br><br>00 No access may be performed by this master<br>01 Only read accesses may be performed by this master<br>10 Only write accesses may be performed by this master<br>11 Both read and write accesses may be performed by this master |

## 28.4.2 Flash Bank 0 Control Register (FMC_PFB0CR)

Address: 4001_F000h base + 4h offset = 4001_F004h




### FMC_PFB0CR field descriptions

| Field | Description |
|---|---|
| 31–28<br>B0RWSC[3:0] | Bank 0 Read Wait State Control<br>This read-only field defines the number of wait states required to access the bank 0 flash memory.<br>The relationship between the read access time of the flash array (expressed in system clock cycles) and RWSC is defined as:<br>Access time of flash array [system clocks] = RWSC + 1<br>The FMC automatically calculates this value based on the ratio of the system clock speed to the flash clock speed. For example, when this ratio is 4:1, the field's value is 3h. |
| 27–24<br>CLCK_WAY[3:0] | Cache Lock Way x<br>These bits determine if the given cache way is locked such that its contents will not be displaced by future misses.<br>The bit setting definitions are for each bit in the field.<br>0 Cache way is unlocked and may be displaced<br>1 Cache way is locked and its contents are not displaced |
| 23–20<br>CINV_WAY[3:0] | Cache Invalidate Way x |

*Table continues on the next page...*

## FMC_PFB0CR field descriptions (continued)

| Field | Description |
|---|---|
| | These bits determine if the given cache way is to be invalidated (cleared). When a bit within this field is written, the corresponding cache way is immediately invalidated: the way's tag, data, and valid contents are cleared. This field always reads as zero.<br><br>Cache invalidation takes precedence over locking. The cache is invalidated by system reset. System software is required to maintain memory coherency when any segment of the flash memory is programmed or erased. Accordingly, cache invalidations must occur after a programming or erase event is completed and before the new memory image is accessed.<br><br>The bit setting definitions are for each bit in the field.<br><br>0 No cache way invalidation for the corresponding cache<br>1 Invalidate cache way for the corresponding cache: clear the tag, data, and vld bits of ways selected |
| 19<br>S_B_INV | Invalidate Prefetch Speculation Buffer<br><br>This bit determines if the FMC's prefetch speculation buffer and the single entry page buffer are to be invalidated (cleared). When this bit is written, the speculation buffer and single entry buffer are immediately cleared. This bit always reads as zero.<br><br>0 Speculation buffer and single entry buffer are not affected.<br>1 Invalidate (clear) speculation buffer and single entry buffer. |
| 18–17<br>B0MW[1:0] | Bank 0 Memory Width<br><br>This read-only field defines the width of the bank 0 memory.<br><br>00 32 bits<br>01 64 bits<br>10 128 bits<br>11 Reserved |
| 16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7–5<br>CRC[2:0] | Cache Replacement Control<br><br>This 3-bit field defines the replacement algorithm for accesses that are cached.<br><br>000 LRU replacement algorithm per set across all four ways<br>001 Reserved<br>010 Independent LRU with ways [0-1] for ifetches, [2-3] for data<br>011 Independent LRU with ways [0-2] for ifetches, [3] for data<br>1xx Reserved |
| 4<br>B0DCE | Bank 0 Data Cache Enable<br><br>This bit controls whether data references are loaded into the cache.<br><br>0 Do not cache data references.<br>1 Cache data references. |
| 3<br>B0ICE | Bank 0 Instruction Cache Enable<br><br>This bit controls whether instruction fetches are loaded into the cache. |

*Table continues on the next page...*

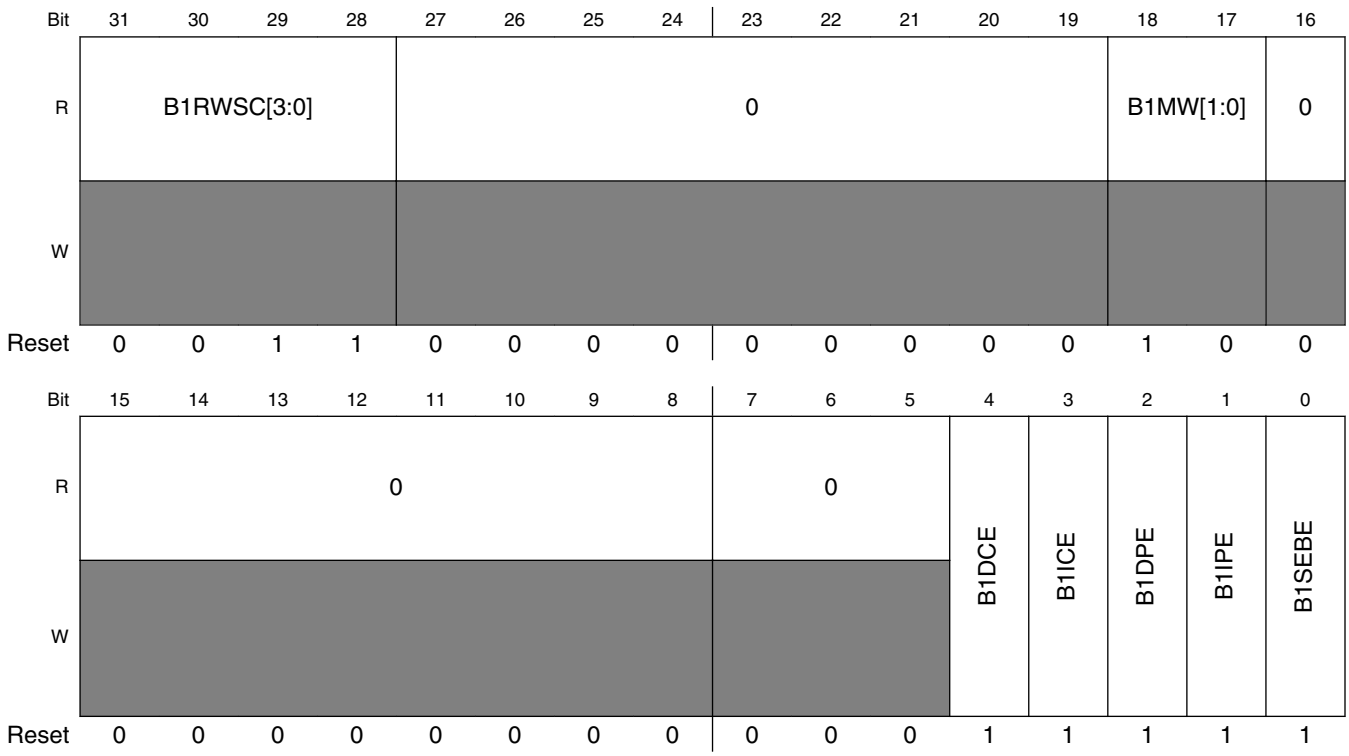
## FMC_PFB0CR field descriptions (continued)

| Field | Description |
|---|---|
| | 0 Do not cache instruction fetches.<br>1 Cache instruction fetches. |
| 2<br>B0DPE | Bank 0 Data Prefetch Enable<br>This bit controls whether prefetches (or speculative accesses) are initiated in response to data references.<br>0 Do not prefetch in response to data references.<br>1 Enable prefetches in response to data references. |
| 1<br>B0IPE | Bank 0 Instruction Prefetch Enable<br>This bit controls whether prefetches (or speculative accesses) are initiated in response to instruction fetches.<br>0 Do not prefetch in response to instruction fetches.<br>1 Enable prefetches in response to instruction fetches. |
| 0<br>B0SEBE | Bank 0 Single Entry Buffer Enable<br>This bit controls whether the single entry page buffer is enabled in response to flash read accesses. Its operation is independent from bank 1's cache.<br>A high-to-low transition of this enable forces the page buffer to be invalidated.<br>0 Single entry buffer is disabled.<br>1 Single entry buffer is enabled. |

## 28.4.3 Flash Bank 1 Control Register (FMC_PFB1CR)

This register has a format similar to that for PFB0CR, except it controls the operation of flash bank 1, and the "global" cache control fields are empty.

Address: 4001_F000h base + 8h offset = 4001_F008h




**FMC_PFB1CR field descriptions**

| Field | Description |
|---|---|
| 31–28<br>B1RWSC[3:0] | Bank 1 Read Wait State Control<br>This read-only field defines the number of wait states required to access the bank 1 flash memory.<br>The relationship between the read access time of the flash array (expressed in system clock cycles) and RWSC is defined as:<br>Access time of flash array [system clocks] = RWSC + 1<br>The FMC automatically calculates this value based on the ratio of the system clock speed to the flash clock speed. For example, when this ratio is 4:1, the field's value is 3h. |
| 27–19<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 18–17<br>B1MW[1:0] | Bank 1 Memory Width<br>This read-only field defines the width of the bank 1 memory. |

*Table continues on the next page...*

## FMC_PFB1CR field descriptions (continued)

| Field | Description |
|---|---|
| | 00 32 bits<br>01 64 bits<br>10 128 bits<br>11 Reserved |
| 16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>B1DCE | Bank 1 Data Cache Enable<br><br>This bit controls whether data references are loaded into the cache.<br><br>0 Do not cache data references.<br>1 Cache data references. |
| 3<br>B1ICE | Bank 1 Instruction Cache Enable<br><br>This bit controls whether instruction fetches are loaded into the cache.<br><br>0 Do not cache instruction fetches.<br>1 Cache instruction fetches. |
| 2<br>B1DPE | Bank 1 Data Prefetch Enable<br><br>This bit controls whether prefetches (or speculative accesses) are initiated in response to data references.<br><br>0 Do not prefetch in response to data references.<br>1 Enable prefetches in response to data references. |
| 1<br>B1IPE | Bank 1 Instruction Prefetch Enable<br><br>This bit controls whether prefetches (or speculative accesses) are initiated in response to instruction fetches.<br><br>0 Do not prefetch in response to instruction fetches.<br>1 Enable prefetches in response to instruction fetches. |
| 0<br>B1SEBE | Bank 1 Single Entry Buffer Enable<br><br>This bit controls whether the single entry buffer is enabled in response to flash read accesses. Its operation is independent from bank 0's cache.<br><br>A high-to-low transition of this enable forces the page buffer to be invalidated.<br><br>0 Single entry buffer is disabled.<br>1 Single entry buffer is enabled. |

## 28.4.4 Cache Tag Storage (FMC_TAGVDW0S*n*)

The 128-entry cache is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In TAGVDWxSy, x denotes the way, and y denotes the set. This section represents tag/vld information for all 3 sets (n=0-3) in way 0.

Address: 4001_F000h base + 100h offset + (4d × i), where i=0d to 3d

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | tag[19:6] | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | tag[19:6] | | | | | | | | | | 0 | | | | | valid |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_TAGVDW0S*n* field descriptions**

| Field | Description |
|---|---|
| 31–20<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 19–6<br>tag[19:6] | 14-bit tag for cache entry |
| 5–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>valid | 1-bit valid for cache entry |

## 28.4.5 Cache Tag Storage (FMC_TAGVDW1S*n*)

The 128-entry cache is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In TAGVDWxSy, x denotes the way, and y denotes the set. This section represents tag/vld information for all 3 sets (n=0-3) in way 1.

Address: 4001_F000h base + 110h offset + (4d × i), where i=0d to 3d

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | tag[19:6] | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | tag[19:6] | | | | | | | | | | 0 | | | | | valid |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_TAGVDW1S*n* field descriptions**

| Field | Description |
|---|---|
| 31–20<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 19–6<br>tag[19:6] | 14-bit tag for cache entry |
| 5–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>valid | 1-bit valid for cache entry |

## 28.4.6 Cache Tag Storage (FMC_TAGVDW2S*n*)

The 128-entry cache is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In TAGVDWxSy, x denotes the way, and y denotes the set. This section represents tag/vld information for all 3 sets (n=0-3) in way 2.

Address: 4001_F000h base + 120h offset + (4d × i), where i=0d to 3d

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | tag[19:6] | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | tag[19:6] | | | | | | | | | | 0 | | | | | valid |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_TAGVDW2S*n* field descriptions**

| Field | Description |
|---|---|
| 31–20<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 19–6<br>tag[19:6] | 14-bit tag for cache entry |
| 5–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>valid | 1-bit valid for cache entry |

## 28.4.7 Cache Tag Storage (FMC_TAGVDW3S*n*)

The 128-entry cache is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In TAGVDWxSy, x denotes the way, and y denotes the set. This section represents tag/vld information for all 3 sets (n=0-3) in way 3.

Address: 4001_F000h base + 130h offset + (4d × i), where i=0d to 3d

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | tag[19:6] | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | tag[19:6] | | | | | | | | | | 0 | | | | | valid |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_TAGVDW3S*n* field descriptions**

| Field | Description |
|---|---|
| 31–20<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 19–6<br>tag[19:6] | 14-bit tag for cache entry |
| 5–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>valid | 1-bit valid for cache entry |

## 28.4.8 Cache Data Storage (uppermost word) (FMC_DATAW0S*n*UM)

The cache of sixteen 128-bit entries is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In DATAWxSyUM, DATAWxSyMU, DATAWxSyML, and DATAWxSyLM, x denotes the way, y denotes the set, and the final two letters identify the word: UM (uppermost), MU (mid-upper), ML (mid-lower), and LM (lowermost). This section represents data for the uppermost word (bits [127:96]) of all 4 sets (n=0-3) in way 0.

Address: 4001_F000h base + 200h offset + (16d × i), where i=0d to 3d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | data[127:96] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_DATAW0S*n*UM field descriptions**

| Field | Description |
|---|---|
| 31–0<br>data[127:96] | Bits [127:96] of data entry |

## 28.4.9 Cache Data Storage (mid-upper word) (FMC_DATAW0S*n*MU)

The cache of sixteen 128-bit entries is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In DATAWxSyUM, DATAWxSyMU, DATAWxSyML, and DATAWxSyLM, x denotes the way, y denotes the set, and the final two letters identify the word: UM (uppermost), MU (mid-upper), ML (mid-lower), and LM (lowermost). This section represents data for the mid-upper word (bits [95:64]) of all 4 sets (n=0-3) in way 0.

Address: 4001_F000h base + 204h offset + (16d × i), where i=0d to 3d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | data[95:64] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_DATAW0S*n*MU field descriptions**

| Field | Description |
|---|---|
| 31–0<br>data[95:64] | Bits [95:64] of data entry |

## 28.4.10 Cache Data Storage (mid-lower word) (FMC_DATAW0S*n*ML)

The cache of sixteen 128-bit entries is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In DATAWxSyUM, DATAWxSyMU, DATAWxSyML, and DATAWxSyLM, x denotes the way, y denotes the set, and the final two letters identify the word: UM (uppermost), MU (mid-upper), ML (mid-lower), and LM (lowermost). This section represents data for the mid-lower word (bits [63:32]) of all 4 sets (n=0-3) in way 0.

Address: 4001_F000h base + 208h offset + (16d × i), where i=0d to 3d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | data[63:32] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_DATAW0S*n*ML field descriptions**

| Field | Description |
|---|---|
| 31–0<br>data[63:32] | Bits [63:32] of data entry |

## 28.4.11 Cache Data Storage (lowermost word) (FMC_DATAW0S*n*LM)

The cache of sixteen 128-bit entries is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In DATAWxSyUM, DATAWxSyMU, DATAWxSyML, and DATAWxSyLM, x denotes the way, y denotes the set, and the final two letters identify the word: UM (uppermost), MU (mid-upper), ML (mid-lower), and LM (lowermost). This section represents data for the lowermost word (bits [31:0]) of all 4 sets (n=0-3) in way 0.

Address: 4001_F000h base + 20Ch offset + (16d × i), where i=0d to 3d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | data[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_DATAW0S*n*LM field descriptions**

| Field | Description |
|---|---|
| 31–0<br>data[31:0] | Bits [31:0] of data entry |

## 28.4.12 Cache Data Storage (uppermost word) (FMC_DATAW1S*n*UM)

The cache of sixteen 128-bit entries is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In DATAWxSyUM, DATAWxSyMU, DATAWxSyML, and DATAWxSyLM, x denotes the way, y denotes the set, and the final two letters identify the word: UM (uppermost), MU (mid-upper), ML (mid-lower), and LM (lowermost). This section represents data for the uppermost word (bits [127:96]) of all 4 sets (n=0-3) in way 0.

Address: 4001_F000h base + 240h offset + (16d × i), where i=0d to 3d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | data[127:96] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_DATAW1S*n*UM field descriptions**

| Field | Description |
|---|---|
| 31–0<br>data[127:96] | Bits [127:96] of data entry |

## 28.4.13 Cache Data Storage (mid-upper word) (FMC_DATAW1S*n*MU)

The cache of sixteen 128-bit entries is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In DATAWxSyUM, DATAWxSyMU, DATAWxSyML, and DATAWxSyLM, x denotes the way, y denotes the set, and the final two letters identify the word: UM (uppermost), MU (mid-upper), ML (mid-lower), and LM (lowermost). This section represents data for the mid-upper word (bits [95:64]) of all 4 sets (n=0-3) in way 0.

Address: 4001_F000h base + 244h offset + (16d × i), where i=0d to 3d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | data[95:64] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_DATAW1S*n*MU field descriptions**

| Field | Description |
|---|---|
| 31–0<br>data[95:64] | Bits [95:64] of data entry |

## 28.4.14 Cache Data Storage (mid-lower word) (FMC_DATAW1S*n*ML)

The cache of sixteen 128-bit entries is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In DATAWxSyUM, DATAWxSyMU, DATAWxSyML, and DATAWxSyLM, x denotes the way, y denotes the set, and the final two letters identify the word: UM (uppermost), MU (mid-upper), ML (mid-lower), and LM (lowermost). This section represents data for the mid-lower word (bits [63:32]) of all 4 sets (n=0-3) in way 0.

Address: 4001_F000h base + 248h offset + (16d × i), where i=0d to 3d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | data[63:32] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_DATAW1S*n*ML field descriptions**

| Field | Description |
|---|---|
| 31–0<br>data[63:32] | Bits [63:32] of data entry |

## 28.4.15 Cache Data Storage (lowermost word) (FMC_DATAW1S*n*LM)

The cache of sixteen 128-bit entries is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In DATAWxSyUM, DATAWxSyMU, DATAWxSyML, and DATAWxSyLM, x denotes the way, y denotes the set, and the final two letters identify the word: UM (uppermost), MU (mid-upper), ML (mid-lower), and LM (lowermost). This section represents data for the lowermost word (bits [31:0]) of all 4 sets (n=0-3) in way 0.

Address: 4001_F000h base + 24Ch offset + (16d × i), where i=0d to 3d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | data[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_DATAW1S*n*LM field descriptions**

| Field | Description |
|---|---|
| 31–0<br>data[31:0] | Bits [31:0] of data entry |

## 28.4.16 Cache Data Storage (uppermost word) (FMC_DATAW2S*n*UM)

The cache of sixteen 128-bit entries is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In DATAWxSyUM, DATAWxSyMU, DATAWxSyML, and DATAWxSyLM, x denotes the way, y denotes the set, and the final two letters identify the word: UM (uppermost), MU (mid-upper), ML (mid-lower), and LM (lowermost). This section represents data for the uppermost word (bits [127:96]) of all 4 sets (n=0-3) in way 0.

Address: 4001_F000h base + 280h offset + (16d × i), where i=0d to 3d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | data[127:96] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_DATAW2S*n*UM field descriptions**

| Field | Description |
|---|---|
| 31–0<br>data[127:96] | Bits [127:96] of data entry |

## 28.4.17 Cache Data Storage (mid-upper word) (FMC_DATAW2S*n*MU)

The cache of sixteen 128-bit entries is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In DATAWxSyUM, DATAWxSyMU, DATAWxSyML, and DATAWxSyLM, x denotes the way, y denotes the set, and the final two letters identify the word: UM (uppermost), MU (mid-upper), ML (mid-lower), and LM (lowermost). This section represents data for the mid-upper word (bits [95:64]) of all 4 sets (n=0-3) in way 0.

Address: 4001_F000h base + 284h offset + (16d × i), where i=0d to 3d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | data[95:64] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_DATAW2S*n*MU field descriptions**

| Field | Description |
|---|---|
| 31–0<br>data[95:64] | Bits [95:64] of data entry |

## 28.4.18 Cache Data Storage (mid-lower word) (FMC_DATAW2S*n*ML)

The cache of sixteen 128-bit entries is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In DATAWxSyUM, DATAWxSyMU, DATAWxSyML, and DATAWxSyLM, x denotes the way, y denotes the set, and the final two letters identify the word: UM (uppermost), MU (mid-upper), ML (mid-lower), and LM (lowermost). This section represents data for the mid-lower word (bits [63:32]) of all 4 sets (n=0-3) in way 0.

Address: 4001_F000h base + 288h offset + (16d × i), where i=0d to 3d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | data[63:32] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_DATAW2S*n*ML field descriptions**

| Field | Description |
|---|---|
| 31–0<br>data[63:32] | Bits [63:32] of data entry |

## 28.4.19 Cache Data Storage (lowermost word) (FMC_DATAW2S*n*LM)

The cache of sixteen 128-bit entries is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In DATAWxSyUM, DATAWxSyMU, DATAWxSyML, and DATAWxSyLM, x denotes the way, y denotes the set, and the final two letters identify the word: UM (uppermost), MU (mid-upper), ML (mid-lower), and LM (lowermost). This section represents data for the lowermost word (bits [31:0]) of all 4 sets (n=0-3) in way 0.

Address: 4001_F000h base + 28Ch offset + (16d × i), where i=0d to 3d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | data[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_DATAW2S*n*LM field descriptions**

| Field | Description |
|---|---|
| 31–0<br>data[31:0] | Bits [31:0] of data entry |

## 28.4.20 Cache Data Storage (uppermost word) (FMC_DATAW3S*n*UM)

The cache of sixteen 128-bit entries is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In DATAWxSyUM, DATAWxSyMU, DATAWxSyML, and DATAWxSyLM, x denotes the way, y denotes the set, and the final two letters identify the word: UM (uppermost), MU (mid-upper), ML (mid-lower), and LM (lowermost). This section represents data for the uppermost word (bits [127:96]) of all 4 sets (n=0-3) in way 0.

Address: 4001_F000h base + 2C0h offset + (16d × i), where i=0d to 3d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | data[127:96] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_DATAW3S*n*UM field descriptions**

| Field | Description |
|---|---|
| 31–0<br>data[127:96] | Bits [127:96] of data entry |

## 28.4.21 Cache Data Storage (mid-upper word) (FMC_DATAW3S*n*MU)

The cache of sixteen 128-bit entries is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In DATAWxSyUM, DATAWxSyMU, DATAWxSyML, and DATAWxSyLM, x denotes the way, y denotes the set, and the final two letters identify the word: UM (uppermost), MU (mid-upper), ML (mid-lower), and LM (lowermost). This section represents data for the mid-upper word (bits [95:64]) of all 4 sets (n=0-3) in way 0.

Address: 4001_F000h base + 2C4h offset + (16d × i), where i=0d to 3d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | data[95:64] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_DATAW3S*n*MU field descriptions**

| Field | Description |
|---|---|
| 31–0<br>data[95:64] | Bits [95:64] of data entry |

## 28.4.22 Cache Data Storage (mid-lower word) (FMC_DATAW3S*n*ML)

The cache of sixteen 128-bit entries is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In DATAWxSyUM, DATAWxSyMU, DATAWxSyML, and DATAWxSyLM, x denotes the way, y denotes the set, and the final two letters identify the word: UM (uppermost), MU (mid-upper), ML (mid-lower), and LM (lowermost). This section represents data for the mid-lower word (bits [63:32]) of all 4 sets (n=0-3) in way 0.

Address: 4001_F000h base + 2C8h offset + (16d × i), where i=0d to 3d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | data[63:32] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_DATAW3S*n*ML field descriptions**

| Field | Description |
|---|---|
| 31–0<br>data[63:32] | Bits [63:32] of data entry |

## 28.4.23 Cache Data Storage (lowermost word) (FMC_DATAW3S*n*LM)

The cache of sixteen 128-bit entries is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In DATAWxSyUM, DATAWxSyMU, DATAWxSyML, and DATAWxSyLM, x denotes the way, y denotes the set, and the final two letters identify the word: UM (uppermost), MU (mid-upper), ML (mid-lower), and LM (lowermost). This section represents data for the lowermost word (bits [31:0]) of all 4 sets (n=0-3) in way 0.

Address: 4001_F000h base + 2CCh offset + (16d × i), where i=0d to 3d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | data[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_DATAW3S*n*LM field descriptions**

| Field | Description |
|---|---|
| 31–0<br>data[31:0] | Bits [31:0] of data entry |

## 28.5 Functional description

The FMC is a flash acceleration unit with flexible buffers for user configuration. Besides managing the interface between the device and the flash memory and FlexMemory, the FMC can be used to restrict access from crossbar switch masters and customize the cache and buffers to provide single-cycle system-clock data-access times. Whenever a hit occurs for the prefetch speculation buffer, the cache, or the single-entry buffer, the requested data is transferred within a single system clock.

### 28.5.1 Default configuration

Upon system reset, the FMC is configured to provide a significant level of buffering for transfers from the flash memory or FlexMemory:

- Crossbar masters 0, 1, 2 have read access to bank 0 and bank 1.
- These masters have write access to a portion of bank 1 when FlexNVM is used with FlexRAM as EEPROM.
- For bank 0 and bank 1:
  - Prefetch support for data and instructions is enabled for crossbar masters 0, 1, 2.
  - The cache is configured for least recently used (LRU) replacement for all four ways.
  - The cache is configured for data or instruction replacement.
  - The single-entry buffer is enabled.

### 28.5.2 Configuration options

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

The FMC's cache and buffering controls within PFB0CR and PFB1CR allow the tuning of resources to suit particular applications' needs. The cache and two buffers are each controlled individually. The register controls enable buffering and prefetching per memory bank and access type (instruction fetch or data reference). The cache also supports three types of LRU replacement algorithms:

- LRU per set across all four ways,

- LRU with ways [0-1] for instruction fetches and ways [2-3] for data fetches, and
- LRU with ways [0-2] for instruction fetches and way [3] for data fetches.

As an application example: if both instruction fetches and data references are accessing bank 0, control is available to send instruction fetches, data references, or both to the cache or the single-entry buffer. Likewise, speculation can be enabled or disabled for either type of access. If both instruction fetches and data references are cached, the cache's way resources may be divided in several ways between the instruction fetches and data references.

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

### 28.5.3 Wait states

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

1. the ratio of the core clock to the flash clock, and
2. the phase relationship of the core clock and flash clock at the time the read is requested.

The ratio of the core clock to the flash clock is equal to the value of PFB0CR[B0RWSC] + 1 for bank 0 and to the value of PFB1CR[B1RWSC] + 1 for bank 1.

For example, in a system with a 4:1 core-to-flash clock ratio, a read that does not hit in the speculation buffer or the cache can take between 4 and 7 core clock cycles to complete.

- The best-case scenario is a period of 4 core clock cycles because a read from the flash memory takes 1 flash clock, which translates to 4 core clocks.
- The worst-case scenario is a period of 7 core clock cycles, consisting of 4 cycles for the read operation and 3 cycles of delay to align the core and flash clocks.
  - A delay to align the core and flash clocks might occur because you can request a read cycle on any core clock edge, but that edge does not necessarily align with a flash clock edge where the read can start.
  - In this case, the read operation is delayed by a number of core clocks equal to the core-to-flash clock ratio minus one: 4 - 1 = 3. That is, 3 additional core clock cycles are required to synchronize the clocks before the read operation can start.

All wait states and synchronization delays are handled automatically by the Flash Memory Controller. No direct user configuration is required or even allowed to set up the flash wait states.

### 28.5.4 Speculative reads

The FMC has a single buffer that reads ahead to the next word in the flash memory if there is an idle cycle. Speculative prefetching is programmable for each bank for instruction and/or data accesses using the B0DPE and B0IPE fields of PFB0CR and the B1DPE and B1IPE fields of PFB1CR. Because many code accesses are sequential, using the speculative prefetch buffer improves performance in most cases.

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

For example, consider the following scenario:

- Assume a system with a 4:1 core-to-flash clock ratio and with speculative reads enabled.
- The core requests eight sequential longwords in back-to-back requests, meaning there are no core cycle delays except for stalls waiting for flash memory data to be returned.
- None of the data is already stored in the cache or speculation buffer.

In this scenario, the sequence of events for accessing the eight longwords is as follows:

1. The first longword read requires 4 to 7 core clocks. See Wait states for more information.
2. Due to the 128-bit data bus of the flash memory, the second longword read takes only 1 core clock because the data is already available inside the FMC. For the same reason, the third and fourth longword reads each take only 1 core clock.
3. Accessing the fifth longword requires 1 core clock cycle. The flash memory read itself takes 4 clocks, but the access starts immediately after the first read. As a result, 3 clocks for this access overlap with the second, third, and fourth longword reads from the core.
4. Reading the sixth, seventh, and eighth longwords takes only 1 clock each because the data is already available inside the FMC.

## 28.6 Initialization and application information

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

The FMC has no visibility into flash memory erase and program cycles because the Flash Memory module manages them directly. As a result, if an application is executing flash memory commands, the FMC's cache might need to be disabled and/or flushed to prevent the possibility of returning stale data. Use the PFB0CR[CINV_WAY] field to invalidate the cache in this manner.

# Chapter 29
# Flash Memory Module (FTFE)

## 29.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The FTFE module includes the following accessible memory regions:

- Program flash memory for vector space and code store
- For FlexNVM devices: FlexNVM for data store and additional code store
- For FlexNVM devices: FlexRAM for high-endurance data store or traditional RAM
- For program flash only devices: Programming acceleration RAM to speed flash programming

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

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

**CAUTION**

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

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

## 29.1.1 Features

The FTFE module includes the following features.

**NOTE**

See the device's Chip Configuration details for the exact amount of flash memory available on your device.

### 29.1.1.1 Program Flash Memory Features

- Sector size of 4 Kbytes
- Program flash protection scheme prevents accidental program or erase of stored data
- Automated, built-in, program and erase algorithms with verify
- Section programming for faster bulk programming times
- For devices containing only program flash memory: Read access to one program flash block is possible while programming or erasing data in another program flash block
- For devices containing FlexNVM memory: Read access to one program flash block is possible while programming or erasing data in another program flash block, data flash block, or FlexRAM

### 29.1.1.2 FlexNVM memory features

When FlexNVM is partitioned for data flash memory (on devices that contain FlexNVM memory):

- Sector size of 4 Kbytes
- Protection scheme prevents accidental program or erase of stored data

- Automated, built-in program and erase algorithms with verify
- Section programming for faster bulk programming times
- Read access to the data flash block possible while programming or erasing data in the program flash block

#### 29.1.1.3 Programming Acceleration RAM features

- For devices with only program flash memory: RAM to support section programming

#### 29.1.1.4 FlexRAM features

For devices with FlexNVM memory:

- Memory that can be used as traditional RAM or as high-endurance EEPROM storage
- Up to 4 Kbytes of FlexRAM configured for EEPROM or traditional RAM operations
- When configured for EEPROM:
  - Protection scheme prevents accidental program or erase of data written for EEPROM
  - Built-in hardware emulation scheme to automate EEPROM record maintenance functions
  - Programmable EEPROM data set size and FlexNVM partition code facilitating EEPROM memory endurance trade-offs
  - Supports FlexRAM aligned writes of 1, 2, or 4 bytes at a time
  - Read access to FlexRAM possible while programming or erasing data in the program or data flash memory
- When configured for traditional RAM:
  - Read and write access possible to the FlexRAM while programming or erasing data in the program or data flash memory

#### 29.1.1.5 Other FTFE module features

- Internal high-voltage supply generator for flash memory program and erase operations

- Optional interrupt generation upon flash command completion
- Supports MCU security mechanisms which prevent unauthorized access to the flash memory contents

## 29.1.2 Block diagram

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

For devices with FlexNVM feature:

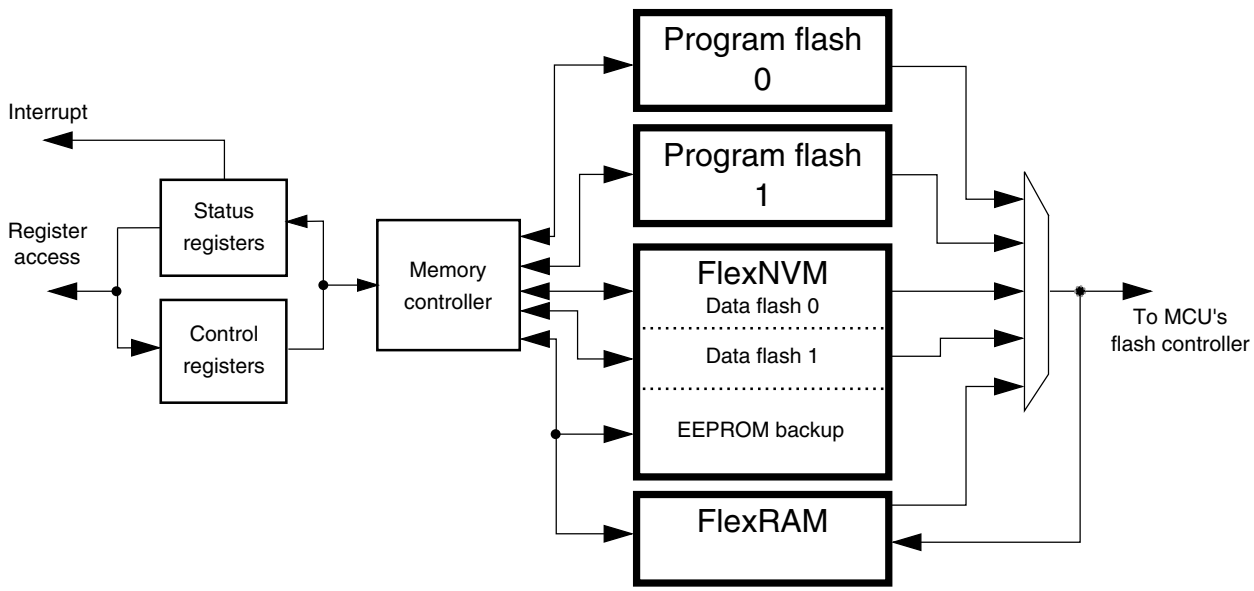



**Figure 29-1. FTFE block diagram**

For devices that contain only program flash:

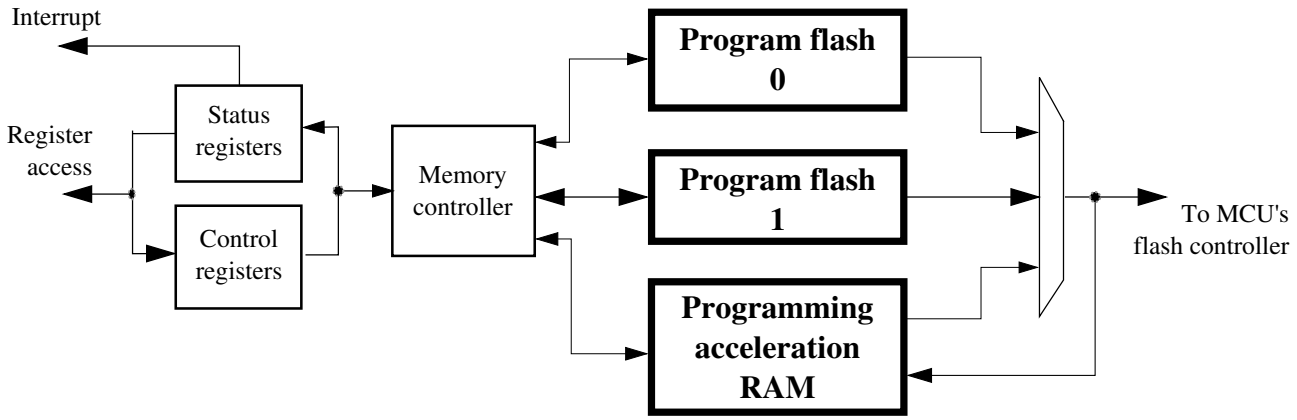



**Figure 29-2. FTFE block diagram**

## 29.1.3 Glossary

**Command write sequence** — A series of MCU writes to the Flash FCCOB register group that initiates and controls the execution of Flash algorithms that are built into the FTFE module.

**Data flash memory** — Partitioned from the FlexNVM block, the data flash memory provides nonvolatile storage for user data, boot code, and additional code store.

**Data flash sector** — The data flash sector is the smallest portion of the data flash memory that can be erased.

**EEPROM** — Using a built-in filing system, the FTFE module emulates the characteristics of an EEPROM by effectively providing a high-endurance, byte-writeable (program and erase) NVM.

**EEPROM backup data header** — The EEPROM backup data header is comprised of a 64-bit field found in EEPROM backup data memory which contains information used by the EEPROM filing system to determine the status of a specific EEPROM backup flash sector.

**EEPROM backup data record** — The EEPROM backup data record is comprised of a 7-bit status field, a 13-bit address field, and a 32-bit data field found in EEPROM backup data memory which is used by the EEPROM filing system. If the status field indicates a record is valid, the data field is mirrored in the FlexRAM at a location determined by the address field.

**EEPROM backup data memory** — Partitioned from the FlexNVM block, EEPROM backup data memory provides nonvolatile storage for the EEPROM filing system representing data written to the FlexRAM requiring highest endurance.

**EEPROM backup data sector** — The EEPROM backup data sector contains one EEPROM header and up to 255 EEPROM backup data records, which are used by the EEPROM filing system.

**Endurance** — The number of times that a flash memory location can be erased and reprogrammed.

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

**Flash block** — A macro within the FTFE module which provides the nonvolatile memory storage.

**FlexMemory** — FTFE configuration that supports data flash, EEPROM, and FlexRAM.

**FlexNVM Block** — The FlexNVM block can be configured to be used as data flash memory, EEPROM backup flash memory, or a combination of both.

**FlexRAM** — The FlexRAM refers to a RAM, dedicated to the FTFE module, that can be configured to store EEPROM data or as traditional RAM. When configured for EEPROM, valid writes to the FlexRAM generates a new EEPROM backup data record stored in the EEPROM backup flash memory.

**FTFE Module** — All flash blocks plus a flash management unit providing high-level control and an interface to MCU buses.

**IFR** — Nonvolatile information register found in each flash block, separate from the main memory array.

**NVM** — Nonvolatile memory. A memory technology that maintains stored data during power-off. The flash array is an NVM using NOR-type flash memory technology.

**NVM Normal Mode** — An NVM mode that provides basic user access to FTFE resources. The CPU or other bus masters initiate flash program and erase operations (or other flash commands) using writes to the FCCOB register group in the FTFE module.

**NVM Special Mode** — An NVM mode enabling external, off-chip access to the memory resources in the FTFE module. A reduced flash command set is available when the MCU is secured. See the Chip Configuration details for information on when this mode is used.

**Double-Phrase** — 128 bits of data with an aligned double-phrase having byte-address[3:0] = 0000.

**Phrase** — 64 bits of data with an aligned phrase having byte-address[2:0] = 000.

**Longword** — 32 bits of data with an aligned longword having byte-address[1:0] = 00.

**Word** — 16 bits of data with an aligned word having byte-address[0] = 0.

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

**Program flash sector** — The smallest portion of the program flash memory (consecutive addresses) that can be erased.

**Retention** — The length of time that data can be kept in the NVM without experiencing errors upon readout. Since erased (1) states are subject to degradation just like programmed (0) states, the data retention limit may be reached from the last erase operation (not from the programming time).

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

**Section program buffer** — Lower quarter of the programming acceleration FlexRAM allocated for storing large amounts of data for programming via the Program Section command.

**Secure** — An MCU state conveyed to the FTFE module as described in the Chip Configuration details for this device. In the secure state, reading and changing NVM contents is restricted.

## 29.2 External signal description

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

## 29.3 Memory map and registers

This section describes the memory map and registers for the FTFE module. Data read from unimplemented memory space in the FTFE module is undefined. Writes to unimplemented or reserved memory space (registers) in the FTFE module are ignored.

### 29.3.1 Flash configuration field description

The program flash memory contains a 16-byte flash configuration field that stores default protection settings (loaded on reset) and security information that allows the MCU to restrict access to the FTFE module.

| Flash Configuration Field Byte Address | Size (Bytes) | Field Description |
|---|---|---|
| 0x0_0400 - 0x0_0407 | 8 | Backdoor Comparison Key. Refer to Verify Backdoor Access Key command and Unsecuring the MCU Using Backdoor Key Access. |
| 0x0_0408 - 0x0_040B | 4 | Program flash protection bytes. Refer to the description of the Program Flash Protection Registers (FPROT0-3). |
| 0x0_040F | 1 | Program flash only devices: Reserved<br>FlexNVM devices: Data flash protection byte. Refer to the description of the Data Flash Protection Register (FDPROT). |
| 0x0_040E | 1 | Program flash only devices: Reserved<br>FlexNVM devices: EEPROM protection byte. Refer to the description of the EEPROM Protection Register (FEPROT). |
| 0x0_040D | 1 | Flash nonvolatile option byte. Refer to the description of the Flash Option Register (FOPT). |
| 0x0_040C | 1 | Flash security byte. Refer to the description of the Flash Security Register (FSEC). |

### 29.3.2 Program flash 0 IFR map

The program flash 0 IFR is a 1 Kbyte nonvolatile information memory that can be read freely, but the user has no erase and limited program capabilities (see the Read Once, Program Once, and Read Resource commands in Read Once Command, Program Once command and Read Resource Command). The contents of the program flash 0 IFR are summarized in the following table and further described in the subsequent paragraphs.

The program flash 0 IFR is located within the program flash 0 memory block.

| Address Range | Size (Bytes) | Field Description |
|---|---|---|
| 0x000 – 0x3BF | 960 | Reserved |
| 0x3C0 – 03xFF | 64 | Program Once Field |

#### 29.3.2.1 Program Once field

The Program Once field in the program flash 0 IFR provides 64 bytes of user data storage separate from the program flash 0 main array. The user can program the Program Once field one time only as there is no program flash IFR erase mechanism available to the user. The Program Once field can be read any number of times. This section of the program flash 0 IFR is accessed in 8 byte records using the Read Once and Program Once commands (see Read Once Command and Program Once command).

### 29.3.3 Data flash 0 IFR map

The following only applies to devices with FlexNVM.

The data flash 0 IFR is a 1 Kbyte nonvolatile information memory that can be read and erased, but the user has limited program capabilities in the data flash 0 IFR (see the Program Partition command in Program Partition command, the Erase All Blocks command in Erase All Blocks Command, and the Read Resource command in Read Resource Command). The contents of the data flash 0 IFR are summarized in the following table and further described in the subsequent paragraphs.

The data flash 0 IFR is located within the data flash 0 memory block.

| Address Range | Size (Bytes) | Field Description |
|---|---|---|
| 0x00 – 0x3FB, 0x3FE – 0x3FF | 1022 | Reserved |
| 0x3FD | 1 | EEPROM Data Set Size |
| 0x3FC | 1 | FlexNVM Partition Code |

#### 29.3.3.1 EEPROM Data Set Size

The EEPROM data set size byte in the data flash 0 IFR supplies information which determines the amount of FlexRAM used in each of the available EEPROM subsystems. To program the EEESPLIT and EEESIZE values, see the Program Partition command described in Program Partition command.

**Table 29-1. EEPROM Data Set Size**

| Data flash IFR: 0x03FD | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 1 | EEESPLIT | | EEESIZE | | | |
| | = Unimplemented or Reserved | | | | | | |

**Table 29-2. EEPROM Data Set Size Field Description**

| Field | Description |
|---|---|
| 7-6<br>Reserved | This read-only bitfield is reserved and must always be written as one. |
| 5-4<br>EEESPLIT | **EEPROM Split Factor** — Determines the relative sizes of the two EEPROM subsystems. Each subsystem is allocated half of the available EEPROM-backup as defined by DEPART.<br>‘00’ = Subsystem A: EEESIZE*1/8, subsystem B: EEESIZE*7/8<br>‘01’ = Subsystem A: EEESIZE*1/4, subsystem B: EEESIZE*3/4<br>‘10’ = Subsystem A: EEESIZE*1/2, subsystem B: EEESIZE*1/2<br>‘11’ = Subsystem A: EEESIZE*1/2, subsystem B: EEESIZE*1/2 |
| 3-0<br>EEESIZE | **EEPROM Size** — Encoding of the total available FlexRAM for EEPROM use.<br>**NOTE:** EEESIZE must be 0 bytes (1111b) when the FlexNVM partition code (FlexNVM partition code) is set to 'No EEPROM'.<br>'0000' = Reserved<br>'0001' = Reserved<br>'0010' = 4,096 Bytes<br>'0011' = 2,048 Bytes<br>'0100' = 1,024 Bytes<br>'0101' = 512 Bytes<br>'0110' = 256 Bytes<br>'0111' = 128 Bytes<br>'1000' = 64 Bytes<br>'1001' = 32 Bytes<br>'1010' = Reserved<br>'1011' = Reserved<br>'1100' = Reserved<br>'1101' = Reserved<br>'1110' = Reserved<br>'1111' = 0 Bytes |

### 29.3.3.2 FlexNVM partition code

The FlexNVM partition code byte in the data flash 0 IFR supplies a code which specifies how to split the FlexNVM block between data flash memory and EEPROM backup memory supporting EEPROM functions. To program the DEPART value, see the Program Partition command described in Program Partition command.

**Table 29-3. FlexNVM partition code**

| Data Flash IFR: 0x03FC | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 1 | 1 | 1 | DEPART | | | |
| | = Unimplemented or Reserved | | | | | | |

**Table 29-4. FlexNVM partition code field description**

| Field | Description |
|---|---|
| 7-4<br>Reserved | This read-only bitfield is reserved and must always be written as one. |
| 3-0<br>DEPART | **FlexNVM Partition Code** — Encoding of the data flash / EEPROM backup split within the FlexNVM memory block. FlexNVM memory not partitioned for data flash is used to store EEPROM records. |

| DEPART | Data flash (KByte) | EEPROM backup (KByte) |
|---|---|---|
| 0000 | 128 | 0 |
| 0001 | Reserved | Reserved |
| 0010 | Reserved | Reserved |
| 0011 | 96 | 32 |
| 0100 | 64 | 64 |
| 0101 | 0 | 128 |
| 0110 | Reserved | Reserved |
| 0111 | Reserved | Reserved |
| 1000 | 0 | 128 |
| 1001 | Reserved | Reserved |
| 1010 | Reserved | Reserved |
| 1011 | 32 | 96 |
| 1100 | 64 | 64 |
| 1101 | 128 | 0 |
| 1110 | Reserved | Reserved |
| 1111 | 128 | 0 |

## 29.3.4 Register descriptions

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

**NOTE**

While a command is running (FSTAT[CCIF]=0), register writes are not accepted to any register except FCNFG and FSTAT. The no-write rule is relaxed during the start-up reset

sequence, prior to the initial rise of CCIF. During this initialization period the user may write any register. All register writes are also disabled (except for registers FCNFG and FSTAT) whenever an erase suspend request is active (FCNFG[ERSSUSP]=1).

### FTFE memory map

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4002_0000 | Flash Status Register (FTFE_FSTAT) | 8 | R/W | 00h | 29.34.1/ 631 |
| 4002_0001 | Flash Configuration Register (FTFE_FCNFG) | 8 | R/W | 00h | 29.34.2/ 632 |
| 4002_0002 | Flash Security Register (FTFE_FSEC) | 8 | R | Undefined | 29.34.3/ 635 |
| 4002_0003 | Flash Option Register (FTFE_FOPT) | 8 | R | Undefined | 29.34.4/ 636 |
| 4002_0004 | Flash Common Command Object Registers (FTFE_FCCOB3) | 8 | R/W | 00h | 29.34.5/ 637 |
| 4002_0005 | Flash Common Command Object Registers (FTFE_FCCOB2) | 8 | R/W | 00h | 29.34.5/ 637 |
| 4002_0006 | Flash Common Command Object Registers (FTFE_FCCOB1) | 8 | R/W | 00h | 29.34.5/ 637 |
| 4002_0007 | Flash Common Command Object Registers (FTFE_FCCOB0) | 8 | R/W | 00h | 29.34.5/ 637 |
| 4002_0008 | Flash Common Command Object Registers (FTFE_FCCOB7) | 8 | R/W | 00h | 29.34.5/ 637 |
| 4002_0009 | Flash Common Command Object Registers (FTFE_FCCOB6) | 8 | R/W | 00h | 29.34.5/ 637 |
| 4002_000A | Flash Common Command Object Registers (FTFE_FCCOB5) | 8 | R/W | 00h | 29.34.5/ 637 |
| 4002_000B | Flash Common Command Object Registers (FTFE_FCCOB4) | 8 | R/W | 00h | 29.34.5/ 637 |
| 4002_000C | Flash Common Command Object Registers (FTFE_FCCOBB) | 8 | R/W | 00h | 29.34.5/ 637 |
| 4002_000D | Flash Common Command Object Registers (FTFE_FCCOBA) | 8 | R/W | 00h | 29.34.5/ 637 |
| 4002_000E | Flash Common Command Object Registers (FTFE_FCCOB9) | 8 | R/W | 00h | 29.34.5/ 637 |
| 4002_000F | Flash Common Command Object Registers (FTFE_FCCOB8) | 8 | R/W | 00h | 29.34.5/ 637 |
| 4002_0010 | Program Flash Protection Registers (FTFE_FPROT3) | 8 | R/W | Undefined | 29.34.6/ 638 |
| 4002_0011 | Program Flash Protection Registers (FTFE_FPROT2) | 8 | R/W | Undefined | 29.34.6/ 638 |
| 4002_0012 | Program Flash Protection Registers (FTFE_FPROT1) | 8 | R/W | Undefined | 29.34.6/ 638 |

*Table continues on the next page...*

**FTFE memory map (continued)**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4002_0013 | Program Flash Protection Registers (FTFE_FPROT0) | 8 | R/W | Undefined | 29.34.6/ 638 |
| 4002_0016 | EEPROM Protection Register (FTFE_FEPROT) | 8 | R/W | Undefined | 29.34.7/ 639 |
| 4002_0017 | Data Flash Protection Register (FTFE_FDPROT) | 8 | R/W | Undefined | 29.34.8/ 641 |

## 29.34.1 Flash Status Register (FTFE_FSTAT)

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

The CCIF, RDCOLERR, ACCERR, and FPVIOL bits are readable and writable. The MGSTAT0 bit is read only. The unassigned bits read 0 and are not writable.

**NOTE**

When set, the Access Error (ACCERR) and Flash Protection Violation (FPVIOL) bits in this register prevent the launch of any more commands or writes to the FlexRAM (when EEERDY is set) until the flag is cleared (by writing a one to it).

Address: 4002_0000h base + 0h offset = 4002_0000h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | CCIF | RDCOLERR | ACCERR | FPVIOL | 0 | | | MGSTAT0 |
| Write | w1c | w1c | w1c | w1c | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTFE_FSTAT field descriptions**

| Field | Description |
|---|---|
| 7<br>CCIF | Command Complete Interrupt Flag<br><br>The CCIF flag indicates that a FTFE command or EEPROM file system operation has completed. The CCIF flag is cleared by writing a 1 to CCIF to launch a command, and CCIF stays low until command completion or command violation. The CCIF flag is also cleared by a successful write to FlexRAM while enabled for EEE, and CCIF stays low until the EEPROM file system has created the associated EEPROM data record.<br><br>The CCIF bit is reset to 0 but is set to 1 by the memory controller at the end of the reset initialization sequence. Depending on how quickly the read occurs after reset release, the user may or may not see the 0 hardware reset value.<br><br>0 FTFE command or EEPROM file system operation in progress<br>1 FTFE command or EEPROM file system operation has completed |

*Table continues on the next page...*

**FTFE_FSTAT field descriptions (continued)**

| Field | Description |
|---|---|
| 6<br>RDCOLERR | FTFE Read Collision Error Flag<br><br>The RDCOLERR error bit indicates that the MCU attempted a read from an FTFE resource that was being manipulated by an FTFE command (CCIF=0). Any simultaneous access is detected as a collision error by the block arbitration logic. The read data in this case cannot be guaranteed. The RDCOLERR bit is cleared by writing a 1 to it. Writing a 0 to RDCOLERR has no effect.<br><br>0 No collision error detected<br>1 Collision error detected |
| 5<br>ACCERR | Flash Access Error Flag<br><br>The ACCERR error bit indicates an illegal access has occurred to an FTFE resource caused by a violation of the command write sequence or issuing an illegal FTFE command. While ACCERR is set, the CCIF flag cannot be cleared to launch a command. The ACCERR bit is cleared by writing a 1 to it. Writing a 0 to the ACCERR bit has no effect.<br><br>0 No access error detected<br>1 Access error detected |
| 4<br>FPVIOL | Flash Protection Violation Flag<br><br>The FPVIOL error bit indicates an attempt was made to program or erase an address in a protected area of program flash or data flash memory during a command write sequence or a write was attempted to a protected area of the FlexRAM while enabled for EEPROM. While FPVIOL is set, the CCIF flag cannot be cleared to launch a command. The FPVIOL bit is cleared by writing a 1 to it. Writing a 0 to the FPVIOL bit has no effect.<br><br>0 No protection violation detected<br>1 Protection violation detected |
| 3–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>MGSTAT0 | Memory Controller Command Completion Status Flag<br><br>The MGSTAT0 status flag is set if an error is detected during execution of an FTFE command or during the flash reset sequence. As a status flag, this bit cannot (and need not) be cleared by the user like the other error flags in this register.<br><br>The value of the MGSTAT0 bit for "command-N" is valid only at the end of the "command-N" execution when CCIF=1 and before the next command has been launched. At some point during the execution of "command-N+1," the previous result is discarded and any previous error is cleared. |

## 29.34.2 Flash Configuration Register (FTFE_FCNFG)

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

The erase control bits (ERSAREQ and ERSSUSP) have write restrictions. SWAP, PFLSH, RAMRDY, and EEERDY are read-only status bits. The unassigned bits read as noted and are not writable. The reset values for the SWAP, PFLSH, RAMRDY, and EEERDY bits are determined during the reset sequence.

Address: 4002_0000h base + 1h offset = 4002_0001h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | CCIE | RDCOLLIE | ERSAREQ | ERSSUSP | SWAP | PFLSH | RAMRDY | EEERDY |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## FTFE_FCNFG field descriptions

| Field | Description |
|---|---|
| 7<br>CCIE | Command Complete Interrupt Enable<br><br>The CCIE bit controls interrupt generation when an FTFE command completes.<br><br>0 Command complete interrupt disabled<br>1 Command complete interrupt enabled. An interrupt request is generated whenever the FSTAT[CCIF] flag is set. |
| 6<br>RDCOLLIE | Read Collision Error Interrupt Enable<br><br>The RDCOLLIE bit controls interrupt generation when an FTFE read collision error occurs.<br><br>0 Read collision error interrupt disabled<br>1 Read collision error interrupt enabled. An interrupt request is generated whenever an FTFE read collision error is detected (see the description of FSTAT[RDCOLERR]). |
| 5<br>ERSAREQ | Erase All Request<br><br>This bit issues a request to the memory controller to execute the Erase All Blocks command and release security. ERSAREQ is not directly writable but is under indirect user control. Refer to the device's Chip Configuration details on how to request this command.<br><br>The ERSAREQ bit sets when an erase all request is triggered external to the FTFE and CCIF is set (no command is currently being executed). ERSAREQ is cleared by the FTFE when the operation completes.<br><br>0 No request or request complete<br>1 Request to:<br>1. run the Erase All Blocks command,<br>2. verify the erased state,<br>3. program the security byte in the Flash Configuration Field to the unsecure state, and<br>4. release MCU security by setting the FSEC[SEC] field to the unsecure state. |
| 4<br>ERSSUSP | Erase Suspend<br><br>The ERSSUSP bit allows the user to suspend (interrupt) the Erase Flash Sector command while it is executing.<br><br>0 No suspend requested<br>1 Suspend the current Erase Flash Sector command execution. |
| 3<br>SWAP | Swap<br><br>The SWAP flag indicates which half of the program flash space is located at relative address 0x0000. The state of the SWAP flag is set by the FTFE during the reset sequence. See Swap Control command (program flash only devices) for information on swap management. |

*Table continues on the next page...*

## FTFE_FCNFG field descriptions (continued)

| Field | Description |
|---|---|
| | 0 For devices with FlexNVM: Logical program flash 0 block is located at relative address 0x0000<br>For devices with program flash only: Logical program flash 0 block is located at relative address 0x0000<br>1 For devices with FlexNVM: Reserved<br>For devices with program flash only: Logical program flash 1 block is located at relative address 0x0000 |
| 2<br>PFLSH | FTFE configuration<br>0 For devices with FlexNVM: FTFE configuration supports two logical program flash blocks and two logical FlexNVM blocks<br>For devices with program flash only: Reserved<br>1 For devices with FlexNVM: Reserved<br>For devices with program flash only: FTFE configuration supports four logical program flash blocks |
| 1<br>RAMRDY | RAM Ready<br>This flag indicates the current status of the FlexRAM/programming acceleration RAM.<br>For devices with FlexNVM: The state of the RAMRDY flag is normally controlled by the Set FlexRAM Function command. During the reset sequence, the RAMRDY flag is cleared if the FlexNVM block is partitioned for EEPROM and is set if the FlexNVM block is not partitioned for EEPROM. The RAMRDY flag is cleared if the Program Partition command is run to partition the FlexNVM block for EEPROM. The RAMRDY flag sets after completion of the Erase All Blocks command or execution of the erase-all operation triggered external to the FTFE.<br>For devices without FlexNVM: This bit should always be set.<br>0 For devices with FlexNVM: FlexRAM is not available for traditional RAM access.<br>For devices without FlexNVM: Programming acceleration RAM is not available.<br>1 For devices with FlexNVM: FlexRAM is available as traditional RAM only; writes to the FlexRAM do not trigger EEPROM operations.<br>For devices without FlexNVM: Programming acceleration RAM is available. |
| 0<br>EEERDY | For devices with FlexNVM: This flag indicates if the EEPROM backup data has been copied to the FlexRAM and is therefore available for read access.<br>During the reset sequence, the EEERDY flag remains clear while CCIF=0 and only sets if the FlexNVM block is partitioned for EEPROM.<br>For devices without FlexNVM: This bit is reserved.<br>0 For devices with FlexNVM: FlexRAM is not available for EEPROM operation.<br>1 For devices with FlexNVM: FlexRAM is available for EEPROM operations where:<br>• reads from the FlexRAM return data previously written to the FlexRAM in EEPROM mode and<br>• writes launch an EEPROM operation to store the written data in the FlexRAM and EEPROM backup. |

## 29.34.3 Flash Security Register (FTFE_FSEC)

This read-only register holds all bits associated with the security of the MCU and FTFE module.

During the reset sequence, the register is loaded with the contents of the flash security byte in the Flash Configuration Field located in program flash memory. The Flash basis for the values is signified by X in the reset value.

Address: 4002_0000h base + 2h offset = 4002_0002h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | KEYEN | | MEEN | | FSLACC | | SEC | |
| Write | | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**FTFE_FSEC field descriptions**

| Field | Description |
|---|---|
| 7–6<br>KEYEN | Backdoor Key Security Enable<br><br>These bits enable and disable backdoor key access to the FTFE module.<br><br>00 Backdoor key access disabled<br>01 Backdoor key access disabled (preferred KEYEN state to disable backdoor key access)<br>10 Backdoor key access enabled<br>11 Backdoor key access disabled |
| 5–4<br>MEEN | Mass Erase Enable Bits<br><br>Enables and disables mass erase capability of the FTFE module. The state of the MEEN bits is only relevant when the SEC bits are set to secure outside of NVM Normal Mode. When the SEC field is set to unsecure, the MEEN setting does not matter.<br><br>00 Mass erase is enabled<br>01 Mass erase is enabled<br>10 Mass erase is disabled<br>11 Mass erase is enabled |
| 3–2<br>FSLACC | Freescale Failure Analysis Access Code<br><br>These bits enable or disable access to the flash memory contents during returned part failure analysis at Freescale. When SEC is secure and FSLACC is denied, access to the program flash contents is denied and any failure analysis performed by Freescale factory test must begin with a full erase to unsecure the part.<br><br>When access is granted (SEC is unsecure, or SEC is secure and FSLACC is granted), Freescale factory testing has visibility of the current flash contents. The state of the FSLACC bits is only relevant when the SEC bits are set to secure. When the SEC field is set to unsecure, the FSLACC setting does not matter. |

*Table continues on the next page...*

**FTFE_FSEC field descriptions (continued)**

| Field | Description |
|---|---|
| | 00 Freescale factory access granted<br>01 Freescale factory access denied<br>10 Freescale factory access denied<br>11 Freescale factory access granted |
| 1–0<br>SEC | Flash Security<br><br>These bits define the security state of the MCU. In the secure state, the MCU limits access to FTFE module resources. The limitations are defined per device and are detailed in the Chip Configuration details. If the FTFE module is unsecured using backdoor key access, the SEC bits are forced to 10b.<br><br>00 MCU security status is secure<br>01 MCU security status is secure<br>10 MCU security status is unsecure (The standard shipping condition of the FTFE is unsecure.)<br>11 MCU security status is secure |

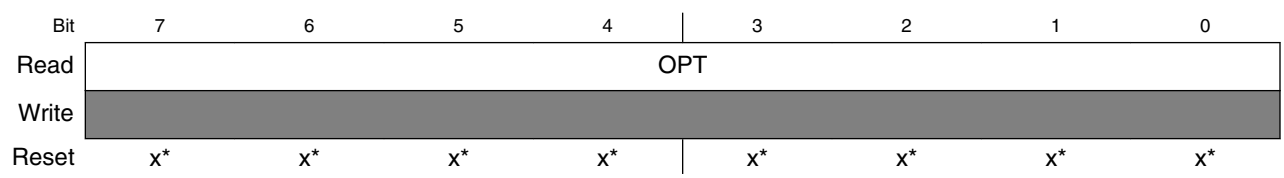
## 29.34.4 Flash Option Register (FTFE_FOPT)

The flash option register allows the MCU to customize its operations by examining the state of these read-only bits, which are loaded from NVM at reset. The function of the bits is defined in the device's Chip Configuration details.

All bits in the register are read-only.

During the reset sequence, the register is loaded from the flash nonvolatile option byte in the Flash Configuration Field located in program flash memory. The flash basis for the values is signified by X in the reset value.

Address: 4002_0000h base + 3h offset = 4002_0003h




* Notes:
- x = Undefined at reset.

**FTFE_FOPT field descriptions**

| Field | Description |
|---|---|
| 7–0<br>OPT | Nonvolatile Option<br><br>These bits are loaded from flash to this register at reset. Refer to the device's Chip Configuration details for the definition and use of these bits. |

## 29.34.5 Flash Common Command Object Registers (FTFE_FCCOB*n*)

The FCCOB register group provides 12 bytes for command codes and parameters. The individual bytes within the set append a 0-B hex identifier to the FCCOB register name: FCCOB0, FCCOB1, ..., FCCOBB.

Address: 4002_0000h base + 4h offset + (1d × i), where i=0d to 11d

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | CCOBn | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTFE_FCCOB*n* field descriptions**

| Field | Description |
|---|---|
| 7–0<br>CCOBn | The FCCOB register provides a command code and relevant parameters to the memory controller. The individual registers that compose the FCCOB data set can be written in any order, but you must provide all needed values, which vary from command to command. First, set up all required FCCOB fields and then initiate the command's execution by writing a 1 to the FSTAT[CCIF] bit. This clears the CCIF bit, which locks all FCCOB parameter fields and they cannot be changed by the user until the command completes (CCIF returns to 1). No command buffering or queueing is provided; the next command can be loaded only after the current command completes.<br><br>Some commands return information to the FCCOB registers. Any values returned to FCCOB are available for reading after the FSTAT[CCIF] flag returns to 1 by the memory controller.<br><br>The following table shows a generic FTFE command format. The first FCCOB register, FCCOB0, always contains the command code. This 8-bit value defines the command to be executed. The command code is followed by the parameters required for this specific FTFE command, typically an address and/or data values.<br><br>**NOTE:** The command parameter table is written in terms of FCCOB Number (which is equivalent to the byte number). This number is a reference to the FCCOB register name and is not the register address. |

| FCCOB Number[1] | Typical Command Parameter Contents [7:0] |
|---|---|
| 0 | FCMD (a code that defines the FTFE command) |
| 1 | Flash address [23:16] |
| 2 | Flash address [15:8] |
| 3 | Flash address [7:0] |
| 4 | Data Byte 0 |
| 5 | Data Byte 1 |
| 6 | Data Byte 2 |
| 7 | Data Byte 3 |
| 8 | Data Byte 4 |
| 9 | Data Byte 5 |

**FTFE_FCCOB*n* field descriptions (continued)**

| Field | Description |
|---|---|
| | **FCCOB Number**[1] / **Typical Command Parameter Contents [7:0]**<br>A — Data Byte 6<br>B — Data Byte 7<br><br>1. Refers to FCCOB register name, not register address<br><br>**FCCOB Endianness and Multi-Byte Access:**<br>The FCCOB register group uses a big endian addressing convention. For all command parameter fields larger than 1 byte, the most significant data resides in the lowest FCCOB register number. The FCCOB register group may be read and written as individual bytes, aligned words (2 bytes) or aligned longwords (4 bytes). |

1. Refers to FCCOB register name, not register address

## 29.34.6 Program Flash Protection Registers (FTFE_FPROT*n*)

The FPROT registers define which logical program flash regions are protected from program and erase operations. Protected flash regions cannot have their content changed; that is, these regions cannot be programmed and cannot be erased by any FTFE command. Unprotected regions can be changed by program and erase operations.

The four FPROT registers allow 32 protectable regions. Each bit protects a 1/32 region of the program flash memory.

| Program flash protection register | Program flash protection bits |
|---|---|
| FPROT0 | PROT[31:24] |
| FPROT1 | PROT[23:16] |
| FPROT2 | PROT[15:8] |
| FPROT3 | PROT[7:0] |

During the reset sequence, the FPROT registers are loaded with the contents of the program flash protection bytes in the Flash Configuration Field as indicated in the following table.

| Program flash protection register | Flash Configuration Field offset address |
|---|---|
| FPROT0 | 0x000B |
| FPROT1 | 0x000A |
| FPROT2 | 0x0009 |
| FPROT3 | 0x0008 |

To change the program flash protection that is loaded during the reset sequence, unprotect the sector of program flash memory that contains the Flash Configuration Field. Then, reprogram the program flash protection byte.

Address: 4002_0000h base + 10h offset + (1d × i), where i=0d to 3d

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | PROT | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**FTFE_FPROT*n* field descriptions**

| Field | Description |
|---|---|
| 7–0<br>PROT | Program Flash Region Protect<br><br>Each program flash region can be protected from program and erase operations by setting the associated PROT bit.<br><br>**In NVM Normal mode:** The protection can only be increased, meaning that currently unprotected memory can be protected, but currently protected memory cannot be unprotected. Since unprotected regions are marked with a 1 and protected regions use a 0, only writes changing 1s to 0s are accepted. This 1-to-0 transition check is performed on a bit-by-bit basis. Those FPROT bits with 1-to-0 transitions are accepted while all bits with 0-to-1 transitions are ignored.<br><br>**In NVM Special mode:** All bits of FPROT are writable without restriction. Unprotected areas can be protected and protected areas can be unprotected.<br><br>**Restriction:** The user must never write to any FPROT register while a command is running (CCIF=0).<br><br>Trying to alter data in any protected area in the program flash memory results in a protection violation error and sets the FSTAT[FPVIOL] bit. A full block erase of a program flash block is not possible if it contains any protected region.<br><br>Each bit in the 32-bit protection register represents 1/32 of the total program flash.<br><br>0 Program flash region is protected.<br>1 Program flash region is not protected |

## 29.34.7 EEPROM Protection Register (FTFE_FEPROT)

For devices with FlexNVM: The FEPROT register defines which EEPROM regions of the FlexRAM are protected against program and erase operations. Protected EEPROM regions cannot have their content changed by writing to it. Unprotected regions can be changed by writing to the FlexRAM.

For devices with program flash only: This register is reserved and not used.

Address: 4002_0000h base + 16h offset = 4002_0016h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | EPROT | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

## FTFE_FEPROT field descriptions

| Field | Description |
|---|---|
| 7–0<br>EPROT | EEPROM Region Protect<br><br>For devices with program flash only: Reserved<br><br>For devices with FlexNVM:<br><br>Individual EEPROM regions can be protected from alteration by setting the associated EPROT bit. The EPROT bits are not used when the FlexNVM Partition Code is set to data flash only. When the FlexNVM Partition Code is set to data flash and EEPROM or EEPROM only, each EPROT bit covers one-eighth of the configured EEPROM data (see the EEPROM Data Set Size parameter description).<br><br>**In NVM Normal mode:** The protection can only be increased. This means that currently-unprotected memory can be protected, but currently-protected memory cannot be unprotected. Since unprotected regions are marked with a 1 and protected regions use a 0, only writes changing 1s to 0s are accepted. This 1-to-0 transition check is performed on a bit-by-bit basis. Those FEPROT bits with 1-to-0 transitions are accepted while all bits with 0-to-1 transitions are ignored.<br><br>**In NVM Special mode:** All bits of the FEPROT register are writable without restriction. Unprotected areas can be protected and protected areas can be unprotected.<br><br>**Restriction:** Never write to the FEPROT register while a command is running (CCIF=0).<br><br>**Reset:** During the reset sequence, the FEPROT register is loaded with the contents of the FlexRAM protection byte in the Flash Configuration Field located in program flash. The flash basis for the reset values is signified by X in the register diagram. To change the EEPROM protection that will be loaded during the reset sequence, the sector of program flash that contains the Flash Configuration Field must be unprotected; then the EEPROM protection byte must be erased and reprogrammed.<br><br>Trying to alter data by writing to any protected area in the EEPROM results in a protection violation error and sets the FSTAT[FPVIOL] bit.<br><br>0 For devices with program flash only: Reserved<br>For devices with FlexNVM: EEPROM region is protected<br><br>1 For devices with program flash only: Reserved<br>For devices with FlexNVM: EEPROM region is not protected |

## 29.34.8 Data Flash Protection Register (FTFE_FDPROT)

The FDPROT register defines which data flash regions are protected against program and erase operations. Protected Flash regions cannot have their content changed; that is, these regions cannot be programmed and cannot be erased by any FTFE command. Unprotected regions can be changed by both program and erase operations.

Address: 4002_0000h base + 17h offset = 4002_0017h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | DPROT | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**FTFE_FDPROT field descriptions**

| Field | Description |
|---|---|
| 7–0<br>DPROT | Data Flash Region Protect<br><br>Individual data flash regions can be protected from program and erase operations by setting the associated DPROT bit. Each DPROT bit protects one-eighth of the partitioned data flash memory space. The granularity of data flash protection cannot be less than the data flash sector size. If an unused DPROT bit is set, the Erase all Blocks command does not execute and sets the FSTAT[FPVIOL] bit.<br><br>**In NVM Normal mode:** The protection can only be increased, meaning that currently unprotected memory can be protected but currently protected memory cannot be unprotected. Since unprotected regions are marked with a 1 and protected regions use a 0, only writes changing 1s to 0s are accepted. This 1-to-0 transition check is performed on a bit-by-bit basis. Those FDPROT bits with 1-to-0 transitions are accepted while all bits with 0-to-1 transitions are ignored.<br><br>**In NVM Special mode:** All bits of the FDPROT register are writable without restriction. Unprotected areas can be protected and protected areas can be unprotected.<br><br>**Restriction:** The user must never write to the FDPROT register while a command is running (CCIF=0).<br><br>**Reset:** During the reset sequence, the FDPROT register is loaded with the contents of the data flash protection byte in the Flash Configuration Field located in program flash memory. The flash basis for the reset values is signified by X in the register diagram. To change the data flash protection that will be loaded during the reset sequence, unprotect the sector of program flash that contains the Flash Configuration Field. Then, erase and reprogram the data flash protection byte.<br><br>Trying to alter data with the program and erase commands in any protected area in the data flash memory results in a protection violation error and sets the FSTAT[FPVIOL] bit. A block erase of any data flash memory block (see the Erase Flash Block command description) is not possible if the data flash block contains any protected region or if the FlexNVM memory has been partitioned for EEPROM.<br><br>0 Data Flash region is protected<br>1 Data Flash region is not protected |

## 29.4 Functional Description

The following sections describe functional details of the FTFE module.

### 29.4.1 Program flash memory swap

The user can configure the memory map of the program flash space such that either half of the program flash memory can exist at relative address 0x0000. This swap feature enables the lower half of the program flash space to be operational while the upper half is being updated for future use.

The Swap Control command handles swapping the two halves of program flash memory within the memory map. See Swap Control command (program flash only devices) for details.

### 29.4.2 Flash Protection

Individual regions within the flash memory can be protected from program and erase operations. Protection is controlled by the following registers:

- FPROT*n* — Four registers protect 32 regions of the program flash memory as shown in the following figure

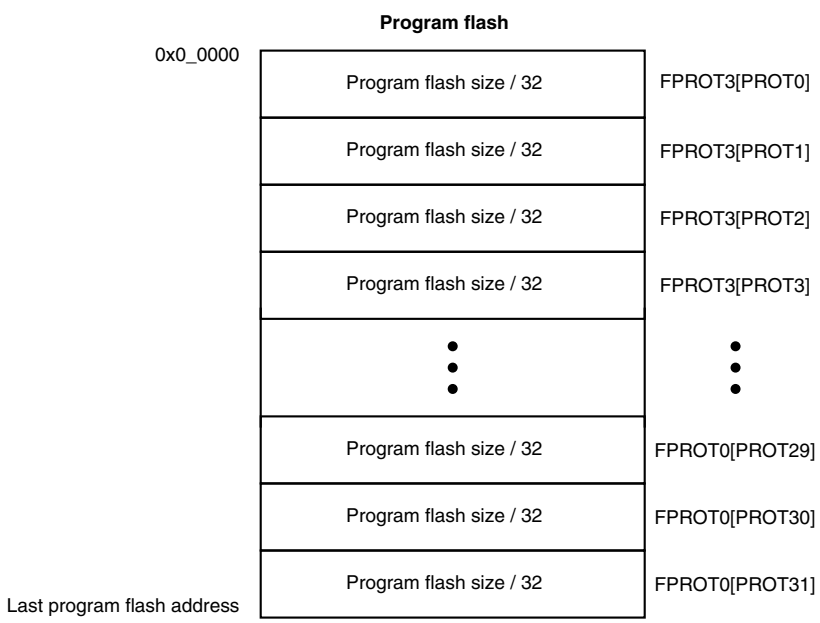



**Figure 29-27. Program flash protection**

- FDPROT —
  - For $2^n$ data flash sizes, protects eight regions of the data flash memory as shown in the following figure

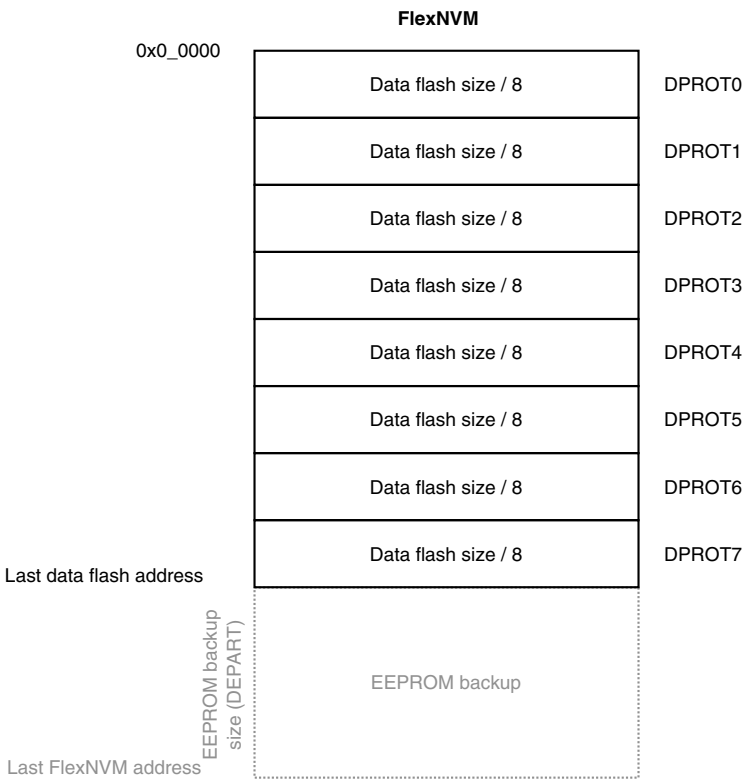



**Figure 29-28. Data flash protection ($2^n$ data flash sizes)**

  - For the non-$2^n$ data flash sizes, the protection granularity is 16KB. Therefore, for 96KB data flash size, only the DPROT[5:0] bits are used.

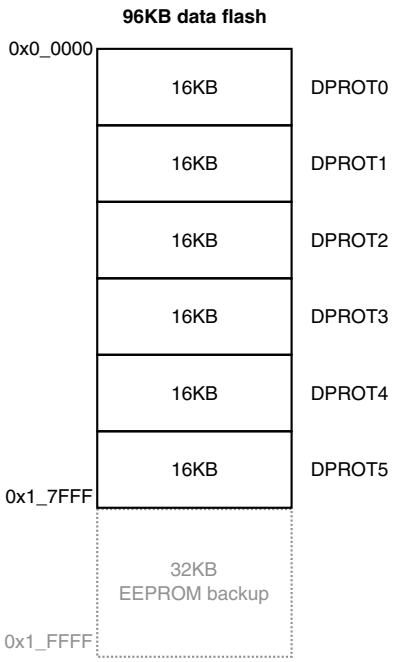



**Figure 29-29. Data flash protection (96KB data flash size)**

- FEPROT — Protects eight regions of the EEPROM memory as shown in the following figure

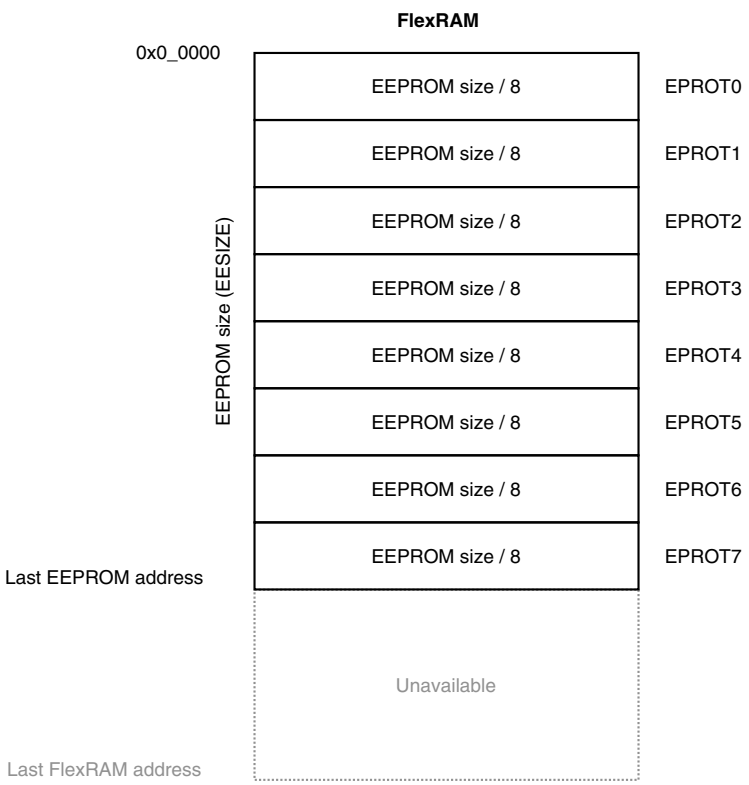



**Figure 29-30. EEPROM protection**

## 29.4.3 FlexNVM Description

This section describes the FlexNVM memory. This section does not apply for devices that contain only program flash memory.

### 29.4.3.1 FlexNVM Block Partitioning for FlexRAM

The user can configure the FlexNVM block as either:

- Basic data flash,
- EEPROM flash records to support the built-in EEPROM feature, or
- A combination of both.

The user's FlexNVM configuration choice is specified using the Program Partition command described in Program Partition command.

**CAUTION**

While different partitions of the FlexNVM block are available, the intention is that a single partition choice is used throughout the entire lifetime of a given application. The FlexNVM partition code choices affect the endurance and data retention characteristics of the device.

### 29.4.3.2 EEPROM User Perspective

The EEPROM system is shown in the following figure.

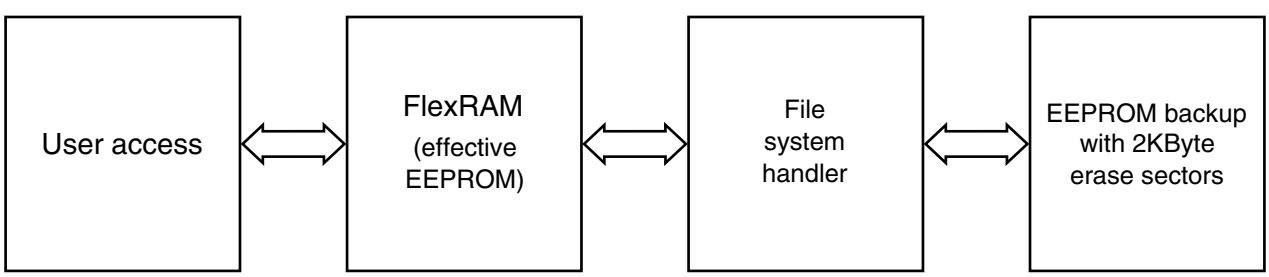



**Figure 29-31. Top Level EEPROM Architecture**

To handle varying customer requirements, the FlexRAM and FlexNVM blocks can be split into partitions as shown in the figure below.

1. **EEPROM partition** (EEESIZE) — The amount of FlexRAM used for EEPROM can be set from 0 Bytes (no EEPROM) to the maximum FlexRAM size (see Table 29-2). The remainder of the FlexRAM not used for EEPROM is not accessible while the FlexRAM is configured for EEPROM (see Set FlexRAM Function command).

The EEPROM partition grows upward from the bottom of the FlexRAM address space.

2. **Data flash partition** (DEPART) — The amount of FlexNVM memory used for data flash can be programmed from 0 bytes (all of the FlexNVM block is available for EEPROM backup) to the maximum size of the FlexNVM block (see Table 29-4).
3. **FlexNVM EEPROM partition** — The amount of FlexNVM memory used for EEPROM backup, which is equal to the FlexNVM block size minus the data flash memory partition size. The EEPROM backup size must be at least 16 times the EEPROM partition size in FlexRAM.
4. **EEPROM split factor** (EEESPLIT) — The FlexRAM partitioned for EEPROM can be divided into two subsystems, each backed by half of the partitioned EEPROM backup. One subsystem (A) is 1/8, 1/4, or 1/2 of the partitioned FlexRAM with the remainder belonging to the other subsystem (B).

The partition information (EEESIZE, DEPART, EEESPLIT) is stored in the data flash IFR and is programmed using the Program Partition command (see Program Partition command). Typically, the Program Partition command is executed only once in the lifetime of the device.

Data flash memory is useful for applications that need to quickly store large amounts of data or store data that is static. The EEPROM partition in FlexRAM is useful for storing smaller amounts of data that will be changed often. The EEPROM partition in FlexRAM can be further sub-divided to provide subsystems, each backed by the same amount of EEPROM backup with subsytem A having higher endurance if the split factor is 1/8 or 1/4.

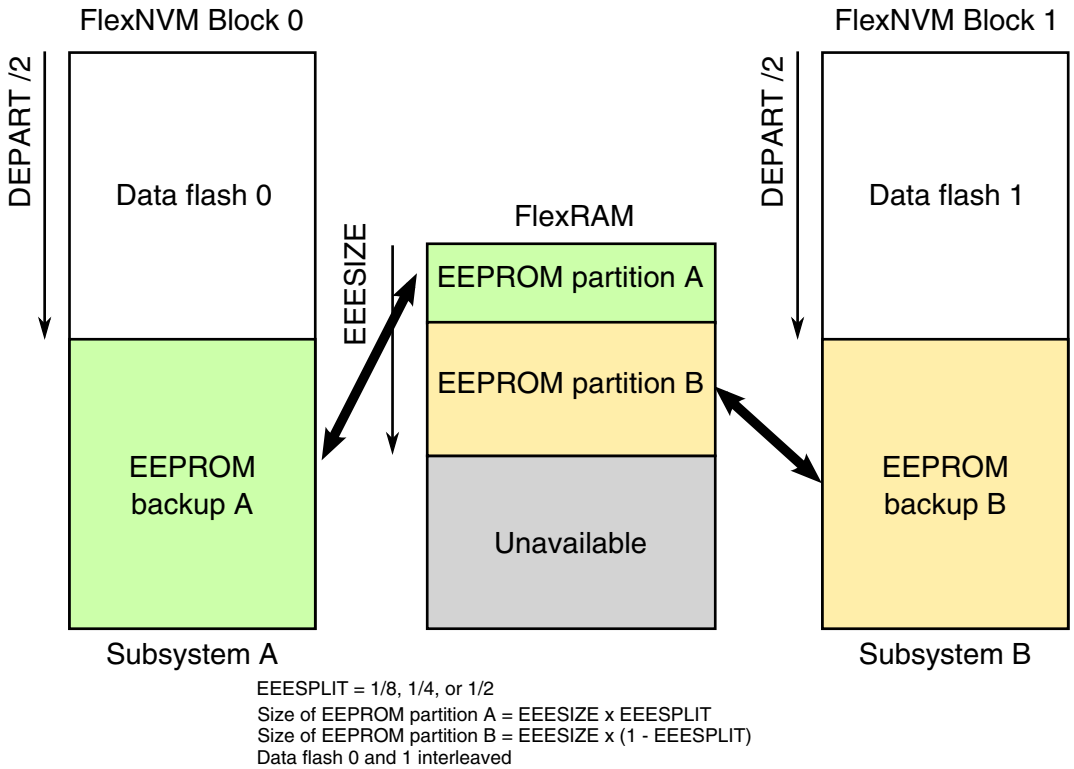



**Figure 29-32. FlexRAM to FlexNVM Memory Mapping for EEPROM**

### 29.4.3.3 EEPROM implementation overview

Out of reset with the FSTAT[CCIF] bit clear, the partition settings (EEESIZE, DEPART, EEESPLIT) are read from the data flash IFR and the EEPROM file system is initialized accordingly. The EEPROM file system locates all valid EEPROM data records in EEPROM backup and copies the newest data to FlexRAM. The FSTAT[CCIF] and FCNFG[EEERDY] bits are set after data from all valid EEPROM data records is copied to the FlexRAM. After the CCIF bit is set, the FlexRAM is available for read or write access.

When configured for EEPROM use, writes to an unprotected location in FlexRAM invokes the EEPROM file system to program a new EEPROM data record in the EEPROM backup memory in a round-robin fashion. As needed, the EEPROM file system identifies the EEPROM backup sector that is being erased for future use and partially erases that EEPROM backup sector. After a write to the FlexRAM, the FlexRAM is not accessible until the FSTAT[CCIF] bit is set. The FCNFG[EEERDY] bit will also be set. If enabled, the interrupt associated with the FSTAT[CCIF] bit can be used to determine when the FlexRAM is available for read or write access.

After a sector in EEPROM backup is full of EEPROM data records, EEPROM data records from the sector holding the oldest data are gradually copied over to a previously-erased EEPROM backup sector. When the sector copy completes, the EEPROM backup sector holding the oldest data is tagged for erase.

### 29.4.3.4 Write endurance to FlexRAM for EEPROM

When the FlexNVM partition code is not set to full data flash, the EEPROM data set size can be set to any of several non-zero values.

The bytes not assigned to data flash via the FlexNVM partition code are used by the FTFE to obtain an effective endurance increase for the EEPROM data. The built-in EEPROM record management system raises the number of program/erase cycles that can be attained prior to device wear-out by cycling the EEPROM data through a larger EEPROM NVM storage space.

While different partitions of the FlexNVM are available, the intention is that a single choice for the FlexNVM partition code and EEPROM data set size is used throughout the entire lifetime of a given application. The EEPROM endurance equation and graph shown below assume that only one configuration is ever used.

$$\text{Writes\_subsystem} = \frac{\text{EEPROM} - 2 \times \text{EEESPLIT} \times \text{EEESIZE}}{\text{EEESPLIT} \times \text{EEESIZE}} \times \text{Write\_efficiency} \times n_{\text{nvmcycee}}$$

where

- Writes_subsystem — minimum number of writes to each FlexRAM location for subsystem (each subsystem can have different endurance)
- EEPROM — allocated FlexNVM for each EEPROM subsystem based on DEPART; entered with Program Partition command
- EEESPLIT — FlexRAM split factor for subsystem; entered with the Program Partition command
- EEESIZE — allocated FlexRAM based on DEPART; entered with Program Partition command
- Write_efficiency —
  - 0.25 for 8-bit writes to FlexRAM
  - 0.50 for 16-bit or 32-bit writes to FlexRAM
- $n_{nvmcycee}$ — EEPROM-backup cycling endurance

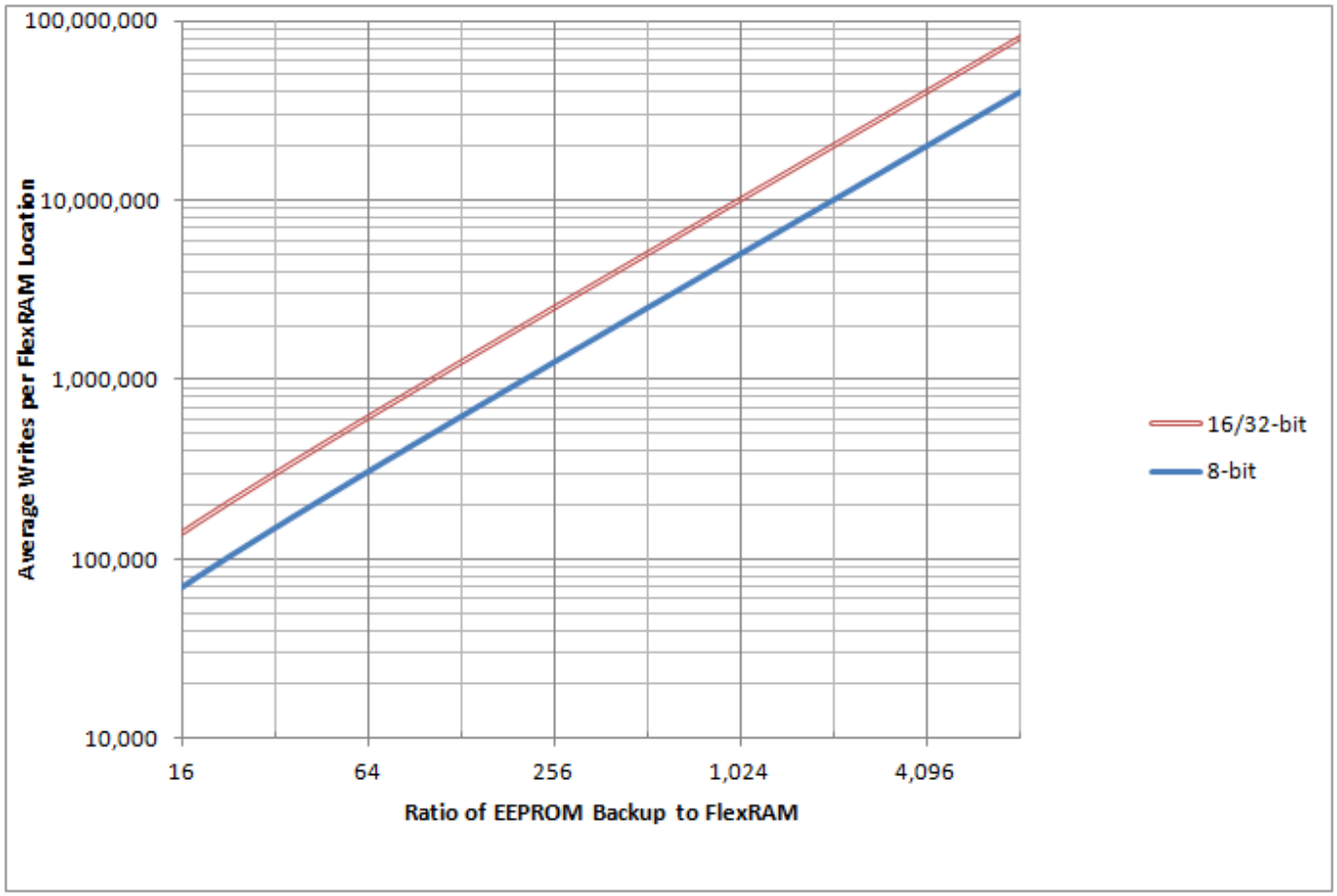



**Figure 29-33. EEPROM backup writes to FlexRAM**

## 29.4.4 Interrupts

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

**Table 29-30. FTFE Interrupt Sources**

| FTFE Event | Readable Status Bit | Interrupt Enable Bit |
|---|---|---|
| FTFE Command Complete | FSTAT[CCIF] | FCNFG[CCIE] |
| FTFE Read Collision Error | FSTAT[RDCOLERR] | FCNFG[RDCOLLIE] |

### Note

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

## 29.4.5 Flash Operation in Low-Power Modes

### 29.4.5.1 Wait Mode

When the MCU enters wait mode, the FTFE module is not affected. The FTFE module can recover the MCU from wait via the command complete interrupt (see Interrupts).

### 29.4.5.2 Stop Mode

When the MCU requests stop mode, if an FTFE command is active (CCIF = 0) the command execution completes before the MCU is allowed to enter stop mode.

**CAUTION**

The MCU should never enter stop mode while any FTFE command is running (CCIF = 0).

**NOTE**

While the MCU is in very-low-power modes (VLPR, VLPW, VLPS), the FTFE module does not accept flash commands.

## 29.4.6 Functional modes of operation

The FTFE module has two operating modes: NVM Normal and NVM Special. The operating mode affects the command set availability (see Table 29-31). Refer to the Chip Configuration details of this device for how to activate each mode.

## 29.4.7 Flash memory reads and ignored writes

The FTFE module requires only the flash address to execute a flash memory read. MCU read access is available to all flash memory.

The MCU must not read from the flash memory while commands are running (as evidenced by CCIF=0) on that block. Read data cannot be guaranteed from a flash block while any command is processing within that block. The block arbitration logic detects any simultaneous access and reports this as a read collision error (see the FSTAT[RDCOLERR] bit).

### 29.4.8 Read while write (RWW)

The following simultaneous accesses are allowed for devices with FlexNVM:

- The user may read from the program flash memory while commands (typically program and erase operations) are active in the data flash and FlexRAM memory space.
- The user may read from one logical program flash memory space while commands are active in another logical program flash memory space.
- The MCU can fetch instructions from program flash during both data flash program and erase operations and while EEPROM-backup is maintained by the EEPROM commands.
- Conversely, the user may read from data flash and FlexRAM while program and erase commands are executing on the program flash.
- The user may also read from one logical data flash memory space while commands other than Program Partition are active in the other logical data flash memory space.
- When configured as traditional RAM, writes to the FlexRAM are allowed during data flash operations.

Simultaneous data flash operations and FlexRAM writes, when FlexRAM is used for EEE, are not possible.

The following simultaneous accesses are allowed for devices with program flash only:

- The user may read from one logical program flash memory space while commands are active in the other logical program flash memory space.

Simultaneous operations are further discussed in Allowed simultaneous flash operations.

### 29.4.9 Flash Program and Erase

All flash functions except read require the user to setup and launch an FTFE command through a series of peripheral bus writes. The user cannot initiate any further FTFE commands until notified that the current command has completed. The FTFE command structure and operation are detailed in FTFE Command Operations.

### 29.4.10 FTFE Command Operations

FTFE command operations are typically used to modify flash memory contents. The next sections describe:

- The command write sequence used to set FTFE command parameters and launch execution
- A description of all FTFE commands available

#### 29.4.10.1 Command Write Sequence

FTFE commands are specified using a command write sequence illustrated in Figure 29-34. The FTFE module performs various checks on the command (FCCOB) content and continues with command execution if all requirements are fulfilled.

Before launching a command, the ACCERR and FPVIOL bits in the FSTAT register must be zero and the CCIF flag must read 1 to verify that any previous command has completed. If CCIF is zero, the previous command execution is still active, a new command write sequence cannot be started, and all writes to the FCCOB registers are ignored.

Attempts to launch an FTFE command in VLP mode will be ignored.

##### 29.4.10.1.1 Load the FCCOB Registers

The user must load the FCCOB registers with all parameters required by the desired FTFE command. The individual registers that make up the FCCOB data set can be written in any order.

##### 29.4.10.1.2 Launch the Command by Clearing CCIF

Once all relevant command parameters have been loaded, the user launches the command by clearing the FSTAT[CCIF] bit by writing a '1' to it. The CCIF flag remains zero until the FTFE command completes.

The FSTAT register contains a blocking mechanism, which prevents a new command from launching (can't clear CCIF) if the previous command resulted in an access error (FSTAT[ACCERR]=1) or a protection violation (FSTAT[FPVIOL]=1). In error scenarios, two writes to FSTAT are required to initiate the next command: the first write clears the error flags, the second write clears CCIF.

##### 29.4.10.1.3 Command Execution and Error Reporting

The command processing has several steps:

1. The FTFE reads the command code and performs a series of parameter checks and protection checks, if applicable, which are unique to each command.

If the parameter check fails, the FSTAT[ACCERR] (access error) flag is set. ACCERR reports invalid instruction codes and out-of bounds addresses. Usually, access errors suggest that the command was not set-up with valid parameters in the FCCOB register group.

Program and erase commands also check the address to determine if the operation is requested to execute on protected areas. If the protection check fails, the FSTAT[FPVIOL] (protection error) flag is set.

Command processing never proceeds to execution when the parameter or protection step fails. Instead, command processing is terminated after setting the FSTAT[CCIF] bit.

2. If the parameter and protection checks pass, the command proceeds to execution. Run-time errors, such as failure to erase verify, may occur during the execution phase. Run-time errors are reported in the FSTAT[MGSTAT0] bit. A command may have access errors, protection errors, and run-time errors, but the run-time errors are not seen until all access and protection errors have been corrected.
3. Command execution results, if applicalbe, are reported back to the user via the FCCOB and FSTAT registers.
4. The FTFE sets the FSTAT[CCIF] bit signifying that the command has completed.

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

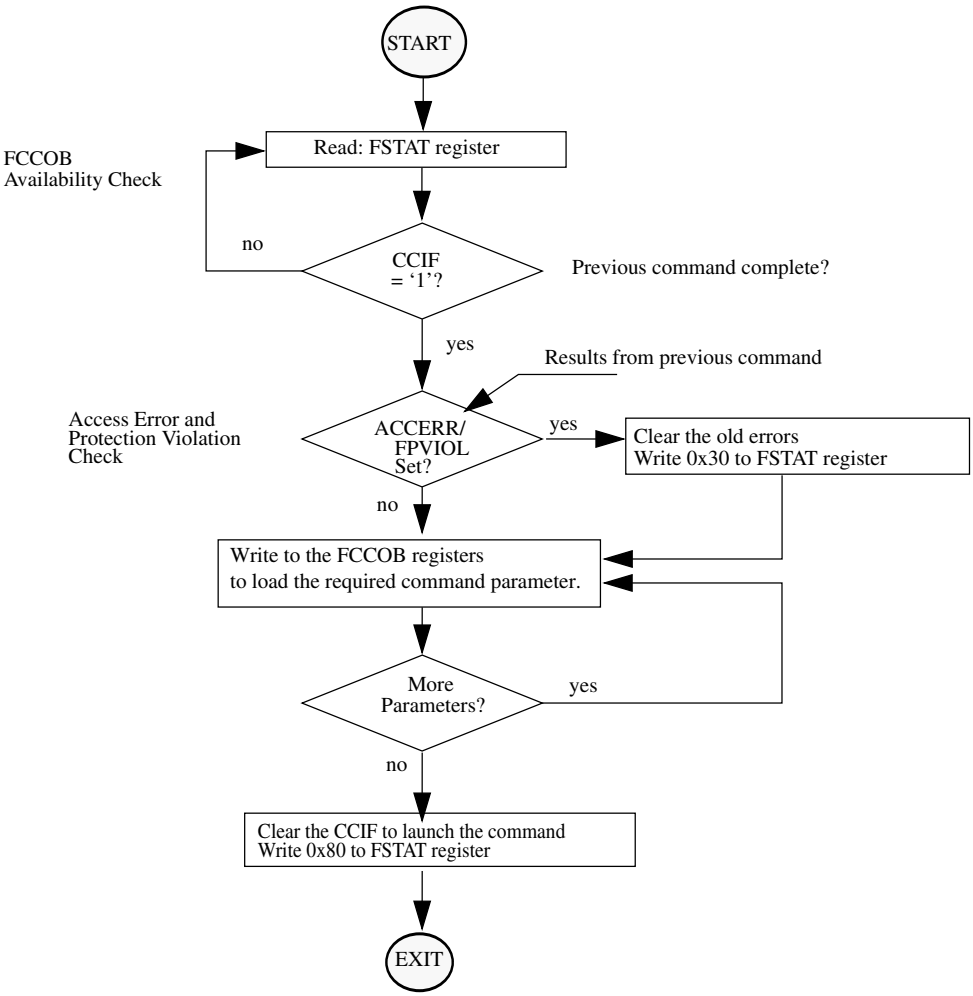



**Figure 29-34. Generic Flash Command Write Sequence Flowchart**

## 29.4.10.2 Flash commands

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

| FCMD | Command | Program flash 0 | Program flash 1 (Devices with only program flash) | Data flash (Devices with FlexNVM) | FlexRAM (Devices with FlexNVM) | Function |
|---|---|---|---|---|---|---|
| 0x00 | Read 1s Block | × | × | × | | Verify that a program flash or data flash block is erased. FlexNVM block must not be partitioned for EEPROM. |
| 0x01 | Read 1s Section | × | × | × | | Verify that a given number of program flash or data flash locations from a starting address are erased. |
| 0x02 | Program Check | × | × | × | | Tests previously-programmed phrases at margin read levels. |
| 0x03 | Read Resource | IFR,ID | IFR | IFR | | Read 8 bytes from program flash IFR, data flash IFR, or version ID. |
| 0x07 | Program Phrase | × | × | × | | Program 8 bytes in a program flash block or a data flash block. |
| 0x08 | Erase Flash Block | × | × | × | | Erase a program flash block or data flash block. An erase of any flash block is only possible when unprotected. FlexNVM block must not be partitioned for EEPROM. |
| 0x09 | Erase Flash Sector | × | × | × | | Erase all bytes in a program flash or data flash sector. |

*Table continues on the next page...*

| FCMD | Command | Program flash 0 | Program flash 1 (Devices with only program flash) | Data flash (Devices with FlexNVM) | FlexRAM (Devices with FlexNVM) | Function |
|---|---|---|---|---|---|---|
| 0x0B | Program Section | × | × | × | × | Program data from the Section Program Buffer to a program flash or data flash block. |
| 0x40 | Read 1s All Blocks | × | × | × | × | Verify that all program flash, data flash blocks, EEPROM backup data records, and data flash IFR are erased then release MCU security. |
| 0x41 | Read Once | IFR | | | | Read 8 bytes of a dedicated 64 byte field in the program flash 0 IFR. |
| 0x43 | Program Once | IFR | | | | One-time program of 8 bytes of a dedicated 64-byte field in the program flash 0 IFR. |

*Table continues on the next page...*

| FCMD | Command | Program flash 0 | Program flash 1 (Devices with only program flash) | Data flash (Devices with FlexNVM) | FlexRAM (Devices with FlexNVM) | Function |
|---|---|---|---|---|---|---|
| 0x44 | Erase All Blocks | × | × | × | × | Erase all program flash blocks, program flash swap IFR, data flash blocks, FlexRAM, EEPROM backup data records, and data flash IFR. Then, verify-erase and release MCU security.<br>**NOTE:**<br>An erase is only possible when all memory locations are unprotected. |
| 0x45 | Verify Backdoor Access Key | × | x | | | Release MCU security after comparing a set of user-supplied security keys to those stored in the program flash. |
| 0x46 | Swap Control | x | x | | | Handles swap-related activities. |
| 0x80 | Program Partition | | | IFR, × | × | Program the FlexNVM Partition Code and EEPROM Data Set Size into the data flash IFR. format all EEPROM backup data sectors allocated for EEPROM, initialize the FlexRAM. |

*Table continues on the next page...*

| FCMD | Command | Program flash 0 | Program flash 1 (Devices with only program flash) | Data flash (Devices with FlexNVM) | FlexRAM (Devices with FlexNVM) | Function |
|---|---|---|---|---|---|---|
| 0x81 | Set FlexRAM Function | | | × | × | Switches FlexRAM function between RAM and EEPROM. When switching to EEPROM, FlexNVM is not available while valid data records are being copied from EEPROM backup to FlexRAM. |

### 29.4.10.3 Flash commands by mode

The following table shows the flash commands that can be executed in each flash operating mode.

**Table 29-31. Flash commands by mode**

| FCMD | Command | NVM Normal | | | NVM Special | | |
|---|---|---|---|---|---|---|---|
| | | Unsecure | Secure | MEEN=10 | Unsecure | Secure | MEEN=10 |
| 0x00 | Read 1s Block | × | × | × | × | — | — |
| 0x01 | Read 1s Section | × | × | × | × | — | — |
| 0x02 | Program Check | × | × | × | × | — | — |
| 0x03 | Read Resource | × | × | × | × | — | — |
| 0x07 | Program Phrase | × | × | × | × | — | — |
| 0x08 | Erase Flash Block | × | × | × | × | — | — |
| 0x09 | Erase Flash Sector | × | × | × | × | — | — |
| 0x0B | Program Section | × | × | × | × | — | — |
| 0x40 | Read 1s All Blocks | × | × | × | × | × | — |
| 0x41 | Read Once | × | × | × | × | — | — |
| 0x43 | Program Once | × | × | × | × | — | — |
| 0x44 | Erase All Blocks | × | × | × | × | × | — |
| 0x45 | Verify Backdoor Access Key | × | × | × | × | — | — |
| 0x46 | Swap Control | × | × | × | × | — | — |
| 0x80 | Program Partition | × | × | × | × | — | — |

*Table continues on the next page...*

**Table 29-31. Flash commands by mode (continued)**

| FCMD | Command | NVM Normal | | | NVM Special | | |
|---|---|---|---|---|---|---|---|
| | | Unsecure | Secure | MEEN=10 | Unsecure | Secure | MEEN=10 |
| 0x81 | Set FlexRAM Function | × | × | × | × | — | — |

### 29.4.10.4 Allowed simultaneous flash operations

Only the operations marked 'OK' in the following table are permitted to run simultaneously on the program flash, data flash, and FlexRAM memories. Some operations cannot be executed simultaneously because certain hardware resources are shared by the memories. The priority has been placed on permitting program flash reads while program and erase operations execute on the FlexNVM and FlexRAM. This provides read (program flash) while write (FlexNVM, FlexRAM) functionality.

For devices containing FlexNVM:

**Table 29-32. Allowed Simultaneous Memory Operations**

| | | Program flash 0/1 | | | Data flash | | | FlexRAM | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Read | Program Phrase | Erase Flash Sector[1] | Read | Program Phrase | Erase Flash Sector[1] | Read | E-Write[2] | R-Write[3] |
| Program flash 1/0 | Read | | OK | OK | | OK | OK | | OK | |
| | Program Phrase | OK | | | OK | | | OK | | OK |
| | Erase Flash Sector[1] | OK | | | OK | | | OK | | OK |
| Data flash 1/0 | Read | | OK | OK | | OK | OK | | | |
| | Program Phrase | OK | | | OK | | | OK | | OK |
| | Erase Flash Sector[1] | OK | | | OK | | | OK | | OK |
| FlexRAM | Read | | OK | OK | | OK | OK | | | |
| | E-Write[2] | OK | | | | | | | | |
| | R-Write[3] | | OK | OK | | OK | OK | | | |

1. Also applies to Erase Flash Block
2. When FlexRAM configured for EEPROM (EEERDY=1).
3. When FlexRAM configured as traditional RAM (RAMRDY=1); single cycle operation.

For devices containing program flash only:

**Table 29-33. Allowed Simultaneous Memory Operations**

| | | Program flash X[1] | | | |
|---|---|---|---|---|---|
| | | Read | Program Phrase | Erase Flash Sector | Erase Flash Block |
| Program flash Y[1] | Read | | OK | OK | OK |
| | Program Phrase | OK | | | |
| | Erase Flash Sector | OK | | | |
| | Erase Flash Block | OK | | | |

1. P-Flash X refers to any of the P-Flash blocks (0, 1) and P-Flash Y refers to any of the P-Flash blocks (0, 1), but not the same block. Thus, it is possible to read from any of the blocks while programming or erasing another.

## 29.4.11 Margin Read Commands

The Read-1s commands (Read 1s All Blocks, Read 1s Block, and Read 1s Section) and the Program Check command have a margin choice parameter that allows the user to apply non-standard read reference levels to the program flash and data flash array reads performed by these commands. Using the preset 'user' and 'factory' margin levels, these commands perform their associated read operations at tighter tolerances than a 'normal' read. These non-standard read levels are applied only during the command execution. All simple (uncommanded) flash array reads to the MCU always use the standard, un-margined, read reference level.

Only the 'normal' read level should be employed during normal flash usage. The non-standard, 'user' and 'factory' margin levels should be employed only in special cases. They can be used during special diagnostic routines to gain confidence that the device is not suffering from the end-of-life data loss customary of flash memory devices.

Erased ('1') and programmed ('0') bit states can degrade due to elapsed time and data cycling (number of times a bit is erased and re-programmed). The lifetime of the erased states is relative to the last erase operation. The lifetime of the programmed states is measured from the last program time.

The 'user' and 'factory' levels become, in effect, a minimum safety margin; i.e. if the reads pass at the tighter tolerances of the 'user' and 'factory' margins, then the 'normal' reads have at least this much safety margin before they experience data loss.

The 'user' margin is a small delta to the normal read reference level. 'User' margin levels can be employed to check that flash memory contents have adequate margin for normal level read operations. If unexpected read results are encountered when checking flash memory contents at the 'user' margin levels, loss of information might soon occur during 'normal' readout.

The 'factory' margin is a bigger deviation from the norm, a more stringent read criteria that should only be attempted immediately (or very soon) after completion of an erase or program command, early in the cycling life. 'Factory' margin levels can be used to check that flash memory contents have adequate margin for long-term data retention at the normal level setting. If unexpected results are encountered when checking flash memory contents at 'factory' margin levels, the flash memory contents should be erased and reprogrammed.

**CAUTION**

Factory margin levels must only be used during verify of the initial factory programming.

## 29.4.12 Flash command descriptions

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

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

Ensure that the ACCERR and FPVIOL bits in the FSTAT register are cleared prior to starting the command write sequence. As described in Launch the Command by Clearing CCIF, a new command cannot be launched while these error flags are set.

Do not attempt to read a flash block while the FTFE is running a command (CCIF = 0) on that same block. The FTFE may return invalid data to the MCU with the collision error flag (FSTAT[RDCOLERR]) set.

When required by the command, address bit 23 selects between:

- program flash memory (=0)
- data flash memory (=1)

**CAUTION**

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

### 29.4.12.1 Read 1s Block command

The Read 1s Block command checks to see if an entire program flash or data flash logical block has been erased to the specified margin level. The FCCOB flash address bits determine which logical block is erase-verified.

**Table 29-34. Read 1s Block Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x00 (RD1BLK) |
| 1 | Flash address [23:16] in the flash block to be verified |
| 2 | Flash address [15:8] in the flash block to be verified |
| 3 | Flash address [7:0][1] in the flash block to be verified |
| 4 | Read-1 Margin Choice |

1. Must be 128-bit aligned (Flash address [3:0] = 0000).

After clearing CCIF to launch the Read 1s Block command, the FTFE sets the read margin for 1s according to Table 29-35 and then reads all locations within the selected program flash or data flash block.

When the data flash is targeted, DEPART must be set for no EEPROM, else the Read 1s Block command aborts setting the FSTAT[ACCERR] bit. If the FTFE fails to read all 1s (i.e. the flash block is not fully erased), the FSTAT[MGSTAT0] bit is set. The CCIF flag sets after the Read 1s Block operation has completed.

**Table 29-35. Margin Level Choices for Read 1s Block**

| Read Margin Choice | Margin Level Description |
|---|---|
| 0x00 | Use the 'normal' read level for 1s |
| 0x01 | Apply the 'User' margin to the normal read-1 level |
| 0x02 | Apply the 'Factory' margin to the normal read-1 level |

**Table 29-36. Read 1s Block Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| An invalid margin choice is specified | FSTAT[ACCERR] |
| Program flash is selected and the address is out of program flash range | FSTAT[ACCERR] |
| Data flash is selected and the address is out of data flash range | FSTAT[ACCERR] |
| Data flash is selected with EEPROM enabled | FSTAT[ACCERR] |
| Flash address is not 128-bit aligned | FSTAT[ACCERR] |
| Read-1s fails | FSTAT[MGSTAT0] |

### 29.4.12.2 Read 1s Section command

The Read 1s Section command checks if a section of program flash or data flash memory is erased to the specified read margin level. The Read 1s Section command defines the starting address and the number of 128 bits to be verified.

**Table 29-37. Read 1s Section Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x01 (RD1SEC) |
| 1 | Flash address [23:16] of the first 128 bits to be verified |
| 2 | Flash address [15:8] of the first 128 bits to be verified |
| 3 | Flash address [7:0][1] of the first 128 bits to be verified |
| 4 | Number of 128 bits to be verified [15:8] |
| 5 | Number of 128 bits to be verified [7:0] |
| 6 | Read-1 Margin Choice |

1. Must be 128-bit aligned (Flash address [3:0] = 0000).

Upon clearing CCIF to launch the Read 1s Section command, the FTFE sets the read margin for 1s according to Table 29-38 and then reads all locations within the specified section of flash memory.

If the FTFE fails to read all 1s (i.e. the flash section is not erased), the FSTAT(MGSTAT0) bit is set. The CCIF flag sets after the Read 1s Section operation completes.

**Table 29-38. Margin Level Choices for Read 1s Section**

| Read Margin Choice | Margin Level Description |
|---|---|
| 0x00 | Use the 'normal' read level for 1s |
| 0x01 | Apply the 'User' margin to the normal read-1 level |
| 0x02 | Apply the 'Factory' margin to the normal read-1 level |

**Table 29-39. Read 1s Section Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| An invalid margin code is supplied | FSTAT[ACCERR] |
| An invalid flash address is supplied | FSTAT[ACCERR] |
| Flash address is not 128-bit aligned | FSTAT[ACCERR] |
| The requested section crosses a logical flash block boundary | FSTAT[ACCERR] |
| The requested number of 128 bits is zero | FSTAT[ACCERR] |
| Read-1s fails | FSTAT[MGSTAT0] |

### 29.4.12.3 Program Check command

The Program Check command tests a previously programmed program flash or data flash longword to see if it reads correctly at the specified margin level.

**Table 29-40. Program Check Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x02 (PGMCHK) |
| 1 | Flash address [23:16] |
| 2 | Flash address [15:8] |
| 3 | Flash address [7:0][1] |
| 4 | Margin Choice |
| 8 | Byte 0 expected data |
| 9 | Byte 1 expected data |
| A | Byte 2 expected data |
| B | Byte 3 expected data |

1. Must be longword aligned (Flash address [1:0] = 00).

Upon clearing CCIF to launch the Program Check command, the FTFE sets the read margin for 1s based on the provided margin choice according to Table 29-41. The Program Check operation then reads the specified longword, and compares the actual read data to the expected data provided by the FCCOB. If the comparison at margin-1 fails, the MGSTAT0 bit is set.

The FTFE will then set the read margin for 0s based on the provided margin choice.The Program Check operation will then read the specified longword and compare the actual read data to the expected data provided by the FCCOB. If the comparison at margin-0 fails, the MGSTAT0 bit will be set. The CCIF flag will set after the Program Check operation has completed.

The starting address must be longword aligned (the lowest two bits of the byte address must be 00):

- Byte 0 data is expected at the supplied 32-bit aligned address,
- Byte 1 data is expected at byte address specified + 0b01,
- Byte 2 data is expected at byte address specified + 0b10, and
- Byte 3 data is expected at byte address specified + 0b11.

**NOTE**

See the description of margin reads, Margin Read Commands

**Table 29-41. Margin Level Choices for Program Check**

| Read Margin Choice | Margin Level Description |
|---|---|
| 0x01 | Read at 'User' margin-1 and 'User' margin-0 |
| 0x02 | Read at 'Factory' margin-1 and 'Factory' margin-0 |

**Table 29-42. Program Check Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| An invalid flash address is supplied | FSTAT[ACCERR] |
| Flash address is not longword aligned | FSTAT[ACCERR] |
| An invalid margin choice is supplied | FSTAT[ACCERR] |
| Either of the margin reads does not match the expected data | FSTAT[MGSTAT0] |

### 29.4.12.4 Read Resource Command

The Read Resource command is provided for the user to read data from special-purpose memory resources located within the Flash module. The special-purpose memory resources available include program flash IFR, data flash IFR space, and the Version ID field. The Version ID field contains an 8 byte code that indicates a specific FTFE implementation.

**Table 29-43. Read Resource Command FCCOB Requirements**

| FCCOB Number | FCCOB contents [7:0] |
|---|---|
| 0 | 0x03 (RDRSRC) |
| 1 | Flash address [23:16] |
| 2 | Flash address [15:8] |
| 3 | Flash address [7:0][1] |
| 4 | Resource select code (see Table 29-44) |
| Returned values | |
| 4 | Read Data [64:56] |
| 5 | Read Data [55:48] |
| 6 | Read Data [47:40] |
| 7 | Read Data [39:32] |
| 8 | Read Data [31:24] |
| 9 | Read Data [23:16] |
| A | Read Data [15:8] |
| B | Read Data [7:0] |

1. Must be 64-bit aligned (Flash address [2:0] = 000).

**Table 29-44. Read Resource Select Codes**

| Resource Select Code | Description | Resource Size | Local Address Range |
|---|---|---|---|
| 0x00 | Program Flash 0 IFR | 1024 Bytes | 0x00_0000 - 0x00_03FF |
| 0x00 | Program Flash Swap IFR | 1024 Bytes | 0x04_0000 - 0x04_03FF |
| 0x00 | Data Flash 0 IFR | 1024 Bytes | 0x80_0000 - 0x80_03FF |
| 0x01 | Version ID | 8 Bytes | 0x00_0008 - 0x00_000F |

After clearing CCIF to launch the Read Resource command, eight consecutive bytes are read from the selected resource at the provided relative address and stored in the FCCOB register. The CCIF flag will set after the Read Resource operation has completed. The Read Resource command exits with an access error if an invalid resource code is provided or if the address for the applicable area is out-of-range.

**Table 29-45. Read Resource Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| An invalid resource code is entered | FSTAT[ACCERR] |
| Flash address is out-of-range for the targeted resource. | FSTAT[ACCERR] |
| Flash address is not 64-bit aligned | FSTAT[ACCERR] |

### 29.4.12.5 Program Phrase command

The Program Phrase command programs eight previously-erased bytes in the program flash memory or in the data flash memory using an embedded algorithm.

**CAUTION**

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

**Table 29-46. Program Phrase Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x07 (PGM8) |
| 1 | Flash address [23:16] |
| 2 | Flash address [15:8] |
| 3 | Flash address [7:0][1] |

*Table continues on the next page...*

**Table 29-46. Program Phrase Command FCCOB Requirements (continued)**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 4 | Byte 0 program value |
| 5 | Byte 1 program value |
| 6 | Byte 2 program value |
| 7 | Byte 3 program value |
| 8 | Byte 4 program value |
| 9 | Byte 5 program value |
| A | Byte 6 program value |
| B | Byte 7 program value |

1. Must be 64-bit aligned (Flash address [2:0] = 000)

Upon clearing CCIF to launch the Program Phrase command, the FTFE programs the data bytes into the flash using the supplied address. The protection status is always checked. The swap indicator address is implicitly protected from programming in NVM Normal and Special modes. The targeted flash locations must be currently unprotected (see the description of the FPROT registers) to permit execution of the Program Phrase operation.

The programming operation is unidirectional. It can only move NVM bits from the erased state ('1') to the programmed state ('0'). Erased bits that fail to program to the '0' state are flagged as errors in MGSTAT0. The CCIF flag is set after the Program Phrase operation completes.

The starting address must be 64-bit aligned (flash address [2:0] = 000):

- Byte 0 data is written to the starting address ('start'),
- Byte 1 data is programmed to byte address start+0b01,
- Byte 2 data is programmed to byte address start+0b10, and
- Byte 3 data is programmed to byte address start+0b11, etc.

**Table 29-47. Program Phrase Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| An invalid flash address is supplied | FSTAT[ACCERR] |
| Flash address is not 64-bit aligned | FSTAT[ACCERR] |
| Flash address points to a protected area | FSTAT[FPVIOL] |
| Any errors have been encountered during the verify operation. | FSTAT[MGSTAT0] |

### 29.4.12.6 Erase Flash Block Command

The Erase Flash Block operation erases all addresses in a single program flash or data flash block.

**Table 29-48. Erase Flash Block Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x08 (ERSBLK) |
| 1 | Flash address [23:16] in the flash block to be erased |
| 2 | Flash address [15:8] in the flash block to be erased |
| 3 | Flash address [7:0][1] in the flash block to be erased |

1. Must be 128-bit aligned (Flash address [3:0] = 0000).

Upon clearing CCIF to launch the Erase Flash Block command, the FTFE erases the main array of the selected flash block and verifies that it is erased. When the data flash is targeted, DEPART must be set for no EEPROM (see Table 29-4) else the Erase Flash Block command aborts setting the FSTAT[ACCERR] bit. The Erase Flash Block command aborts and sets the FSTAT[FPVIOL] bit if any region within the block is protected (see the description of the program flash protection (FPROT) registers and the data flash protection (FDPROT) registers). The swap indicator address is implicitly protected from block erase unless the swap system is in the update mode and the program flash block being erased is the non-active block that contains the swap indicator address. If the erase verify fails, the MGSTAT0 bit in FSTAT is set. The CCIF flag will set after the Erase Flash Block operation has completed.

**Table 29-49. Erase Flash Block Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| Program flash is selected and the address is out of program flash range | FSTAT[ACCERR] |
| Data flash is selected and the address is out of data flash range | FSTAT[ACCERR] |
| Data flash is selected with EEPROM enabled | FSTAT[ACCERR] |
| Flash address is not 128-bit aligned | FSTAT[ACCERR] |
| Any area of the selected flash block is protected | FSTAT[FPVIOL] |
| Any errors have been encountered during the verify operation[1] | FSTAT[MGSTAT0] |

1. User margin read may be run using the Read 1s Block command to verify all bits are erased.

### 29.4.12.7 Erase Flash Sector command

The Erase Flash Sector operation erases all addresses in a flash sector.

**Table 29-50. Erase Flash Sector Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x09 (ERSSCR) |
| 1 | Flash address [23:16] in the flash sector to be erased |
| 2 | Flash address [15:8] in the flash sector to be erased |
| 3 | Flash address [7:0][1] in the flash sector to be erased |

1. Must be 128-bit aligned (flash address [3:0] = 0000).

After clearing CCIF to launch the Erase Flash Sector command, the FTFE erases the selected program flash or data flash sector and then verifies that it is erased. The Erase Flash Sector command aborts if the selected sector is protected (see the description of the FPROT registers). The swap indicator address in each program flash block is implicitly protected from sector erase unless the swap system is in the update mode and the program flash sector containing the swap indicator address being erased is in the non-active block. If the erase-verify fails the FSTAT[MGSTAT0] bit is set. The CCIF flag is set after the Erase Flash Sector operation completes. The Erase Flash Sector command is suspendable (see the FCNFG[ERSSUSP] bit and Figure 29-35).

**Table 29-51. Erase Flash Sector Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| An invalid Flash address is supplied | FSTAT[ACCERR] |
| Flash address is not 128-bit aligned | FSTAT[ACCERR] |
| The selected program flash or data flash sector is protected | FSTAT[FPVIOL] |
| Any errors have been encountered during the verify operation[1] | FSTAT[MGSTAT0] |

1. User margin read may be run using the Read 1s Section command to verify all bits are erased.

### 29.4.12.7.1 Suspending an Erase Flash Sector Operation

To suspend an Erase Flash Sector operation set the FCNFG[ERSSUSP] bit (see Flash configuration field description) when CCIF is clear and the CCOB command field holds the code for the Erase Flash Sector command. During the Erase Flash Sector operation (see Erase Flash Sector command), the flash samples the state of the ERSSUSP bit at convenient points. If the FTFE detects that the ERSSUSP bit is set, the Erase Flash Sector operation is suspended and the FTFE sets CCIF. While ERSSUSP is set, all writes to flash registers are ignored except for writes to the FSTAT and FCNFG registers.

If an Erase Flash Sector operation effectively completes before the FTFE detects that a suspend request has been made, the FTFE clears the ERSSUSP bit prior to setting CCIF. When an Erase Flash Sector operation has been successfully suspended, the FTFE sets

CCIF and leaves the ERSSUSP bit set. While CCIF is set, the ERSSUSP bit can only be cleared to prevent the withdrawal of a suspend request before the FTFE has acknowledged it.

#### 29.4.12.7.2 Resuming a Suspended Erase Flash Sector Operation

If the ERSSUSP bit is still set when CCIF is cleared to launch the next command, the previous Erase Flash Sector operation resumes. The FTFE acknowledges the request to resume a suspended operation by clearing the ERSSUSP bit. A new suspend request can then be made by setting ERSSUSP. A single Erase Flash Sector operation can be suspended and resumed multiple times.

There is a minimum elapsed time limit between the request to resume the Erase Flash Sector operation (CCIF is cleared) and the request to suspend the operation again (ERSSUSP is set). This minimum time period is required to ensure that the Erase Flash Sector operation will eventually complete. If the minimum period is continually violated, i.e. the suspend requests come repeatedly and too quickly, no forward progress is made by the Erase Flash Sector algorithm. The resume/suspend sequence runs indefinitely without completing the erase.

#### 29.4.12.7.3 Aborting a Suspended Erase Flash Sector Operation

The user may choose to abort a suspended Erase Flash Sector operation by clearing the ERSSUSP bit prior to clearing CCIF for the next command launch. When a suspended operation is aborted, the FTFE starts the new command using the new FCCOB contents.

While FCNFG[ERSSUSP] is set, a write to the FlexRAM while FCNFG[EEERDY] is set clears ERSSUSP and aborts the suspended operation. The FlexRAM write operation is executed by the FTFE.

**Note**

Aborting the erase leaves the bitcells in an indeterminate, partially-erased state. Data in this sector is not reliable until a new erase command fully completes.

The following figure shows how to suspend and resume the Erase Flash Sector operation.

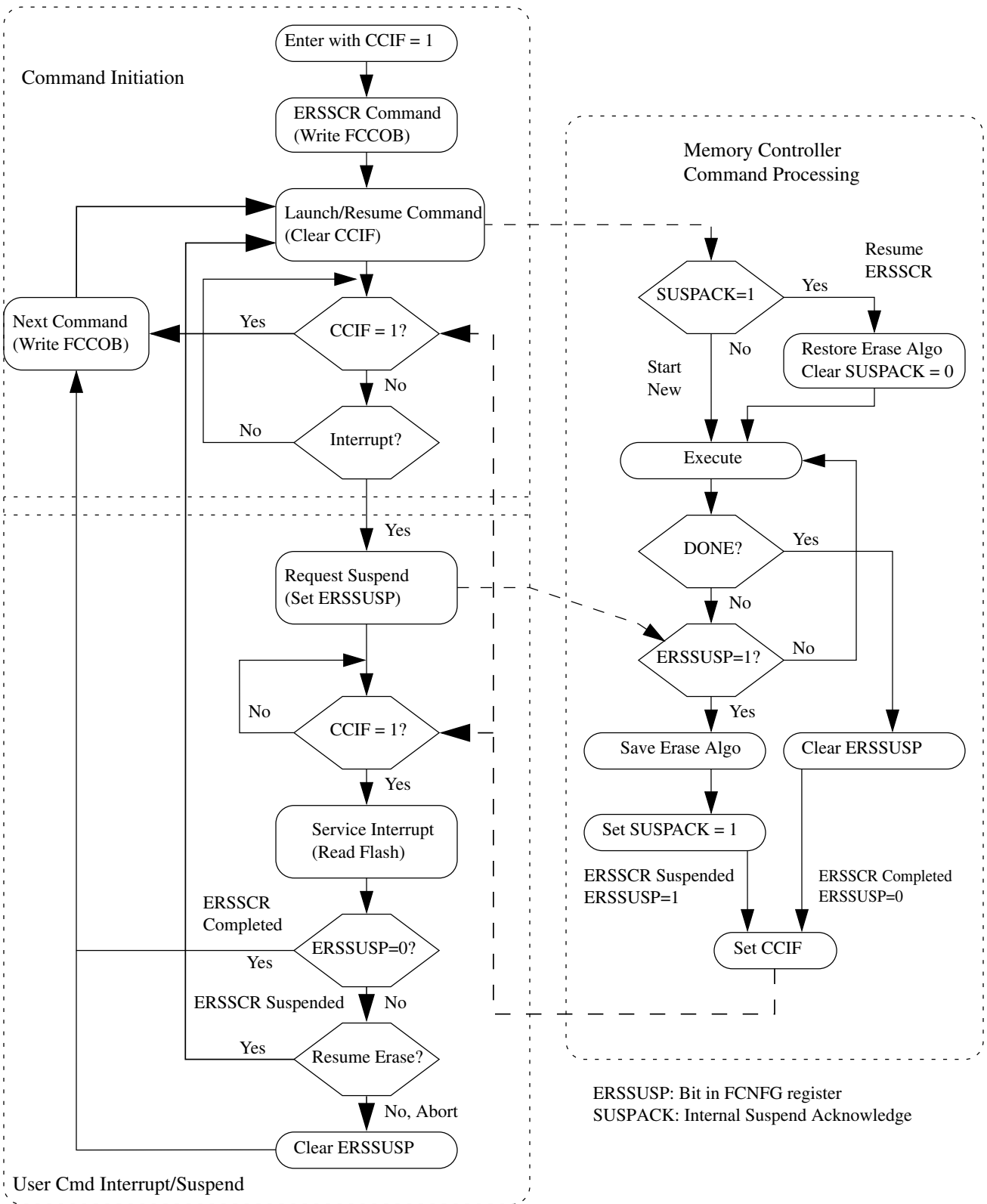



**Figure 29-35. Suspend and Resume of Erase Flash Sector Operation**

### 29.4.12.8 Program Section command

The Program Section operation programs the data found in the section program buffer to previously erased locations in the flash memory using an embedded algorithm. Data is preloaded into the section program buffer by writing to the FlexRAM while it is set to function as traditional RAM (see Flash sector programming).

The section program buffer is limited to the lower quarter of the FlexRAM (byte addresses 0x0000-0x03FF). Data written to the remainder of the FlexRAM is ignored and may be overwritten during Program Section command execution.

**CAUTION**

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

**Table 29-52. Program Section Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x0B (PGMSEC) |
| 1 | Flash address [23:16] |
| 2 | Flash address [15:8] |
| 3 | Flash address [7:0][1] |
| 4 | Number of 128 bits to program [15:8] |
| 5 | Number of 128 bits to program [7:0] |

1. Must be 128-bit aligned (Flash address [3:0] = 0000).

After clearing CCIF to launch the Program Section command, the FTFE will block access to the programming acceleration RAM (program flash only devices) or FlexRAM (FlexNVM devices) and program the data residing in the Section Program Buffer into the flash memory starting at the flash address provided.

The starting address must be unprotected (see the description of the FPROT registers) to permit execution of the Program Section operation. The swap indicator address in both program flash blocks is implicitly protected from programming. If the swap indicator address is encountered during the Program Section operation, it will be bypassed without setting FPVIOL and the contents will not be programmed. Programming, which is not allowed to cross a flash sector boundary, continues until all requested double-phrases have been programmed.

After the Program Section operation has completed, the CCIF flag will set and normal access to the FlexRAM is restored. The contents of the Section Program Buffer is not changed by the Program Section operation.

**Table 29-53. Program Section Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| An invalid flash address is supplied | FSTAT[ACCERR] |
| Flash address is not 128-bit aligned | FSTAT[ACCERR] |
| The requested section crosses a program flash sector boundary | FSTAT[ACCERR] |
| The requested number of double phrases is zero | FSTAT[ACCERR] |
| The space required to store data for the requested number of double phrases is more than one quarter the size of the programming acceleration RAM (program flash only devices) or FlexRAM (FlexNVM devices) | FSTAT[ACCERR] |
| The FlexRAM is not set to function as a traditional RAM, i.e. set if RAMRDY=0 | FSTAT[ACCERR] |
| The flash address falls in a protected area | FSTAT[FPVIOL] |
| Any errors have been encountered during the verify operation | FSTAT[MGSTAT0] |

#### 29.4.12.8.1 Flash sector programming

The process of programming an entire flash sector using the Program Section command is as follows:

1. If required, execute the Set FlexRAM Function command to make the FlexRAM available as traditional RAM and initialize the FlexRAM to all ones.
2. Launch the Erase Flash Sector command to erase the flash sector to be programmed.
3. Beginning with the starting address of the programming acceleration RAM (program flash only devices) or FlexRAM (FlexNVM devices), sequentially write enough data to the RAM to fill an entire flash sector. This area of the RAM serves as the section program buffer.

**NOTE**

In step 1, the section program buffer was initialized to all ones, the erased state of the flash memory.

The section program buffer can be written to while the operation launched in step 2 is executing, i.e. while CCIF = 0.

4. Execute the Program Section command to program the contents of the section program buffer into the selected flash sector.
5. To program additional flash sectors, repeat steps 2 through 4.
6. To restore EEPROM functionality, execute the Set FlexRAM Function command to make the FlexRAM available for EEPROM.

### 29.4.12.9 Read 1s All Blocks Command

The Read 1s All Blocks command checks if the program flash blocks, data flash blocks, EEPROM backup records, and data flash IFR have been erased to the specified read margin level, if applicable, and releases security if the readout passes, i.e. all data reads as '1'.

**Table 29-54. Read 1s All Blocks Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x40 (RD1ALL) |
| 1 | Read-1 Margin Choice |

After clearing CCIF to launch the Read 1s All Blocks command, the FTFE :

- sets the read margin for 1s according to Table 29-55,
- checks the contents of the program flash, data flash, EEPROM backup records, and data flash IFR are in the erased state.

If the FTFE confirms that these memory resources are erased, security is released by setting the FSEC[SEC] field to the unsecure state. The security byte in the flash configuration field (see Flash configuration field description) remains unaffected by the Read 1s All Blocks command. If the read fails, i.e. all flash memory resources are not in the fully erased state, the FSTAT[MGSTAT0] bit is set.

The EEERDY and RAMRDY bits are clear during the Read 1s All Blocks operation and are restored at the end of the Read 1s All Blocks operation.

The CCIF flag sets after the Read 1s All Blocks operation has completed.

**Table 29-55. Margin Level Choices for Read 1s All Blocks**

| Read Margin Choice | Margin Level Description |
|---|---|
| 0x00 | Use the 'normal' read level for 1s |
| 0x01 | Apply the 'User' margin to the normal read-1 level |
| 0x02 | Apply the 'Factory' margin to the normal read-1 level |

**Table 29-56. Read 1s All Blocks Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| An invalid margin choice is specified | FSTAT[ACCERR] |
| Read-1s fails | FSTAT[MGSTAT0] |

### 29.4.12.10 Read Once Command

The Read Once command provides read access to a reserved 64-byte field located in the program flash 0 IFR (see Program flash 0 IFR map and Program Once field). Access to this field is via 8 records, each 8 bytes long. The Read Once field is programmed using the Program Once command described in Program Once command.

**Table 29-57. Read Once Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x41 (RDONCE) |
| 1 | Read Once record index (0x00 - 0x07) |
| Returned Values | |
| 4 | Read Once byte 0 value |
| 5 | Read Once byte 1 value |
| 6 | Read Once byte 2 value |
| 7 | Read Once byte 3 value |
| 8 | Read Once byte 4 value |
| 9 | Read Once byte 5 value |
| A | Read Once byte 6 value |
| B | Read Once byte 7 value |

After clearing CCIF to launch the Read Once command, an 8-byte Read Once record is read from the program flash IFR and stored in the FCCOB register. The CCIF flag is set after the Read Once operation completes. Valid record index values for the Read Once command range from 0x00 to 0x07. During execution of the Read Once command, any attempt to read addresses within the program flash block containing this 64-byte field returns invalid data. The Read Once command can be executed any number of times.

**Table 29-58. Read Once Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| An invalid record index is supplied | FSTAT[ACCERR] |

### 29.4.12.11 Program Once command

The Program Once command enables programming to a reserved 64-byte field in the program flash 0 IFR (see Program flash 0 IFR map and Program Once field). Access to the Program Once field is via 8 records, each 8 bytes long. The Program Once field can

be read using the Read Once command (see Read Once Command) or using the Read Resource command (see Read Resource Command). Each Program Once record can be programmed only once since the program flash 0 IFR cannot be erased.

**Table 29-59. Program Once Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x43 (PGMONCE) |
| 1 | Program Once record index (0x00 - 0x0F) |
| 2 | Not Used |
| 3 | Not Used |
| 4 | Program Once Byte 0 value |
| 5 | Program Once Byte 1 value |
| 6 | Program Once Byte 2 value |
| 7 | Program Once Byte 3 value |
| 8 | Program Once Byte 4 value |
| 9 | Program Once Byte 5 value |
| A | Program Once Byte 6 value |
| B | Program Once Byte 7 value |

After clearing CCIF to launch the Program Once command, the FTFE first verifies that the selected record is erased. If erased, then the selected record is programmed using the values provided. The Program Once command also verifies that the programmed values read back correctly. The CCIF flag is set after the Program Once operation has completed.

The reserved program flash 0 IFR location accessed by the Program Once command cannot be erased and any attempt to program one of these records when the existing value is not Fs (erased) is not allowed. Valid record index values for the Program Once command range from 0x00 to 0x07. During execution of the Program Once command, any attempt to read addresses within program flash 0 returns invalid data.

**Table 29-60. Program Once Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| An invalid record index is supplied | FSTAT[ACCERR] |
| The requested record has already been programmed to a non-erased value[1] | FSTAT[ACCERR] |
| Any errors have been encountered during the verify operation. | FSTAT[MGSTAT0] |

1. If a Program Once record is initially programmed to 0xFFFF_FFFF_FFFF_FFFF, the Program Once command is allowed to execute again on that same record.

### 29.4.12.12 Erase All Blocks Command

The Erase All Blocks operation erases all flash memory, initializes the FlexRAM, verifies all memory contents, and releases MCU security.

**Table 29-61. Erase All Blocks Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x44 (ERSALL) |

After clearing CCIF to launch the Erase All Blocks command, the FTFE erases all program flash memory, program flash swap IFR space, data flash memory, data flash IFR space, EEPROM backup memory, and FlexRAM, then verifies that all are erased.

If the FTFE verifies that all flash memories and the FlexRAM were properly erased, security is released by setting the FSEC[SEC] field to the unsecure state and the FCNFG[RAMRDY] bit is set. The Erase All Blocks command aborts if any flash or FlexRAM region is protected. The swap indicator address in the program flash blocks are not implicitly protected from the erase operation. The security byte and all other contents of the flash configuration field (see Flash configuration field description) are erased by the Erase All Blocks command. If the erase-verify fails, the FSTAT[MGSTAT0] bit is set. The CCIF flag is set after the Erase All Blocks operation completes.

**Table 29-62. Erase All Blocks Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| Any region of the program flash memory, data flash memory, or FlexRAM is protected | FSTAT[FPVIOL] |
| Any errors have been encountered during the verify operation[1] | FSTAT[MGSTAT0] |

1. User margin read may be run using the Read 1s All Blocks command to verify all bits are erased.

#### 29.4.12.12.1 Triggering an erase all external to the flash module

The functionality of the Erase All Blocks command is also available in an uncommanded fashion outside of the flash memory. Refer to the device's Chip Configuration details for information on this functionality.

Before invoking the external erase all function, the FCCOB0 register must not contain 0x44. When invoked, the erase-all function erases all program flash memory, program flash swap IFR space, data flash memory, data flash IFR space, EEPROM backup, and FlexRAM regardless of the state of the FSTAT[ACCERR and FPVIOL] flags or the protection settings or the state of the flash swap system. If the post-erase verify passes, the routine then releases security by setting the FSEC[SEC] field register to the unsecure state and the FCNFG[RAMRDY] bit sets. The security byte in the Flash Configuration

Field is also programmed to the unsecure state. The status of the erase-all request is reflected in the FCNFG[ERSAREQ] bit. The FCNFG[ERSAREQ] bit is cleared once the operation completes and the normal FSTAT error reporting is available as described in Erase All Blocks Command.

**CAUTION**

Since the IFR Swap Field in the program flash swap IFR containing the swap indicator address is erased during the Erase All Blocks command operation, the swap system becomes uninitialized. The Swap Control command must be run with the initialization code to set the swap indicator address and initialize the swap system.

### 29.4.12.13 Verify Backdoor Access Key command

The Verify Backdoor Access Key command only executes if the mode and security conditions are satisfied (see Flash commands by mode). Execution of the Verify Backdoor Access Key command is further qualified by the FSEC[KEYEN] bits. The Verify Backdoor Access Key command releases security if user-supplied keys in the FCCOB match those stored in the Backdoor Comparison Key bytes of the Flash Configuration Field. The column labeled Flash Configuration Field offset address shows the location of the matching byte in the Flash Configuration Field.

**Table 29-63. Verify Backdoor Access Key Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] | Flash Configuration Field Offset Address |
|---|---|---|
| 0 | 0x45 (VFYKEY) | |
| 1-3 | Not Used | |
| 4 | Key Byte 0 | 0x0_0000 |
| 5 | Key Byte 1 | 0x0_0001 |
| 6 | Key Byte 2 | 0x0_0002 |
| 7 | Key Byte 3 | 0x0_0003 |
| 8 | Key Byte 4 | 0x0_0004 |
| 9 | Key Byte 5 | 0x0_0005 |
| A | Key Byte 6 | 0x0_0006 |
| B | Key Byte 7 | 0x0_0007 |

After clearing CCIF to launch the Verify Backdoor Access Key command, the FTFE checks the FSEC[KEYEN] bits to verify that this command is enabled. If not enabled, the FTFE sets the FSTAT[ACCERR] bit and terminates. If the command is enabled, the FTFE compares the key provided in FCCOB to the backdoor comparison key in the Flash

Configuration Field. If the backdoor keys match, the FSEC[SEC] field is changed to the unsecure state and security is released. If the backdoor keys do not match, security is not released and all future attempts to execute the Verify Backdoor Access Key command are immediately aborted and the FSTAT[ACCERR] bit is (again) set to 1 until a reset of the FTFE module occurs. If the entire 8-byte key is all zeros or all ones, the Verify Backdoor Access Key command fails with an access error. The CCIF flag is set after the Verify Backdoor Access Key operation completes.

**Table 29-64. Verify Backdoor Access Key Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| The supplied key is all-0s or all-Fs | FSTAT[ACCERR] |
| An incorrect backdoor key is supplied | FSTAT[ACCERR] |
| Backdoor key access has not been enabled (see the description of the FSEC register) | FSTAT[ACCERR] |
| This command is launched and the backdoor key has mismatched since the last power down reset | FSTAT[ACCERR] |

### 29.4.12.14 Swap Control command (program flash only devices)

The Swap Control command handles specific activities associated with swapping the two halves of program flash memory within the memory map.

**Table 29-65. Swap Control Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x46 (SWAP) |
| 1 | Flash address [23:16] |
| 2 | Flash address [15:8] |
| 3 | Flash address [7:0][1] |
| 4 | Swap Control Code:<br>0x01 - Initialize Swap System<br>0x02 - Set Swap in Update State<br>0x04 - Set Swap in Complete State<br>0x08 - Report Swap Status |
| Returned values | |

*Table continues on the next page...*

**Table 29-65. Swap Control Command FCCOB Requirements (continued)**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 5 | Current Swap Mode:<br>0x00 - Uninitialized<br>0x01 - Ready<br>0x02 - Update<br>0x03 - Update-Erased<br>0x04 - Complete |
| 6 | Current Swap Block Status:<br>For devices with FlexNVM:<br>0x00 - Program flash block 0 at 0x0_0000<br>0x01 - Program flash block 1 at 0x0_0000<br>For devices with program flash only:<br>0x00 - Program flash block 0 at 0x0_0000<br>0x01 - Program flash block 1 at 0x0_0000 |
| 7 | Next Swap Block Status (after any reset):<br>For devices with FlexNVM:<br>0x00 - Program flash block 0 at 0x0_0000<br>0x01 - Program flash block 1 at 0x0_0000<br>For devices with program flash only:<br>0x00 - Program flash block 0 at 0x0_0000<br>0x01 - Program flash block 1 at 0x0_0000 |

1. Must be 128-bit aligned (Flash address [3:0] = 0000).

Upon clearing CCIF to launch the Swap Control command, the FTFE will handle swap-related activities based on the Swap Control code provided in FCCOB4 as follows:

- 0x01 (Initialize Swap System to UPDATE-ERASED State) - After verifying that the current swap state is UNINITIALIZED and that the flash address provided is in Program flash block 0 but not in the Flash Configuration Field, the flash address (shifted with bits[3:0] removed) will be programmed into the IFR Swap Field found in the program flash swap IFR. After the swap indicator address has been programmed into the IFR Swap Field, the swap enable word will be programmed to 0x0000. After the swap enable word has been programmed, the swap indicator, located within the Program flash block 0 address provided, will be programmed to 0xFF00.
- 0x02 (Progress Swap to UPDATE State) - After verifying that the current swap state is READY and that the aligned flash address provided matches the one stored in the IFR Swap Field, the swap indicator located within bits [15:0] of the flash address in the currently active program flash block will be programmed to 0xFF00.

- 0x04 (Progress Swap to COMPLETE State) - After verifying that the current swap state is UPDATE-ERASED and that the aligned flash address provided matches the one stored in the IFR Swap Field, the swap indicator located within bits [15:0] of the flash address in the currently active program flash block will be programmed to 0x0000. Before executing with this Swap Control code, the user must erase the non-active swap indicator using the Erase Flash Block or Erase Flash Sector commands and update the application code or data as needed. The non-active swap indicator will be checked at the erase verify level and if the check fails, the current swap state will be changed to UPDATE with ACCERR set.
- 0x08 (Report Swap Status) - After verifying that the aligned flash address provided is in program flash block 0 but not in the Flash Configuration Field, the status of the swap system will be reported as follows:
  - FCCOB5 (Current Swap State) - indicates the current swap state based on the status of the swap enable phrase and the swap indicators. If the MGSTAT0 flag is set after command completion, the swap state returned was not successfully transitioned from and the appropriate swap command code must be attempted again. If the current swap state is UPDATE and the non-active swap indicator is 0xFFFF, the current swap state is changed to UPDATE-ERASED.
  - FCCOB6 (Current Swap Block Status) - indicates which program flash block is currently located at relative flash address 0x0_0000.
  - FCCOB7 (Next Swap Block Status) - indicates which program flash block will be located at relative flash address 0x0_0000 after the next reset of the FTFE module.

**NOTE**

It is recommended that the user execute the Swap Control command to report swap status (code 0x08) after any reset to determine if issues with the swap system were detected during the swap state determination procedure.

**NOTE**

It is recommended that the user write 0xFF to FCCOB5, FCCOB6, and FCCOB7 since the Swap Control command will not always return the swap state and status fields when an ACCERR is detected.

The CCIF flag is set after the Swap Control operation has completed.

The swap indicators are implicitly protected from being programmed during Program Phrase or Program Section command operations and are implicitly unprotected during Swap Control command operations. The swap indicators are implicitly protected from being erased during Erase Flash Block and Erase Flash Sector command operations unless the swap indicator being erased is in the non-active program flash block and the

swap system is in the UPDATE or UPDATE-ERASED state. Once the swap system has been initialized, the Erase All Blocks command can be used to uninitialize the swap system.

**Table 29-66. Swap Control Command Error Handling**

| Error Condition | Swap Control Code | Error Bit |
|---|---|---|
| Command not available in current mode/security[1] | All | FSTAT[ACCERR] |
| Flash address is not in program flash block 0 | All | FSTAT[ACCERR] |
| Flash address is in the Flash Configuration Field | All | FSTAT[ACCERR] |
| Flash address is not 128-bit aligned | All | FSTAT[ACCERR] |
| Flash address does not match the swap indicator address in the IFR | 2, 4 | FSTAT[ACCERR] |
| Swap initialize requested when swap system is not in the uninitialized state | 1 | FSTAT[ACCERR] |
| Swap update requested when swap system is not in the ready state | 2 | FSTAT[ACCERR] |
| Swap complete requested when swap system is not in the update-erased state | 4 | FSTAT[ACCERR] |
| An undefined swap control code is provided | - | FSTAT[ACCERR] |
| Any errors have been encountered during the swap determination and program-verify operations | 1, 2, 4 | FSTAT[MGSTAT0] |
| Any brownouts were detected during the swap determination procedure | 8 | FSTAT[MGSTAT0] |

1. Returned fields will not be updated, i.e. no swap state or status reporting

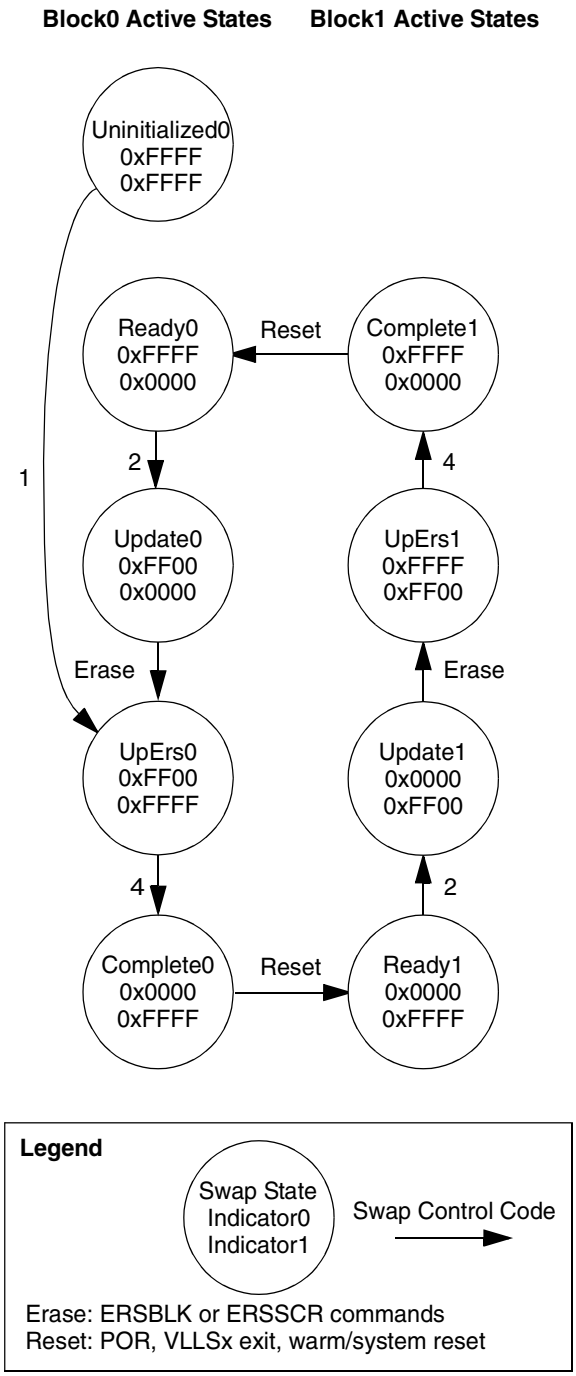



**Figure 29-36. Valid Swap State Sequencing**

## Table 29-67. Swap State Report Mapping

| Case | Swap Enable Field[1] | Swap Indicator 0[1] | Swap Indicator 1[1] | Swap State[2] | State Code | MGSTAT0 | Active Block |
|---|---|---|---|---|---|---|---|
| 1 | 0xFFFF | - | - | Uninitialized | 0 | 0 | 0 |
| 2 | 0x0000 | 0xFF00 | 0x0000 | Update | 2 | 0 | 0 |
| 3 | 0x0000 | 0xFF00- | 0xFFFF | Update-Erased | 3 | 0 | 0 |
| 4 | 0x0000 | 0x0000 | 0xFFFF[3] | Complete[4] | 4 | 0 | 0 |
| 5 | 0x0000 | 0x0000 | 0xFFFF | Ready[5] | 1 | 0 | 1 |
| 6 | 0x0000 | 0x0000 | 0xFF00 | Update | 2 | 0 | 1 |
| 7 | 0x0000 | 0xFFFF | 0xFF00 | Update-Erased | 3 | 0 | 1 |
| 8 | 0x0000 | 0xFFFF[3] | 0x0000 | Complete[4] | 4 | 0 | 1 |
| 9 | 0x0000 | 0xFFFF | 0x0000 | Ready[5] | 1 | 0 | 0 |
| 10 | 0xXXXX | - | - | Uninitialized | 0 | 1 | 0 |
| 11 | 0x0000 | 0xFFFF | 0xFFFF | Uninitialized | 0 | 1 | 0 |
| 12 | 0x0000 | 0xFFXX | 0xFFFF | Ready | 1 | 1 | 0 |
| 13 | 0x0000 | 0xFFXX | 0x0000 | Ready | 1 | 1 | 0 |
| 14[6] | 0x0000 | 0xXXXX | 0x0000 | Ready | 1 | 1 | 0 |
| 15[6] | 0x0000 | 0xFFFF | 0xFFXX | Ready | 1 | 1 | 1 |
| 16 | 0x0000 | 0x0000 | 0xFFXX | Ready | 1 | 1 | 1 |
| 17[6] | 0x0000 | 0x0000 | 0xXXXX | Ready | 1 | 1 | 1 |
| 18 | 0x0000 | 0xFF00 | 0xFFFF[7] | Update | 2 | 1 | 0 |
| 19 | 0x0000 | 0xFF00 | 0xXXXX | Update | 2 | 1 | 0 |
| 20 | 0x0000 | 0xFF(00) | 0xFFXX | Update | 2 | 1 | 0 |
| 21[6] | 0x0000 | 0x0000 | 0x0000 | Update | 2 | 1 | 0 |
| 22[6] | 0x0000 | 0xXXXX | 0xXXXX | Update | 2 | 1 | 0 |
| 23 | 0x0000 | 0xFFFF[7] | 0xFF00 | Update | 2 | 1 | 1 |
| 24 | 0x0000 | 0xXXXX | 0xFF00 | Update | 2 | 1 | 1 |
| 25 | 0x0000 | 0xFFXX | 0xFF(00) | Update | 2 | 1 | 1 |
| 26 | 0x0000 | 0xXX00 | 0xFFFF | Update-Erased | 3 | 1 | 0 |
| 27 | 0x0000 | 0xXXXX | 0xFFFF | Update-Erased | 3 | 1 | 0 |
| 28 | 0x0000 | 0xFFFF | 0xXX00 | Update-Erased | 3 | 1 | 1 |
| 29 | 0x0000 | 0xFFFF | 0xXXXX | Update-Erased | 3 | 1 | 1 |

1. 0xXXXX, 0xFFXX, 0xXX00 indicates a non-valid value was read; 0xFF(00) indicates more 0's than other indicator (if same number of 0's, then swap system defaults to block 0 active)
2. Cases 10-29 due to brownout (abort) detected during program or erase steps related to swap
3. Must read 0xFFFF with erase verify level before transition to Complete allowed
4. No reset since successful Swap Complete execution
5. Reset after successful Swap Complete execution
6. Not a valid case
7. Fails to read 0xFFFF at erase verify level

#### 29.4.12.14.1 Swap state determination

During the reset sequence, the state of the swap system is determined by evaluating the IFR Swap Field in the program flash swap IFR and both swap indicators located in the program flash blocks at the swap indicator address stored in the IFR Swap Field.

**Table 29-68. Program Flash Swap IFR Fields**

| Address Range | Size (Bytes) | Field Description |
|---|---|---|
| 0x000 – 0x001 | 2 | Swap Indicator Address |
| 0x002 – 0x003 | 2 | Swap Enable Word |
| 0x004 – 0x3FF | 1020 | Reserved |

### 29.4.12.15 Program Partition command

The Program Partition command prepares the FlexNVM block for use as data flash, EEPROM backup, or a combination of both and initializes the FlexRAM. The Program Partition command must not be launched from flash memory, since flash memory resources are not accessible during Program Partition command execution.

**CAUTION**

While different partitions of the FlexNVM are available, the intention is that a single partition choice is used throughout the entire lifetime of a given application. The FlexNVM Partition Code choices affect the endurance and data retention characteristics of the device.

**Table 29-69. Program Partition Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x80 (PGMPART) |
| 1 | Not Used |
| 2 | Not Used |
| 3 | Not Used |
| 4 | EEPROM Data Size Code[1] |
| 5 | FlexNVM Partition Code[2] |

1. See Table 29-70 and EEPROM Data Set Size
2. See Table 29-71 and FlexNVM partition code

## Table 29-70. Valid EEPROM Data Set Size Codes

| EEPROM Data Set Size Code (FCCOB4)[1] | | EEPROM Data Set Size (Bytes) Subsystem A + B |
|---|---|---|
| EEESPLIT (FCCOB4[5:4)] | EEESIZE (FCCOB4[3:0]) | |
| 11 | 0xF | 0[2] |
| 01 | 0x9 | 8 + 24 |
| 10 | | 16 + 16 |
| 11 | | 16 + 16 |
| 00 | 0x8 | 8 + 56 |
| 01 | | 16 + 48 |
| 10 | | 32 + 32 |
| 11 | | 32 + 32 |
| 00 | 0x7 | 16 + 112 |
| 01 | | 32 + 96 |
| 10 | | 64 + 64 |
| 11 | | 64 + 64 |
| 00 | 0x6 | 32 + 224 |
| 01 | | 64 + 192 |
| 10 | | 128 + 128 |
| 11 | | 128 + 128 |
| 00 | 0x5 | 64 + 448 |
| 01 | | 128 + 384 |
| 10 | | 256 + 256 |
| 11 | | 256 + 256 |
| 00 | 0x4 | 128 + 896 |
| 01 | | 256 + 768 |
| 10 | | 512 + 512 |
| 11 | | 512 + 512 |
| 00 | 0x3 | 256 + 1,792 |
| 01 | | 512 + 1,536 |
| 10 | | 1,024 + 1,024 |
| 11 | | 1,024 + 1,024 |
| 00 | 0x2 | 512 + 3,584 |
| 01 | | 1,024 + 3,072 |
| 10 | | 2,048 + 2,048 |
| 11 | | 2,048 + 2,048 |

1. FCCOB4[7:6] = 00
2. EEE Data Set Size must be set to 0 Bytes when the FlexNVM Partition Code is set for no EEPROM.

**Table 29-71. Valid FlexNVM Partition Codes**

| FlexNVM Partition Code DEPART (FCCOB5[3:0])[1] | Data flash Size (Kbytes) | EEPROM-backup Size (Kbytes) |
|---|---|---|
| 0000 | 128 | 0 |
| 0011 | 96 | 32 |
| 0100 | 64 | 64 |
| 0101 | 0 | 128 |
| 1000 | 0 | 128 |
| 1011 | 32 | 96 |
| 1100 | 64 | 64 |
| 1101 | 128 | 0 |

1. FCCOB5[7:4] = 0000

After clearing CCIF to launch the Program Partition command, the FTFE first verifies that the EEPROM Data Size Code and FlexNVM Partition Code in the data flash IFR are erased. If erased, the Program Partition command erases the contents of the FlexNVM memory. If the FlexNVM is to be partitioned for EEPROM backup, the allocated EEPROM backup sectors are formatted for EEPROM use. Finally, the partition codes are programmed into the data flash IFR using the values provided. The Program Partition command also verifies that the partition codes read back correctly after programming. If the FlexNVM is partitioned for EEPROM, the allocated EEPROM backup sectors are formatted for EEPROM use. The CCIF flag is set after the Program Partition operation completes.

Prior to launching the Program Partition command, the data flash IFR must be in an erased state, which can be accomplished by executing the Erase All Blocks command or by an external request (see Erase All Blocks Command). The EEPROM Data Size Code and FlexNVM Partition Code are read using the Read Resource command (see Read Resource Command).

**Table 29-72. Program Partition Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| The EEPROM data size and FlexNVM partition code bytes are not initially 0xFFFF | FSTAT[ACCERR] |
| Invalid EEPROM Data Size Code is entered (see Table 29-70 for valid codes) | FSTAT[ACCERR] |
| Invalid FlexNVM Partition Code is entered (see Table 29-71 for valid codes) | FSTAT[ACCERR] |
| FlexNVM Partition Code = full data flash (no EEPROM) and EEPROM Data Size Code allocates FlexRAM for EEPROM | FSTAT[ACCERR] |
| FlexNVM Partition Code allocates space for EEPROM backup, but EEPROM Data Size Code allocates no FlexRAM for EEPROM | FSTAT[ACCERR] |
| FCCOB4[7:6] != 00 | FSTAT[ACCERR] |
| FCCOB5[7:4] != 0000 | FSTAT[ACCERR] |
| Any errors have been encountered during the verify operation | FSTAT[MGSTAT0] |

### 29.4.12.16 Set FlexRAM Function command

The Set FlexRAM Function command changes the function of the FlexRAM:

- When not partitioned for EEPROM, the FlexRAM is typically used as traditional RAM.
- When partitioned for EEPROM, the FlexRAM is typically used to store EEPROM data.

**Table 29-73. Set FlexRAM Function Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x81 (SETRAM) |
| 1 | FlexRAM Function Control Code<br>(see Table 29-74) |

**Table 29-74. FlexRAM Function Control**

| FlexRAM Function Control Code | Action |
|---|---|
| 0xFF | Make FlexRAM available as RAM:<br>• Clear the FCNFG[RAMRDY] and FCNFG[EEERDY] flags<br>• Write a background of ones to all FlexRAM locations<br>• Set the FCNFG[RAMRDY] flag |
| 0x00 | Make FlexRAM available for EEPROM:<br>• Clear the FCNFG[RAMRDY] and FCNFG[EEERDY] flags<br>• Write a background of ones to all FlexRAM locations<br>• Copy-down existing EEPROM data to FlexRAM<br>• Set the FCNFG[EEERDY] flag |

After clearing CCIF to launch the Set FlexRAM Function command, the FTFE sets the function of the FlexRAM based on the FlexRAM Function Control Code.

When making the FlexRAM available as traditional RAM, the FTFE clears the FCNFG[EEERDY] and FCNFG[RAMRDY] flags, overwrites the contents of the entire FlexRAM with a background pattern of all ones, and sets the FCNFG[RAMRDY] flag. The state of the EPROT register does not prevent the FlexRAM from being overwritten. When the FlexRAM is set to function as a RAM, normal read and write accesses to the FlexRAM are available. When large sections of flash memory need to be programmed, e.g. during factory programming, the FlexRAM can be used as the Section Program Buffer for the Program Section command (see Program Section command).

When making the FlexRAM available for EEPROM, the FTFE clears the FCNFG[RAMRDY] and FCNFG[EEERDY] flags, overwrites the contents of the FlexRAM allocated for EEPROM with a background pattern of all ones, and copies the existing EEPROM data from the EEPROM backup record space to the FlexRAM. After completion of the EEPROM copy-down, the FCNFG[EEERDY] flag is set. When the FlexRAM is set to function as EEPROM, normal read and write access to the FlexRAM is available, but writes to the FlexRAM also invoke EEPROM activity.

The CCIF flag will be set after the Set FlexRAM Function operation has completed.

**Table 29-75. Set FlexRAM Function Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| FlexRAM Function Control Code is not defined | FSTAT[ACCERR] |
| FlexRAM Function Control Code is set to make the FlexRAM available for EEPROM, but FlexNVM is not partitioned for EEPROM | FSTAT[ACCERR] |

## 29.4.13 Security

The FTFE module provides security information to the MCU based on contents of the FSEC security register. The MCU then limits access to FTFE resources as defined in the device's Chip Configuration details. During reset, the FTFE module initializes the FSEC register using data read from the security byte of the Flash Configuration Field (see Flash configuration field description).

The following fields are available in the FSEC register. Details of the settings are described in the FSEC register description.

**Table 29-76. FSEC fields**

| FSEC field | Description |
|---|---|
| KEYEN | Backdoor Key Access |
| MEEN | Mass Erase Capability |
| FSLACC | Freescale Factory Access |
| SEC | MCU security |

### 29.4.13.1 FTFE Access by Mode and Security

The following table summarizes how access to the FTFE module is affected by security and operating mode.

**Table 29-77. FTFE Access Summary**

| Operating Mode | MCU Security State | |
|---|---|---|
| | Unsecure | Secure |
| NVM Normal | Full command set | |
| NVM Special | Full command set | Only the Erase All Blocks and Read 1s All Blocks commands. |

### 29.4.13.2 Changing the Security State

The security state out of reset can be permanently changed by programming the security byte of the flash configuration field. This assumes that you are starting from a mode where the necessary program flash erase and program commands are available and that the region of the program flash containing the flash configuration field is unprotected. If the flash security byte is successfully programmed, its new value takes affect after the next MCU reset.

#### 29.4.13.2.1 Unsecuring the MCU Using Backdoor Key Access

The MCU can be unsecured by using the backdoor key access feature which requires knowledge of the contents of the 8-byte backdoor key value stored in the Flash Configuration Field (see Flash configuration field description). If the FSEC[KEYEN] bits are in the enabled state, the Verify Backdoor Access Key command (see Verify Backdoor Access Key command) can be run which allows the user to present prospective keys for comparison to the stored keys. If the keys match, the FSEC[SEC] bits are changed to unsecure the MCU. The entire 8-byte key cannot be all 0s or all 1s, i.e. 0x0000_0000_0000_0000 and 0xFFFF_FFFF_FFFF_FFFF are not accepted by the Verify Backdoor Access Key command as valid comparison values. While the Verify Backdoor Access Key command is active, program flash memory is not available for read access and returns invalid data.

The user code stored in the program flash memory must have a method of receiving the backdoor keys from an external stimulus. This external stimulus would typically be through one of the on-chip serial ports.

If the KEYEN bits are in the enabled state, the MCU can be unsecured by the backdoor key access sequence described below:

1. Follow the command sequence for the Verify Backdoor Access Key command as explained in Verify Backdoor Access Key command

2. If the Verify Backdoor Access Key command is successful, the MCU is unsecured and the FSEC[SEC] bits are forced to the unsecure state

An illegal key provided to the Verify Backdoor Access Key command prohibits future use of the Verify Backdoor Access Key command. A reset of the MCU is the only method to re-enable the Verify Backdoor Access Key command when a comparison fails.

After the backdoor keys have been correctly matched, the MCU is unsecured by changing the FSEC[SEC] bits. A successful execution of the Verify Backdoor Access Key command changes the security in the FSEC register only. It does not alter the security byte or the keys stored in the Flash Configuration Field (Flash configuration field description). After the next reset of the MCU, the security state of the FTFE module reverts back to the Flash security byte in the Flash Configuration Field. The Verify Backdoor Access Key command sequence has no effect on the program and erase protections defined in the program flash protection registers.

If the backdoor keys successfully match, the unsecured MCU has full control of the contents of the Flash Configuration Field. The MCU may erase the sector containing the Flash Configuration Field and reprogram the flash security byte to the unsecure state and change the backdoor keys to any desired value.

### 29.4.14 Reset Sequence

On each system reset the FTFE module executes a sequence which establishes initial values for the flash block configuration parameters, FPROT, FDPROT, FEPROT, FOPT, and FSEC registers and the FCNFG[SWAP, PFLSH, RAMRDY, EEERDY] bits.

CCIF is cleared throughout the reset sequence. The FTFE module holds off all CPU access for a portion of the reset sequence. Flash reads are possible when the hold is removed. Completion of the reset sequence is marked by setting CCIF which enables flash user commands.

If a reset occurs while any FTFE command is in progress, that command is immediately aborted. The state of the word being programmed or the sector/block being erased is not guaranteed. Commands and operations do not automatically resume after exiting reset.

# Chapter 30
# EzPort

## 30.1 Overview

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The EzPort module is a serial flash programming interface that enables In-System Programming (ISP) of flash memory contents in a 32-bit general-purpose microcontroller. Memory contents can be read/erased/programmed from an external source, in a format that is compatible with many standalone flash memory chips, without requiring the removal of the microcontroller from the system board.

### 30.1.1 Block diagram

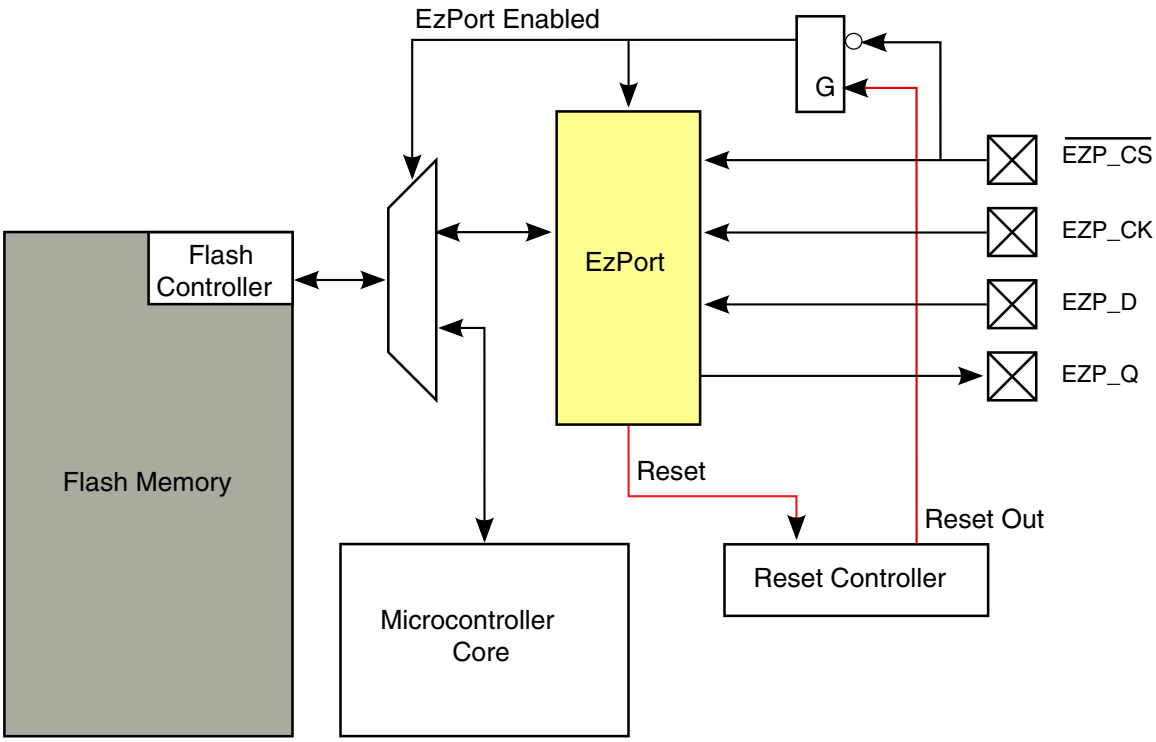



**Figure 30-1. EzPort block diagram**

### 30.1.2 Features

EzPort includes the following features:

- Serial interface that is compatible with a subset of the SPI format.
- Ability to read, erase, and program flash memory.
- Ability to reset the microcontroller, allowing it to boot from the flash memory after the memory has been configured.

### 30.1.3 Modes of operation

The EzPort can operate in one of two modes, enabled or disabled.

- Enabled — When enabled, the EzPort steals access to the flash memory, preventing access from other cores or peripherals. The rest of the microcontroller is disabled to avoid conflicts. The flash is configured for NVM Special mode.
- Disabled — When the EzPort is disabled, the rest of the microcontroller can access flash memory as normal.

The EzPort provides a simple interface to connect an external device to the flash memory on board a 32 bit microcontroller. The interface itself is compatible with the SPI interface, with the EzPort operating as a slave, running in either of the two following modes. The data is transmitted with the most significant bit first.

- CPOL = 0, CPHA = 0
- CPOL = 1, CPHA = 1

Commands are issued by the external device to erase, program, or read the contents of the flash memory. The serial data out from the EzPort is tri-stated unless data is being driven. This allows the signal to be shared among several different EzPort (or compatible) devices in parallel, as long as they have different chip-selects.

## 30.2 External signal descriptions

After the table of EzPort external signals, susbsequent sections explain the signals in more detail.

**Table 30-1. EzPort external signals**

| External Signal | Name | I/O |
|---|---|---|
| EZP_CK | EzPort Clock | Input |
| $\overline{\text{EZP\_CS}}$ | EzPort Chip Select | Input |
| EZP_D | EzPort Serial Data In | Input |
| EZP_Q | EzPort Serial Data Out | Output |

### 30.2.1 EzPort Clock (EZP_CK)

EZP_CK is the serial clock for data transfers. The serial data in (EZP_D) and chip select ($\overline{\text{EZP\_CS}}$) are registered on the rising edge of EZP_CK, while serial data out (EZP_Q) is driven on the falling edge of EZP_CK.

The maximum frequency of the EzPort clock is half the system clock frequency for all commands, except when executing the Read Data or Read FlexRAM commands. When executing the Read Data or Read FlexRAM commands, the EzPort clock has a maximum frequency of 1/8 the system clock frequency.

### 30.2.2 EzPort Chip Select (EZP_CS)

EZP_CS is the chip select for signaling the start and end of serial transfers. While EZP_CS is asserted, if the microcontroller's reset out signal is negated, then EzPort is enabled out of reset; otherwise EzPort is disabled. After EzPort is enabled, asserting EZP_CS starts a serial data transfer, which continues until EZP_CS is negated again. The negation of EZP_CS indicates that the current command has finished and resets the EzPort state machine, so that EzPort is ready to receive the next command.

### 30.2.3 EzPort Serial Data In (EZP_D)

EZP_D is the serial data in for data transfers. EZP_D is registered on the rising edge of EZP_CK. All commands, addresses, and data are shifted in most significant bit first. When the EzPort is driving output data on EZP_Q, the data shifted in EZP_D is ignored.

### 30.2.4 EzPort Serial Data Out (EZP_Q)

EZP_Q is the serial data out for data transfers. EZP_Q is driven on the falling edge of EZP_CK. It is tri-stated unless EZP_CS is asserted and the EzPort is driving data out. All data is shifted out most significant bit first.

## 30.3 Command definition

The EzPort receives commands from an external device and translates the commands into flash memory accesses. The following table lists the supported commands.

**Table 30-2. EzPort commands**

| Command | Description | Code | Address Bytes | Data Bytes | Accepted when secure? |
|---|---|---|---|---|---|
| WREN | Write Enable | 0x06 | 0 | 0 | Yes |
| WRDI | Write Disable | 0x04 | 0 | 0 | Yes |
| RDSR | Read Status Register | 0x05 | 0 | 1 | Yes |

*Table continues on the next page...*

**Table 30-2. EzPort commands (continued)**

| Command | Description | Code | Address Bytes | Data Bytes | Accepted when secure? |
|---|---|---|---|---|---|
| READ | Flash Read Data | 0x03 | 3[1] | 1+ | No |
| FAST_READ | Flash Read Data at High Speed | 0x0B | 3[1] | 1+[2] | No |
| SP | Flash Section Program | 0x02 | 3[3] | 416 - SECTION[4] | No |
| SE | Flash Sector Erase | 0xD8 | 3[3] | 0 | No |
| BE | Flash Bulk Erase | 0xC7 | 0 | 0 | Yes[5] |
| RESET | Reset Chip | 0xB9 | 0 | 0 | Yes |
| WRFCCOB | Write FCCOB Registers | 0xBA | 0 | 12 | Yes[6] |
| FAST_RDFCCOB | Read FCCOB registers at high speed | 0xBB | 0 | 1 - 12[2] | No |
| WRFLEXRAM | Write FlexRAM | 0xBC | 3[1] | 4 | No |
| RDFLEXRAM | Read FlexRAM | 0xBD | 3[1] | 1+ | No |
| FAST_RDFLEXRAM | Read FlexRAM at high speed | 0xBE | 3[1] | 1+[2] | No |

1. Address must be 32-bit aligned (two LSBs must be zero).
2. One byte of dummy data must be shifted in before valid data is shifted out.
3. Address must be 32-bit aligned (two LSBs must be zero).128-bit aligned (four LSBs must be zero).
4. Please see the Flash Memory chapter for a definition of section size. Total number of data bytes programmed must be a multiple of 416.
5. Bulk Erase is accepted when security is set and only when the BEDIS status field is not set.
6. The flash will be in NVM Special mode, restricting the type of commands that can be executed through WRITE_FCCOB when security is enabled.

## 30.3.1 Command descriptions

This section describes the module commands.

### 30.3.1.1 Write Enable

The Write Enable (WREN) command sets the write enable register field in the EzPort status register. The write enable field must be set for a write command (SP, SE, BE, WRFCCOB, or WRFLEXRAM) to be accepted. The write enable register field clears on reset, on a Write Disable command, and at the completion of write command. This command must not be used if a write is already in progress.

### 30.3.1.2 Write Disable

The Write Disable (WRDI) command clears the write enable register field in the status register. This command must not be used if a write is already in progress.

### 30.3.1.3 Read Status Register

The Read Status Register (RDSR) command returns the contents of the EzPort status register.

**Table 30-3. EzPort status register**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | FS | WEF | | | FLEXRAM | BEDIS | WEN | WIP |
| W | | | | | | | | |
| Reset: | 0/1[1] | 0 | 0 | 0 | 0/1[2] | 0/1[3] | 0 | 1[4] |

1. Reset value reflects the status of flash security out of reset.
2. Reset value reflects FlexNVM flash partitioning. If FlexNVM flash has been paritioned for EEPROM, this field is set immediately after reset. Note that FLEXRAM is cleared after the EzPort initialization sequence completes, as indicated by clearing of WIP.
3. Reset value reflects whether bulk erase is enabled or disabled out of reset.
4. Initial value of WIP is 1, but the value clears to 0 after EzPort initialization is complete.

**Table 30-4. EzPort status register field description**

| Field | Description |
|---|---|
| 0<br>WIP | Write in progress.<br>Sets after a write command (SP, SE, BE, WRFCCOB, or WRFLEXRAM) is accepted and clears after the flash memory has completed all operations associated with the write command, as indicated by the Command Complete Interrupt Flag (CCIF) inside the flash. This field is also asserted on reset and cleared when EzPort initialization is complete. Only the Read Status Register (RDSR) command is accepted while a write is in progress.<br>0 = Write is not in progress. Accept any command.<br>1 = Write is in progress. Only accept RDSR command. |
| 1<br>WEN | Write enable<br>Enables the write comman that follows. It is a control field that must be set before a write command (SP, SE, BE, WRFCCOB, or WRFLEXRAM) is accepted. Is set by the Write Enable (WREN) command and cleared by reset or a Write Disable (WRDI) command. This field also clears when the flash memory has completed all operations associated with the command.<br>0 = Disables the following write command.<br>1 = Enables the following write command. |
| 2<br>BEDIS | Bulk erase disable<br>Indicates whether bulk erase (BE) is disabled when flash is secure.<br>0 = BE is enabled.<br>1 = BE is disabled if FS is also set. Attempts to issue a BE command will result in the WEF flag being set. |
| 3<br>FLEXRAM | For devices with FlexRAM: FlexRAM mode<br>Indicates the current mode of the FlexRAM. Valid only when WIP is cleared.<br>0 = FlexRAM is in RAM mode. RD/WRFLEXRAM command can be used to read/write data in FlexRAM.<br>1 = FlexRAM is in EEPROM mode. SP command is not accepted. RD/WRFLEXRAM command can be used to read/write data in the FlexRAM. |

*Table continues on the next page...*

**Table 30-4. EzPort status register field description (continued)**

| Field | Description |
|---|---|
| 6<br>WEF | Write error flag<br>Indicates whether there has been an error while executing a write command (SP, SE, BE, WRFCCOB, or WRFLEXRAM). The WEF flag will set if Flash Access Error Flag (ACCERR), Flash Protection Violation (FPVIOL), or Memory Controller Command Completion Status (MGSTAT0) inside the flash memory is set at the completion of the write command. See the flash memory chapter for further description of these flags and their sources. The WEF flag clears after a Read Status Register (RDSR) command.<br>0 = No error on previous write command.<br>1 = Error on previous write command. |
| 7<br>FS | Flash security<br>Indicates whether the flash is secure. See Table 30-2 for the list of commands that will be accepted when flash is secure. Flash security can be disabled by performing a BE command.<br>0 = Flash is not secure.<br>1 = Flash is secure. |

### 30.3.1.4 Read Data

The Read Data (READ) command returns data from the flash memory or FlexNVM, depending on the initial address specified in the command word. The initial address must be 32-bit aligned with the two LSBs being zero.

Data continues being returned for as long as the EzPort chip select ($\overline{\text{EZP\_CS}}$) is asserted, with the address automatically incrementing. In this way, the entire contents of flash can be returned by one command. Attempts to read from an address which does not fall within the valid address range for the flash memory regions returns unknown data. See Flash memory map for EzPort access.

For this command to return the correct data, the EzPort clock (EZP_CK) must run at the internal system clock divided by eight or slower. This command is not accepted if the WEF, WIP, or FS field in the EzPort status register is set.

### 30.3.1.5 Read Data at High Speed

The Read Data at High Speed (FAST_READ) command is identical to the READ command, except for the inclusion of a dummy byte following the address bytes and before the first data byte is returned.

This command can be run with an EzPort clock (EZP_CK) frequency of half the internal system clock frequency of the microcontroller or slower. This command is not accepted if the WEF, WIP, or FS field in the EzPort status register is set.

### 30.3.1.6 Section Program

The Section Program (SP) command programs up to one section of flash memory that has previously been erased. Please see the Flash Memory chapter for a definition of section size. The starting address of the memory to program is sent after the command word and must be a 32-bit aligned address with the two LSBs being zero.128-bit aligned address with the four LSBs being zero.

As data is shifted in, the EzPort buffers the data in FlexRAMSystem RAM before executing an SP command within the flashsequentially moving the data into flash using the Program Longword command. For this reason, the number of bytes to be programmed must be a multiple of 416 and up to one flash section can be programmed at a time. For more details, see the Flash Block Guide.

Attempts to program more than one section, across a sector boundary or from an initial address which does not fall within the valid address range for the flash causes the WEF flag to set. See Flash memory map for EzPort access.

For devices with FlexRAM: This command requires the FlexRAM to be configured for traditional RAM operation. By default, after entering EzPort mode, the FlexRAM is configured for traditional RAM operation. If the user reconfigures FlexRAM for EEPROM operation, then the user should use the WRFCCOB command to configure FlexRAM back to traditional RAM operation before issuing an SP command. See the Flash Memory chapter for details on how the FlexRAM function is modified.

This command is not accepted if the WEF, WIP, FLEXRAM, or FS field is set or if the WEN field is not set in the EzPort status register.

### 30.3.1.7 Sector Erase

The Sector Erase (SE) command erases the contents of one sector of flash memory. The three byte address sent after the command byte can be any address within the sector to erase, but must be a 128-bit aligned address (the four LSBs must be zero). Attempts to erase from an initial address which does not fall within the valid address range (see Flash memory map for EzPort access) for the flash results in the WEF flag being set.

This command is not accepted if the WEF, WIP or FS field is set or if the WEN field is not set in the EzPort status register.

### 30.3.1.8 Bulk Erase

The Bulk Erase (BE) command erases the entire contents of flash memory, ignoring any protected sectors or flash security. Flash security is disabled upon successful completion of the BE command.

Attempts to issue a BE command while the BEDIS and FS fields are set results in the WEF flag being set in the EzPort status register. Also, this command is not accepted if the WEF or WIP field is set or if the WEN field is not set in the EzPort status register.

### 30.3.1.9 EzPort Reset Chip

The Reset Chip (RESET) command forces the chip into the reset state. If the EzPort chip select ($\overline{\text{EZP\_CS}}$) pin is asserted at the end of the reset period, EzPort is enabled; otherwise, it is disabled. This command allows the chip to boot up from flash memory after being programmed by an external source.

This command is not accepted if the WIP field is set in the EzPort status register.

### 30.3.1.10 Write FCCOB Registers

The Write FCCOB Registers (WRFCCOB) command allows the user to write to the flash common command object registers and execute any command allowed by the flash.

**NOTE**

When security is enabled, the flash is configured in NVM Special mode, restricting the commands that can be executed by the flash.

After receiving 12 bytes of data, EzPort writes the data to the FCCOB 0-B registers in the flash and then automatically launches the command within the flash. If greater or less than 12 bytes of data is received, this command has unexpected results and may result in the WEF flag being set.

This command is not accepted if the WEF or WIP field is set or if the WEN field is not set in the EzPort status register.

### 30.3.1.11 Read FCCOB Registers at High Speed

The Read FCCOB Registers at High Speed (FAST_RDFCCOB) command allows the user to read the contents of the flash common command object registers. After receiving the command, EzPort waits for one dummy byte of data before returning FCCOB register data starting at FCCOB 0 and ending with FCCOB B.

This command can be run with an EzPort clock (EZP_CK) frequency half the internal system clock frequency of the microcontroller or slower. Attempts to read greater than 12 bytes of data returns unknown data. This command is not accepted if the WEF, WIP, or FS fields in the EzPort status register are 1.

### 30.3.1.12 Write FlexRAM

This command is only applicable for devices with FlexRAM.

The Write FlexRAM (WRFLEXRAM) command allows the user to write four bytes of data to the FlexRAM. If the FlexRAM is configured for EEPROM operation, the WRFLEXRAM command can effectively be used to create data records in the EEPROM flash memory.

By default, after entering EzPort mode, the FlexRAM is configured for traditional RAM operation and functions as direct RAM. The user can alter the FlexRAM configuration by using WRFCCOB to execute a Set FlexRAM or Program Partition command within the flash.

The address of the FlexRAM location to be written is sent after the command word and must be a 32-bit aligned address (the two LSBs must be zero). Attempts to write an address which does not fall within the valid address range for the FlexRAM results in the value of the WEF flag being 1. See Flash memory map for EzPort access for more information.

After receiving four bytes of data, EzPort writes the data to the FlexRAM. If greater or less than four bytes of data is received, this command has unexpected results and may result in the value of the WEF flag being 1.

This command is not accepted if the WEF, WIP or FS fields are 1 or if the WEN field is 0 in the EzPort status register.

### 30.3.1.13 Read FlexRAM

This command is only applicable for devices with FlexRAM.

The Read FlexRAM (RDFLEXRAM) command returns data from the FlexRAM. If the FlexRAM is configured for EEPROM operation, the RDFLEXRAM command can effectively be used to read data from EEPROM flash memory.

Data continues being returned for as long as the EzPort chip select ($\overline{\text{EZP\_CS}}$) is asserted, with the address automatically incrementing. In this way, the entire contents of FlexRAM can be returned by one command.

The initial address must be 32-bit aligned (the two LSBs must be zero). Attempts to read from an address which does not fall within the valid address range for the FlexRAM returns unknown data. See Flash memory map for EzPort access for more information.

For this command to return the correct data, the EzPort clock (EZP_CK) must run at the internal system clock divided by eight or slower. This command is not accepted if the WEF, WIP, or FS fields in the EzPort status register are set.

#### 30.3.1.14 Read FlexRAM at High Speed

This command is only applicable for devices with FlexRAM.

The Read FlexRAM at High Speed (FAST_RDFLEXRAM) command is identical to the RDFLEXRAM command, except for the inclusion of a dummy byte following the address bytes and before the first data byte is returned.

This command can be run with an EzPort clock (EZP_CK) frequency up to and including half the internal system clock frequency of the microcontroller. This command is not accepted if the WEF, WIP, or FS fields in the EzPort status register are set.

## 30.4 Flash memory map for EzPort access

The following table shows the flash memory map for access through EzPort.

**NOTE**

The flash block address map for access through EzPort may not conform to the system memory map. Changes are made to allow the EzPort address width to remain 24 bits.

**Table 30-5. Flash Memory Map for EzPort Access**

| Valid start address | Size | Flash block | Valid commands |
|---|---|---|---|
| 0x0000_0000 | See device's chip configuration details | Flash | READ, FAST_READ, SP, SE, BE |
| 0x0000_0000 | See device's chip configuration details | FlexRAM | RDFLEXRAM, FAST_RDFLEXRAM, WRFLEXRAM, BE |

# Chapter 31
# External Bus Interface (FlexBus)

## 31.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

This chapter describes external bus data transfer operations and error conditions. It describes transfers initiated by the core processor (or any other bus master) and includes detailed timing diagrams showing the interaction of signals in supported bus operations.

### 31.1.1 Definition

The FlexBus multifunction external bus interface controller is a hardware module that:

- Provides memory expansion and provides connection to external peripherals with a parallel bus
- Can be directly connected to the following asynchronous or synchronous slave-only devices with little or no additional circuitry:
  - External ROMs
  - Flash memories
  - Programmable logic devices
  - Other simple target (slave) devices

### 31.1.2 Features

FlexBus offers the following features:

- Six independent, user-programmable chip-select signals ($\overline{\text{FB_CS}}$5 –$\overline{\text{FB_CS}}$0)
- 8-bit, 16-bit, and 32-bit port sizes with configuration for multiplexed or nonmultiplexed address and data buses
- 8-bit, 16-bit, 32-bit, and 16-byte transfers
- Programmable burst and burst-inhibited transfers selectable for each chip-select and transfer direction
- Programmable address-setup time with respect to the assertion of a chip-select
- Programmable address-hold time with respect to the deassertion of a chip-select and transfer direction
- Extended address latch enable option to assist with glueless connections to synchronous and asynchronous memory devices

## 31.2 Signal descriptions

This table describes the external signals involved in data-transfer operations.

**NOTE**

Not all of the following signals may be available on a particular device. See the Chip Configuration details for information on which signals are available.

**Table 31-1. FlexBus signal descriptions**

| Signal | I/O | Function |
|---|---|---|
| FB_A31–FB_A0 | O | Address Bus<br>When FlexBus is used in a nonmultiplexed configuration, this is the address bus. When FlexBus is used in a multiplexed configuration, this bus is not used. |
| FB_D31–FB_D0 | I/O | Data Bus—During the first cycle, this bus drives the upper address byte, addr[31:24].<br>When FlexBus is used in a nonmultiplexed configuration, this is the data bus, FB_D. When FlexBus is used in a multiplexed configuration, this is the address and data bus, FB_AD.<br>The number of byte lanes carrying the data is determined by the port size associated with the matching chip-select.<br>When FlexBus is used in a multiplexed configuration, the full 32-bit address is driven on the first clock of a bus cycle (address phase). After the first clock, the data is driven on the bus (data phase). During the data phase, the address is driven on the pins not used for data. For example, in 16-bit mode, the lower address is driven on FB_AD15–FB_AD0, and in 8-bit mode, the lower address is driven on FB_AD23–FB_AD0. |

*Table continues on the next page...*

## Table 31-1. FlexBus signal descriptions (continued)

| Signal | I/O | Function |
|---|---|---|
| FB_CS5–FB_CS0 | O | General Purpose Chip-Selects—Indicate which external memory or peripheral is selected. A particular chip-select is asserted when the transfer address is within the external memory's or peripheral's address space, as defined in CSAR[BA] and CSMR[BAM]. |
| FB_BE_31_24<br>FB_BE_23_16<br>FB_BE_15_8<br>FB_BE_7_0 | O | Byte Enables—Indicate that data is to be latched or driven onto a specific byte lane of the data bus. CSCR[BEM] determines if these signals are asserted on reads and writes or on writes only.<br>For external SRAM or flash devices, the FB_BE outputs should be connected to individual byte strobe signals. |
| FB_OE | O | Output Enable—Sent to the external memory or peripheral to enable a read transfer. This signal is asserted during read accesses only when a chip-select matches the current address decode. |
| FB_R/W | O | Read/Write—Indicates whether the current bus operation is a read operation (FB_R/W high) or a write operation (FB_R/W low). |
| FB_TS | O | Transfer Start—Indicates that the chip has begun a bus transaction and that the address and attributes are valid.<br>An inverted FB_TS is available as an address latch enable (FB_ALE), which indicates when the address is being driven on the FB_AD bus.<br>FB_TS/FB_ALE is asserted for one bus clock cycle.<br>The chip can extend this signal until the first positive clock edge after FB_CS asserts. See CSCR[EXTS] and Extended Transfer Start/Address Latch Enable. |
| FB_ALE | O | Address Latch Enable—Indicates when the address is being driven on the FB_A bus (inverse of FB_TS). |

*Table continues on the next page...*

## Table 31-1. FlexBus signal descriptions (continued)

| Signal | I/O | Function |
|---|---|---|
| FB_TSIZ1–FB_TSIZ0 | O | Transfer Size—Indicates (along with FB_TBST) the data transfer size of the current bus operation. The interface supports 8-, 16-, and 32-bit operand transfers and allows accesses to 8-, 16-, and 32-bit data ports.<br>• 00b = 4 bytes<br>• 01b = 1 byte<br>• 10b = 2 bytes<br>• 11b = 16 bytes (line)<br><br>For misaligned transfers, FB_TSIZ1–FB_TSIZ0 indicate the size of each transfer. For example, if a 32-bit access through a 32-bit port device occurs at a misaligned offset of 1h, 8 bits are transferred first (FB_TSIZ1–FB_TSIZ0 = 01b), 16 bits are transferred next at offset 2h (FB_TSIZ1–FB_TSIZ0 = 10b), and the final 8 bits are transferred at offset 4h (FB_TSIZ1–FB_TSIZ0 = 01b).<br>For aligned transfers larger than the port size, FB_TSIZ1–FB_TSIZ0 behave as follows:<br>• If bursting is used, FB_TSIZ1–FB_TSIZ0 are driven to the transfer size.<br>• If bursting is inhibited, FB_TSIZ1–FB_TSIZ0 first show the entire transfer size and then show the port size.<br><br>For burst-inhibited transfers, FB_TSIZ1–FB_TSIZ0 change with each FB_TS assertion to reflect the next transfer size.<br>For transfers to port sizes smaller than the transfer size, FB_TSIZ1–FB_TSIZ0 indicate the size of the entire transfer on the first access and the size of the current port transfer on subsequent transfers. For example, for a 32-bit write to an 8-bit port, FB_TSIZ1–FB_TSIZ0 are 00b for the first transaction and 01b for the next three transactions. If bursting is used for a 32-bit write to an 8-bit port, FB_TSIZ1–FB_TSIZ0 are driven to 00b for the entire transfer. |
| FB_TBST | O | Transfer Burst—Indicates that a burst transfer is in progress as driven by the chip. A burst transfer can be 2 to 16 beats depending on FB_TSIZ1–FB_TSIZ0 and the port size.<br>**Note:** When a burst transfer is in progress (FB_TBST = 0b), the transfer size is 16 bytes (FB_TSIZ1–FB_TSIZ0 = 11b), and the address is misaligned within the 16-byte boundary, the external memory or peripheral must be able to wrap around the address. |

*Table continues on the next page...*

**Table 31-1. FlexBus signal descriptions (continued)**

| Signal | I/O | Function |
|---|---|---|
| FB_TA | I | Transfer Acknowledge—Indicates that the external data transfer is complete. When FB_TA is asserted during a read transfer, FlexBus latches the data and then terminates the transfer. When FB_TA is asserted during a write transfer, the transfer is terminated.<br><br>If auto-acknowledge is disabled (CSCR[AA] = 0), the external memory or peripheral drives FB_TA to terminate the transfer. If auto-acknowledge is enabled (CSCR[AA] = 1), FB_TA is generated internally after a specified number of wait states, or the external memory or peripheral may assert external FB_TA before the wait-state countdown to terminate the transfer early. The chip deasserts FB_CS one cycle after the last FB_TA is asserted. During read transfers, the external memory or peripheral must continue to drive data until FB_TA is recognized. For write transfers, the chip continues driving data one clock cycle after FB_CS is deasserted.<br><br>The number of wait states is determined by CSCR or the external FB_TA input. If the external FB_TA is used, the external memory or peripheral has complete control of the number of wait states.<br><br>**Note:** External memory or peripherals should assert FB_TA only while the FB_CS signal to the external memory or peripheral is asserted.<br><br>The CSPMCR register controls muxing of FB_TA with other signals. If auto-acknowledge is not used and CSPMCR does not allow FB_TA control, FlexBus may hang. |
| FB_CLK | O | FlexBus Clock Output |

## 31.3 Memory Map/Register Definition

The following tables describe the registers and bit meanings for configuring chip-select operation.

The actual number of chip selects available depends upon the device and its pin configuration. If the device does not support certain chip select signals or the pin is not configured for a chip-select function, then that corresponding set of chip-select registers has no effect on an external pin.

### Note

You must set CSMR0[V] before the chip select registers take effect.

A bus error occurs when writing to reserved register locations.

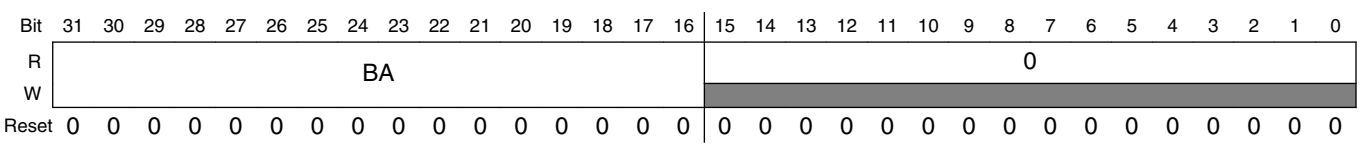
**FB memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4000_C000 | Chip Select Address Register (FB_CSAR0) | 32 | R/W | 0000_0000h | 31.3.1/710 |
| 4000_C004 | Chip Select Mask Register (FB_CSMR0) | 32 | R/W | 0000_0000h | 31.3.2/711 |
| 4000_C008 | Chip Select Control Register (FB_CSCR0) | 32 | R/W | 0000_0000h | 31.3.3/712 |
| 4000_C00C | Chip Select Address Register (FB_CSAR1) | 32 | R/W | 0000_0000h | 31.3.1/710 |
| 4000_C010 | Chip Select Mask Register (FB_CSMR1) | 32 | R/W | 0000_0000h | 31.3.2/711 |
| 4000_C014 | Chip Select Control Register (FB_CSCR1) | 32 | R/W | 0000_0000h | 31.3.3/712 |
| 4000_C018 | Chip Select Address Register (FB_CSAR2) | 32 | R/W | 0000_0000h | 31.3.1/710 |
| 4000_C01C | Chip Select Mask Register (FB_CSMR2) | 32 | R/W | 0000_0000h | 31.3.2/711 |
| 4000_C020 | Chip Select Control Register (FB_CSCR2) | 32 | R/W | 0000_0000h | 31.3.3/712 |
| 4000_C024 | Chip Select Address Register (FB_CSAR3) | 32 | R/W | 0000_0000h | 31.3.1/710 |
| 4000_C028 | Chip Select Mask Register (FB_CSMR3) | 32 | R/W | 0000_0000h | 31.3.2/711 |
| 4000_C02C | Chip Select Control Register (FB_CSCR3) | 32 | R/W | 0000_0000h | 31.3.3/712 |
| 4000_C030 | Chip Select Address Register (FB_CSAR4) | 32 | R/W | 0000_0000h | 31.3.1/710 |
| 4000_C034 | Chip Select Mask Register (FB_CSMR4) | 32 | R/W | 0000_0000h | 31.3.2/711 |
| 4000_C038 | Chip Select Control Register (FB_CSCR4) | 32 | R/W | 0000_0000h | 31.3.3/712 |
| 4000_C03C | Chip Select Address Register (FB_CSAR5) | 32 | R/W | 0000_0000h | 31.3.1/710 |
| 4000_C040 | Chip Select Mask Register (FB_CSMR5) | 32 | R/W | 0000_0000h | 31.3.2/711 |
| 4000_C044 | Chip Select Control Register (FB_CSCR5) | 32 | R/W | 0000_0000h | 31.3.3/712 |
| 4000_C060 | Chip Select port Multiplexing Control Register (FB_CSPMCR) | 32 | R/W | 0000_0000h | 31.3.4/715 |

## 31.3.1 Chip Select Address Register (FB_CSAR*n*)

Specifies the associated chip-select's base address.

Address: 4000_C000h base + 0h offset + (12d × i), where i=0d to 5d




**FB_CSAR*n* field descriptions**

| Field | Description |
|---|---|
| 31–16<br>BA | Base Address<br><br>Defines the base address for memory dedicated to the associated chip-select. BA is compared to bits 31–16 on the internal address bus to determine if the associated chip-select's memory is being accessed. |

*Table continues on the next page...*

**FB_CSAR*n* field descriptions (continued)**

| Field | Description |
|---|---|
| | **NOTE:** Because the FlexBus module is one of the slaves connected to the crossbar switch, it is only accessible within a certain memory range. See the chip memory map for the applicable FlexBus "expansion" address range for which the chip-selects can be active. Set the CSARn and CSMRn registers appropriately before accessing this region. |
| 15–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 31.3.2 Chip Select Mask Register (FB_CSMR*n*)

Specifies the address mask and allowable access types for the associated chip-select.

Address: 4000_C000h base + 4h offset + (12d × i), where i=0d to 5d

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

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

**FB_CSMR*n* field descriptions**

| Field | Description |
|---|---|
| 31–16<br>BAM | Base Address Mask<br>Defines the associated chip-select's block size by masking address bits.<br>0 The corresponding address bit in CSAR is used in the chip-select decode.<br>1 The corresponding address bit in CSAR is a don't care in the chip-select decode. |
| 15–9<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 8<br>WP | Write Protect<br>Controls write accesses to the address range in the corresponding CSAR.<br>0 Write accesses are allowed.<br>1 Write accesses are not allowed. Attempting to write to the range of addresses for which the WP bit is set results in a bus error termination of the internal cycle and no external cycle. |
| 7–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>V | Valid<br>Specifies whether the corresponding CSAR, CSMR, and CSCR contents are valid. Programmed chip-selects do not assert until the V bit is 1b (except for FB_CS0, which acts as the global chip-select). |

*Table continues on the next page...*

**FB_CSMR*n* field descriptions (continued)**

| Field | Description |
|---|---|
| | **NOTE:** At reset, FB_CS0 will fire for any access to the FlexBus memory region. CSMR0[V] must be set as part of the chip select initialization sequence to allow other chip selects to function as programmed.<br><br>0 Chip-select is invalid.<br>1 Chip-select is valid. |

## 31.3.3 Chip Select Control Register (FB_CSCR*n*)

Controls the auto-acknowledge, address setup and hold times, port size, burst capability, and number of wait states for the associated chip select.

**NOTE**

To support the global chip-select (FB_CS0), the CSCR0 reset values differ from the other CSCRs. The reset value of CSCR0 is as follows:

- Bits 31–24 are 0b
- Bit 23–3 are chip-dependent
- Bits 3–0 are 0b

See the chip configuration details for your particular chip for information on the exact CSCR0 reset value.

Address: 4000_C000h base + 8h offset + (12d × i), where i=0d to 5d

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SWS | | | | | | 0 | | SWSEN | EXTS | ASET | | RDAH | | WRAH | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WS | | | | | | BLS | AA | PS | | BEM | BSTR | BSTW | 0 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FB_CSCR*n* field descriptions**

| Field | Description |
|---|---|
| 31–26<br>SWS | Secondary Wait States |

*Table continues on the next page...*

## FB_CSCR*n* field descriptions (continued)

| Field | Description |
|---|---|
| | Used only when the SWSEN bit is 1b. Specifies the number of wait states inserted before an internal transfer acknowledge is generated for a burst transfer (except for the first termination, which is controlled by WS). |
| 25–24<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 23<br>SWSEN | Secondary Wait State Enable<br><br>0 Disabled. A number of wait states (specified by WS) are inserted before an internal transfer acknowledge is generated for all transfers.<br>1 Enabled. A number of wait states (specified by SWS) are inserted before an internal transfer acknowledge is generated for burst transfer secondary terminations. |
| 22<br>EXTS | Extended Transfer Start/Extended Address Latch Enable<br><br>Controls how long $\overline{\text{FB\_TS}}$ /FB_ALE is asserted.<br><br>0 Disabled. $\overline{\text{FB\_TS}}$ /FB_ALE asserts for one bus clock cycle.<br>1 Enabled. $\overline{\text{FB\_TS}}$ /FB_ALE remains asserted until the first positive clock edge after $\overline{\text{FB\_CSn}}$ asserts. |
| 21–20<br>ASET | Address Setup<br><br>Controls when the chip-select is asserted with respect to assertion of a valid address and attributes.<br><br>00 Assert $\overline{\text{FB\_CSn}}$ on the first rising clock edge after the address is asserted (default for all but $\overline{\text{FB\_CS0}}$ ).<br>01 Assert $\overline{\text{FB\_CSn}}$ on the second rising clock edge after the address is asserted.<br>10 Assert $\overline{\text{FB\_CSn}}$ on the third rising clock edge after the address is asserted.<br>11 Assert $\overline{\text{FB\_CSn}}$ on the fourth rising clock edge after the address is asserted (default for $\overline{\text{FB\_CS0}}$ ). |
| 19–18<br>RDAH | Read Address Hold or Deselect<br><br>Controls the address and attribute hold time after the termination during a read cycle that hits in the associated chip-select's address space.<br><br>**NOTE:** • The hold time applies only at the end of a transfer. Therefore, during a burst transfer or a transfer to a port size smaller than the transfer size, the hold time is only added after the last bus cycle.<br>• The number of cycles the address and attributes are held after $\overline{\text{FB\_CSn}}$ deassertion depends on the value of the AA bit.<br><br>00 When AA is 0b, 1 cycle. When AA is 1b, 0 cycles.<br>01 When AA is 0b, 2 cycles. When AA is 1b, 1 cycle.<br>10 When AA is 0b, 3 cycles. When AA is 1b, 2 cycles.<br>11 When AA is 0b, 4 cycles. When AA is 1b, 3 cycles. |
| 17–16<br>WRAH | Write Address Hold or Deselect<br><br>Controls the address, data, and attribute hold time after the termination of a write cycle that hits in the associated chip-select's address space.<br><br>**NOTE:** The hold time applies only at the end of a transfer. Therefore, during a burst transfer or a transfer to a port size smaller than the transfer size, the hold time is only added after the last bus cycle.<br><br>00 1 cycle (default for all but $\overline{\text{FB\_CS0}}$ )<br>01 2 cycles<br>10 3 cycles<br>11 4 cycles (default for $\overline{\text{FB\_CS0}}$ ) |

*Table continues on the next page...*

## FB_CSCR*n* field descriptions (continued)

| Field | Description |
|---|---|
| 15–10<br>WS | Wait States<br><br>Specifies the number of wait states inserted after FlexBus asserts the associated chip-select and before an internal transfer acknowledge is generated (WS = 00h inserts 0 wait states, ..., WS = 3Fh inserts 63 wait states). |
| 9<br>BLS | Byte-Lane Shift<br><br>Specifies if data on FB_AD appears left-aligned or right-aligned during the data phase of a FlexBus access.<br><br>0 Not shifted. Data is left-aligned on FB_AD.<br>1 Shifted. Data is right-aligned on FB_AD. |
| 8<br>AA | Auto-Acknowledge Enable<br><br>Asserts the internal transfer acknowledge for accesses specified by the chip-select address.<br><br>**NOTE:** If AA is 1b for a corresponding FB_CSn and the external system asserts an external $\overline{\text{FB\_TA}}$ before the wait-state countdown asserts the internal FB_TA, the cycle is terminated. Burst cycles increment the address bus between each internal termination.<br><br>**NOTE:** This field must be 1b if CSPMCR disables FB_TA.<br><br>0 Disabled. No internal transfer acknowledge is asserted and the cycle is terminated externally.<br>1 Enabled. Internal transfer acknowledge is asserted as specified by WS. |
| 7–6<br>PS | Port Size<br><br>Specifies the data port width of the associated chip-select, and determines where data is driven during write cycles and where data is sampled during read cycles.<br><br>00 32-bit port size. Valid data is sampled and driven on FB_D[31:0].<br>01 8-bit port size. Valid data is sampled and driven on FB_D[31:24] when BLS is 0b, or FB_D[7:0] when BLS is 1b.<br>1X 16-bit port size. Valid data is sampled and driven on FB_D[31:16] when BLS is 0b, or FB_D[15:0] when BLS is 1b. |
| 5<br>BEM | Byte-Enable Mode<br><br>Specifies whether the corresponding $\overline{\text{FB\_BE}}$ is asserted for read accesses. Certain memories have byte enables that must be asserted during reads and writes. Write 1b to the BEM bit in the relevant CSCR to provide the appropriate mode of byte enable support for these SRAMs.<br><br>0 $\overline{\text{FB\_BE}}$ is asserted for data write only.<br>1 $\overline{\text{FB\_BE}}$ is asserted for data read and write accesses. |
| 4<br>BSTR | Burst-Read Enable<br><br>Specifies whether burst reads are enabled for memory associated with each chip select.<br><br>0 Disabled. Data exceeding the specified port size is broken into individual, port-sized, non-burst reads. For example, a 32-bit read from an 8-bit port is broken into four 8-bit reads.<br>1 Enabled. Enables data burst reads larger than the specified port size, including 32-bit reads from 8- and 16-bit ports, 16-bit reads from 8-bit ports, and line reads from 8-, 16-, and 32-bit ports. |
| 3<br>BSTW | Burst-Write Enable<br><br>Specifies whether burst writes are enabled for memory associated with each chip select. |

*Table continues on the next page...*

**FB_CSCR*n* field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Disabled. Data exceeding the specified port size is broken into individual, port-sized, non-burst writes. For example, a 32-bit write to an 8-bit port takes four byte writes.<br>1 Enabled. Enables burst write of data larger than the specified port size, including 32-bit writes to 8- and 16-bit ports, 16-bit writes to 8-bit ports, and line writes to 8-, 16-, and 32-bit ports. |
| 2–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 31.3.4 Chip Select port Multiplexing Control Register (FB_CSPMCR)

Controls the multiplexing of the FlexBus signals.

**NOTE**

A bus error occurs when you do any of the following:

- Write to a reserved address
- Write to a reserved field in this register, or
- Access this register using a size other than 32 bits.

Address: 4000_C000h base + 60h offset = 4000_C060h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | GROUP1 | | | | GROUP2 | | | | GROUP3 | | | | GROUP4 | | | | GROUP5 | | | | 0 | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FB_CSPMCR field descriptions**

| Field | Description |
|---|---|
| 31–28<br>GROUP1 | FlexBus Signal Group 1 Multiplex control<br><br>Controls the multiplexing of the FB_ALE, $\overline{\text{FB\_CS1}}$ , and $\overline{\text{FB\_TS}}$ signals.<br><br>0000 FB_ALE<br>0001 $\overline{\text{FB\_CS1}}$<br>0010 $\overline{\text{FB\_TS}}$<br>Any other value Reserved |
| 27–24<br>GROUP2 | FlexBus Signal Group 2 Multiplex control<br><br>Controls the multiplexing of the $\overline{\text{FB\_CS4}}$ , FB_TSIZ0, and $\overline{\text{FB\_BE\_31\_24}}$ signals.<br><br>0000 $\overline{\text{FB\_CS4}}$<br>0001 FB_TSIZ0<br>0010 $\overline{\text{FB\_BE\_31\_24}}$<br>Any other value Reserved |
| 23–20<br>GROUP3 | FlexBus Signal Group 3 Multiplex control<br><br>Controls the multiplexing of the $\overline{\text{FB\_CS5}}$ , FB_TSIZ1, and $\overline{\text{FB\_BE\_23\_16}}$ signals. |

*Table continues on the next page...*

**FB_CSPMCR field descriptions (continued)**

| Field | Description |
|---|---|
| | 0000 FB_CS5<br>0001 FB_TSIZ1<br>0010 FB_BE_23_16<br>Any other value Reserved |
| 19–16<br>GROUP4 | FlexBus Signal Group 4 Multiplex control<br><br>Controls the multiplexing of the FB_TBST , FB_CS2 , and FB_BE_15_8 signals.<br><br>0000 FB_TBST<br>0001 FB_CS2<br>0010 FB_BE_15_8<br>Any other value Reserved |
| 15–12<br>GROUP5 | FlexBus Signal Group 5 Multiplex control<br><br>Controls the multiplexing of the FB_TA , FB_CS3 , and FB_BE_7_0 signals.<br><br>**NOTE:** When GROUP5 is not 0000b, you must write 1b to the CSCR[AA] bit. Otherwise, the bus hangs during a transfer.<br><br>0000 FB_TA<br>0001 FB_CS3 . You must also write 1b to CSCR[AA].<br>0010 FB_BE_7_0 . You must also write 1b to CSCR[AA].<br>Any other value Reserved |
| 11–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 31.4 Functional description

### 31.4.1 Modes of operation

FlexBus supports the following modes of operation:

- Multiplexed 32-bit address and 32-bit data
- Multiplexed 32-bit address and 16-bit data (non-multiplexed 16-bit address and 16-bit data)
- Multiplexed 32-bit address and 8-bit data (non-multiplexed 24-bit address and 8-bit data)
- Non-multiplexed 32-bit address and 32-bit data busses

### 31.4.2 Address comparison

When a bus cycle is routed to FlexBus, FlexBus compares the transfer address to the base address (see CSAR[BA]) and base address mask (see CSMR[BAM]). This table describes how FlexBus decides to assert a chip-select and complete the bus cycle based on the address comparison.

| When the transfer address | Then FlexBus |
|---|---|
| Matches one address register configuration | Asserts the appropriate chip-select, generating a FlexBus bus cycle as defined in the appropriate CSCR.<br>If CSMR[WP] is set and a write access is performed, FlexBus terminates the internal bus cycle with a bus error, does not assert a chip-select, and does not perform an external bus cycle. |
| Does not match an address register configuration | Terminates the transfer with a bus error response, does not assert a chip-select, and does not perform a FlexBus cycle. |
| Matches more than one address register configuration | Terminates the transfer with a bus error response, does not assert a chip-select, and does not perform a FlexBus cycle. |

### 31.4.3 Address driven on address bus

FlexBus always drives a 32-bit address on the FB_AD bus regardless of the external memory's or peripheral's address size.

### 31.4.4 Connecting address/data lines

The external device must connect its address and data lines as follows:

- Address lines
  - FB_AD from FB_AD0 upward
- Data lines
  - If CSCR[BLS] = 0, FB_AD from FB_AD31 downward
  - If CSCR[BLS] = 1, FB_AD from FB_AD0 upward

### 31.4.5 Bit ordering

No bit ordering is required when connecting address and data lines to the FB_AD bus. For example, a full 16-bit address/16-bit data device connects its addr15–addr0 to FB_AD16–FB_AD1 and data15–data0 to FB_AD31–FB_AD16. See Data-byte alignment and physical connections for a graphical connection.

### 31.4.6 Data transfer signals

Data transfers between FlexBus and the external memory or peripheral involve these signals:

- Address/data bus (FB_AD31–FB_AD0 )
- Control signals (FB_TS/FB_ALE, FB_TA, FB_CS*n*, FB_OE, FB_R/W, FB_BE*n*)
- Attribute signals (FB_TBST, FB_TSIZ1–FB_TSIZ0)

### 31.4.7 Signal transitions

These signals change on the rising edge of the FlexBus clock (FB_CLK):

- Address
- Write data
- FB_TS/FB_ALE
- FB_CSn
- All attribute signals

FlexBus latches the read data on the rising edge of the clock.

### 31.4.8 Data-byte alignment and physical connections

The device aligns data transfers in FlexBus byte lanes with the number of lanes depending on the data port width.

The following figure shows the byte lanes that external memory or peripheral connects to and the sequential transfers of a 32-bit transfer for the supported port sizes when byte lane shift is disabled. For example, an 8-bit memory connects to the single lane FB_AD31–FB_AD24 (FB_BE_31_24). A 32-bit transfer through this 8-bit port takes four transfers, starting with the LSB to the MSB. A 32-bit transfer through a 32-bit port requires one transfer on each four-byte lane.

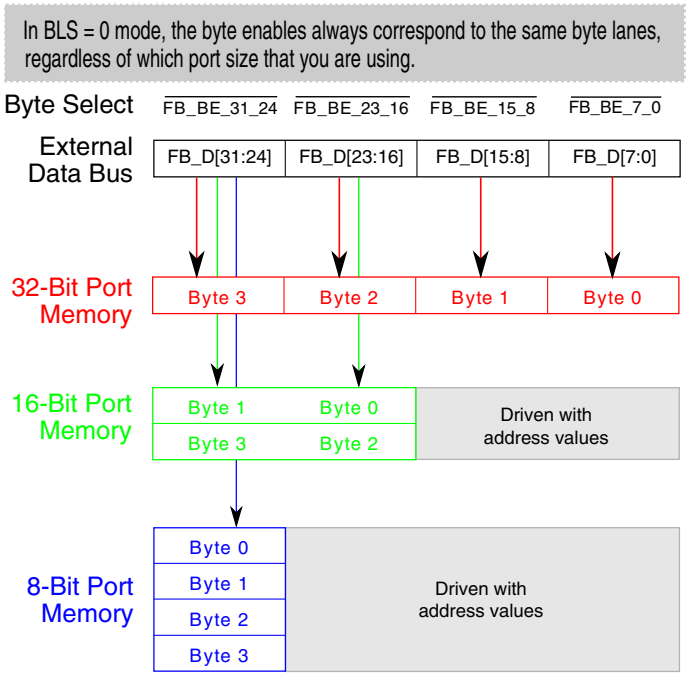



**Figure 31-23. Connections for external memory port sizes (CSCRn[BLS] = 0)**

The following figure shows the byte lanes that external memory or peripheral connects to and the sequential transfers of a 32-bit transfer for the supported port sizes when byte lane shift is enabled.

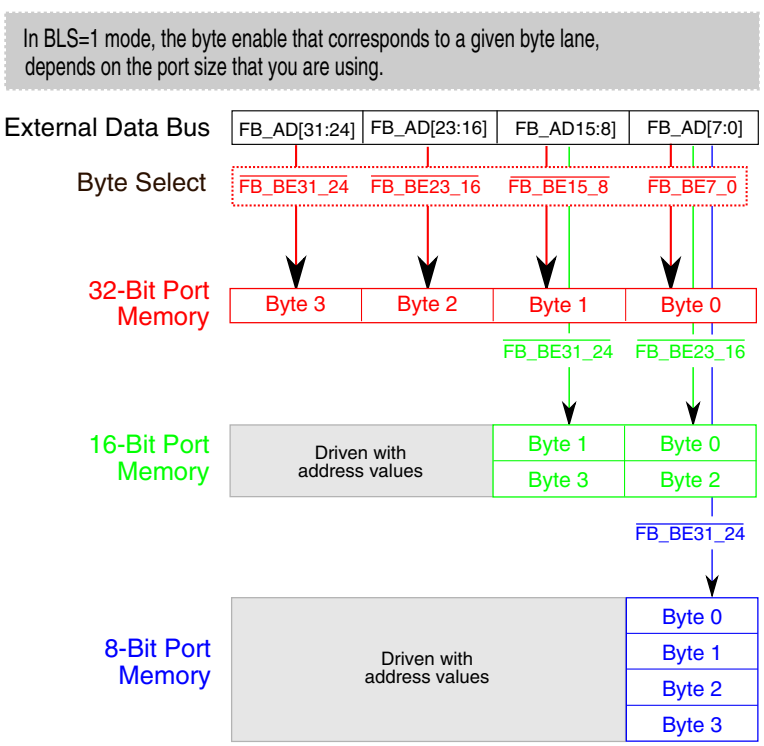



**Figure 31-24. Connections for external memory port sizes (CSCRn[BLS] = 1)**

## 31.4.9 Address/data bus multiplexing

FlexBus supports a single 32-bit wide multiplexed address and data bus (FB_AD31–FB_AD0). FlexBus always drives the full 32-bit address on the first clock of a bus cycle. During the data phase, the FB_AD31– FB_AD0 lines used for data are determined by the programmed port size and BLS setting for the corresponding chip-select. FlexBus continues to drive the address on any FB_AD31– FB_AD0 lines not used for data.

### 31.4.9.1 FlexBus multiplexed operating modes for CSCRn[BLS]=0

This table shows the supported combinations of address and data bus widths when CSCRn[BLS] is 0b.

| Port size and phase | | FB_AD | | | |
|---|---|---|---|---|---|
| | | 31–24 | 23–16 | 15–8 | 7–0 |
| 32-bit | Address phase | Address | | | |
| | Data phase | Data | | | |
| 16-bit | Address phase | Address | | | |
| | Data phase | Data | | Address | |
| 8-bit | Address phase | Address | | | |
| | Data phase | Data | Address | | |

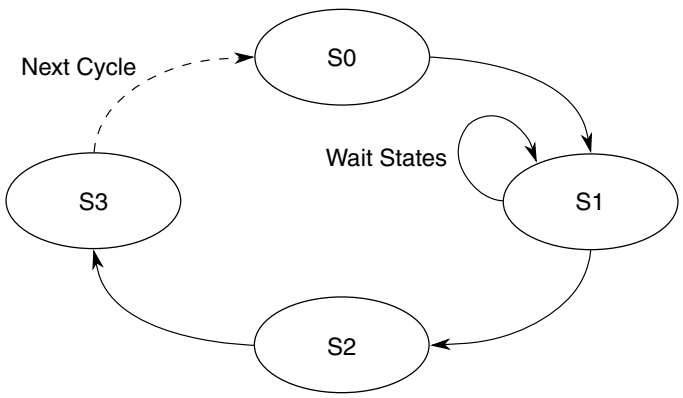
### 31.4.9.2 FlexBus multiplexed operating modes for CSCRn[BLS]=1

This table shows the supported combinations of address and data bus widths when CSCRn[BLS] is 1b.

| Port size and phase | | FB_AD | | | |
|---|---|---|---|---|---|
| | | 31–24 | 23–16 | 15–8 | 7–0 |
| 32-bit | Address phase | Address | | | |
| | Data phase | Data | | | |
| 16-bit | Address phase | Address | | | |
| | Data phase | Address | | Data | |
| 8-bit | Address phase | Address | | | |
| | Data phase | Address | | | Data |

## 31.4.10 Data transfer states

Basic data transfers occur in four clocks or states. (See Figure 31-26 and Figure 31-28 for examples of basic data transfers.) The FlexBus state machine controls the data-transfer operation. This figure shows the state-transition diagram for basic read and write cycles.




The states are described in this table.

| State | Cycle | Description |
|---|---|---|
| S0 | All | The read or write cycle is initiated. On the rising clock edge, FlexBus:<br>• Places a valid address on FB_ADn<br>• Asserts $\overline{\text{FB\_TS}}$/FB_ALE<br>• Drives FB_R/$\overline{\text{W}}$ high for a read and low for a write |
| S1 | All | FlexBus:<br>• Negates $\overline{\text{FB\_TS}}$/FB_ALE on the rising edge of FB_CLK<br>• Asserts FB_CSn<br>• Drives the data on FB_AD31– FB_AD*X* for writes<br>• Tristates FB_AD31– FB_AD*X* for reads<br>• Continues to drive the address on FB_AD pins that are unused for data<br><br>If the external memory or perihperal asserts $\overline{\text{FB\_TA}}$, then the process moves to S2. If $\overline{\text{FB\_TA}}$ is not asserted internally or externally, then S1 repeats. |
| | Read | The external memory or peripheral drives the data before the next rising edge of FB_CLK (the rising edge that begins S2) with $\overline{\text{FB\_TA}}$ asserted. |
| S2 | All | For internal termination, FlexBus negates $\overline{\text{FB_CS}}$n and the transfer is complete. For external termination, the external memory or peripheral negates $\overline{\text{FB\_TA}}$, and FlexBus negates $\overline{\text{FB_CS}}$n after the rising edge of FB_CLK at the end of S2. |
| | Read | FlexBus latches the data on the rising clock edge entering S2. The external memory or peripheral can stop driving the data after this edge or continue to drive the data until the end of S3 or through any additional address hold cycles. |
| S3 | All | FlexBus invalidates the address, data, and FB_R/$\overline{\text{W}}$ on the rising edge of FB_CLK at the beginning of S3, terminating the transfer. |

## 31.4.11 FlexBus Timing Examples

### Note

The timing diagrams throughout this section use signal names that may not be included on your particular device. Ignore these extraneous signals.

### Note

Throughout this section:

- FB_D[*X*] indicates a 32-, 16-, or 8-bit wide data bus
- FB_A[*Y*] indicates an address bus that can be 32, 24, or 16 bits wide.

### 31.4.11.1 Basic Read Bus Cycle

During a read cycle, the MCU receives data from memory or a peripheral device. The following figure shows a read cycle flowchart.

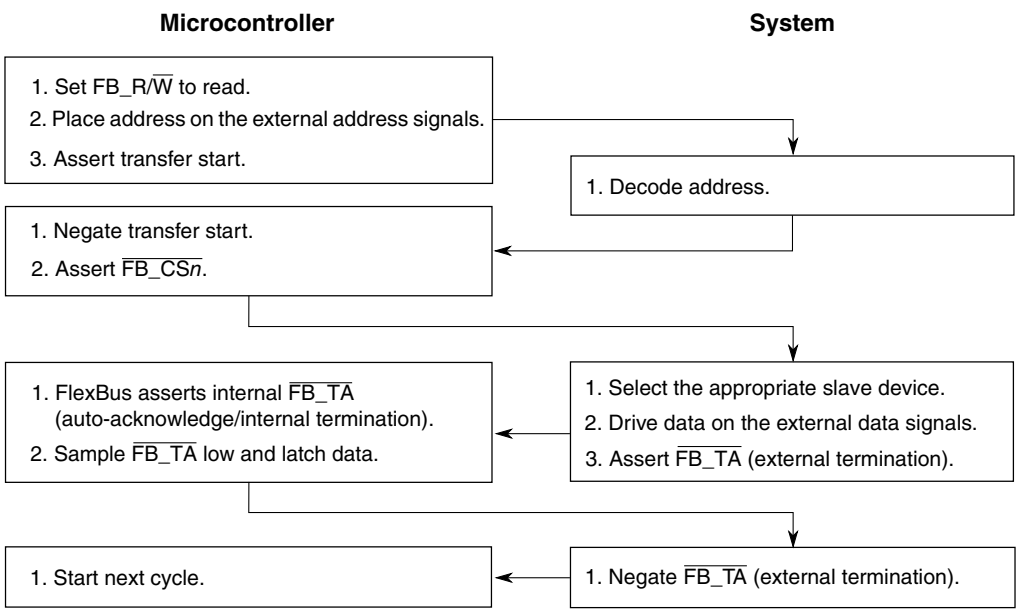



**Figure 31-25. Read Cycle Flowchart**

The read cycle timing diagram is shown in the following figure.

### Note

FB_TA does not have to be driven by the external device for internally-terminated bus cycles.

### Note

The processor drives the data lines during the first clock cycle of the transfer with the full 32-bit address. This may be ignored by standard connected devices using non-multiplexed address and data buses. However, some applications may find this feature beneficial.

The address and data busses are muxed between the FlexBus and another module. At the end of the read bus cycles the address signals are indeterminate.

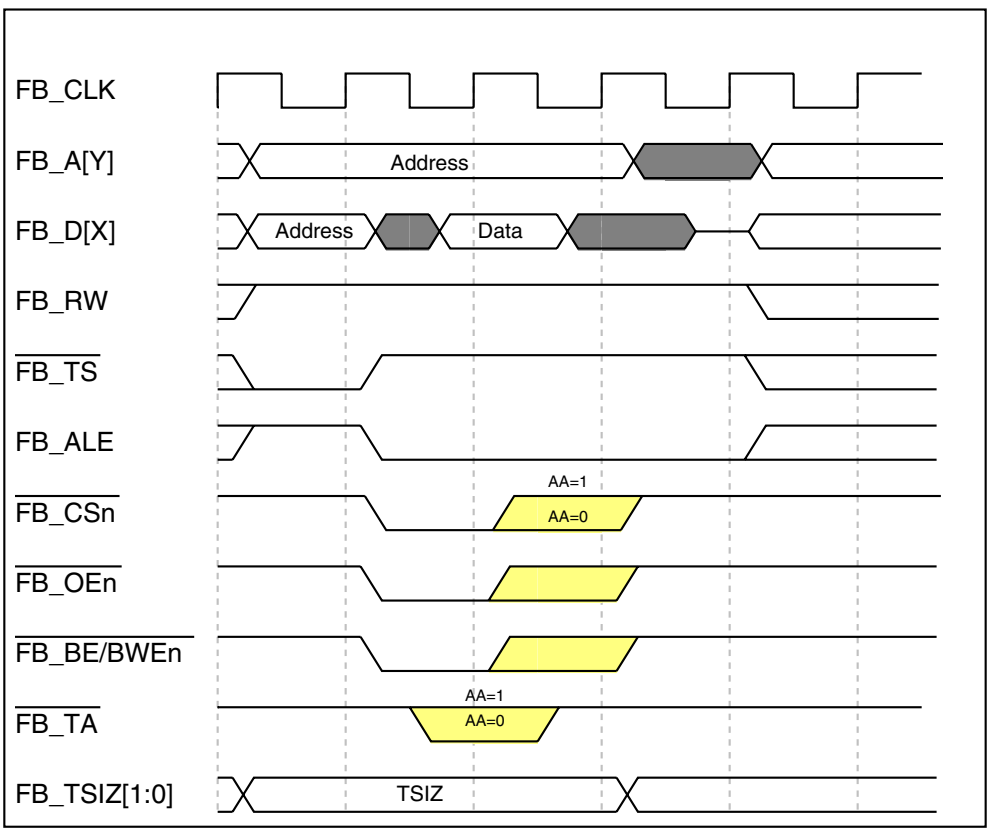



**Figure 31-26. Basic Read-Bus Cycle**

## 31.4.11.2 Basic Write Bus Cycle

During a write cycle, the device sends data to memory or to a peripheral device. The following figure shows the write cycle flowchart.

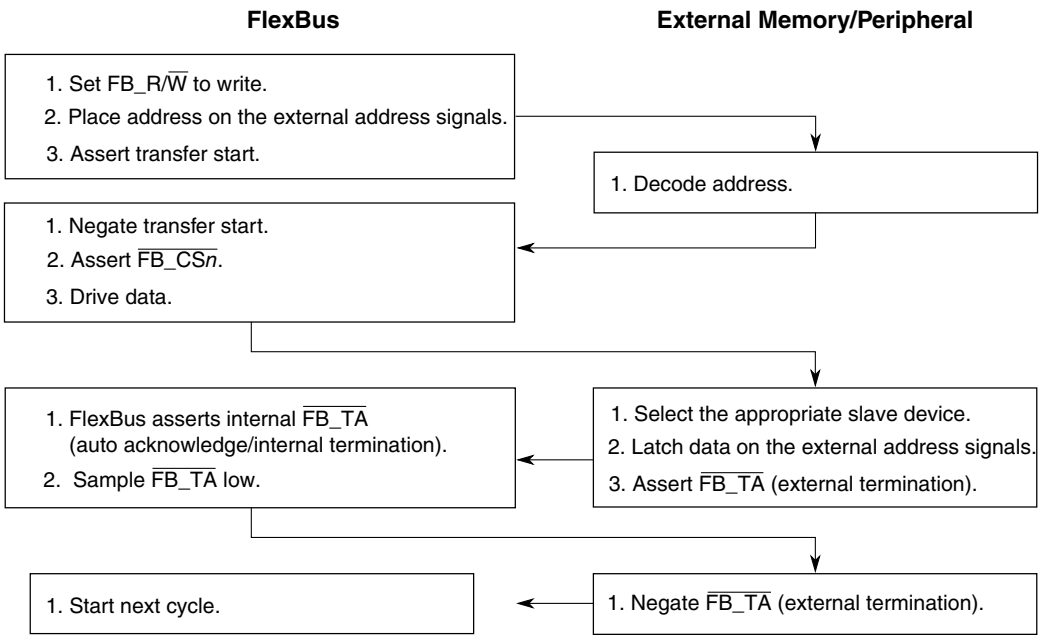



**Figure 31-27. Write-Cycle Flowchart**

The following figure shows the write cycle timing diagram.

## Note

The address and data busses are muxed between the FlexBus and another module. At the end of the write bus cycles, the address signals are indeterminate.

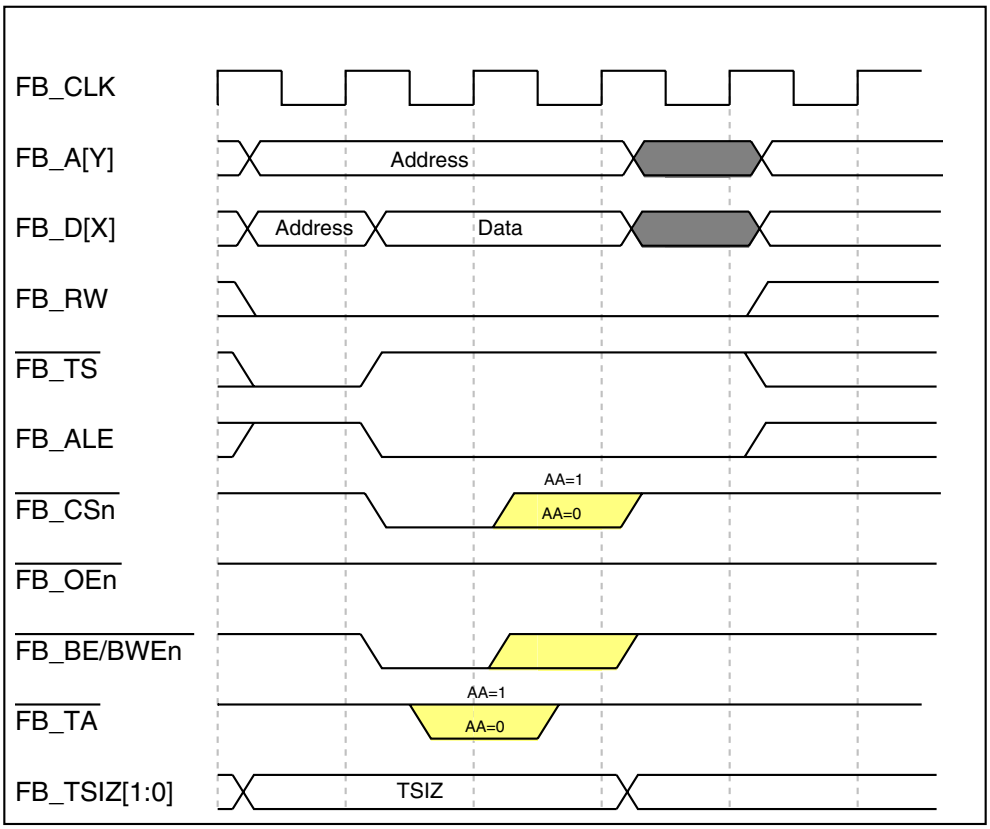



**Figure 31-28. Basic Write-Bus Cycle**

### 31.4.11.3 Bus Cycle Sizing

This section shows timing diagrams for various port size scenarios.

#### 31.4.11.3.1 Bus Cycle Sizing—Byte Transfer, 8-bit Device, No Wait States

The following figure illustrates the basic byte read transfer to an 8-bit device with no wait states:

- The address is driven on the full FB_AD[31:8] bus in the first clock.
- The device tristates FB_AD[31:24] on the second clock and continues to drive address on FB_AD[23:0] throughout the bus cycle.
- The external device returns the read data on FB_AD[31:24] and may tristate the data line or continue driving the data one clock after $\overline{\text{FB\_TA}}$ is sampled asserted.

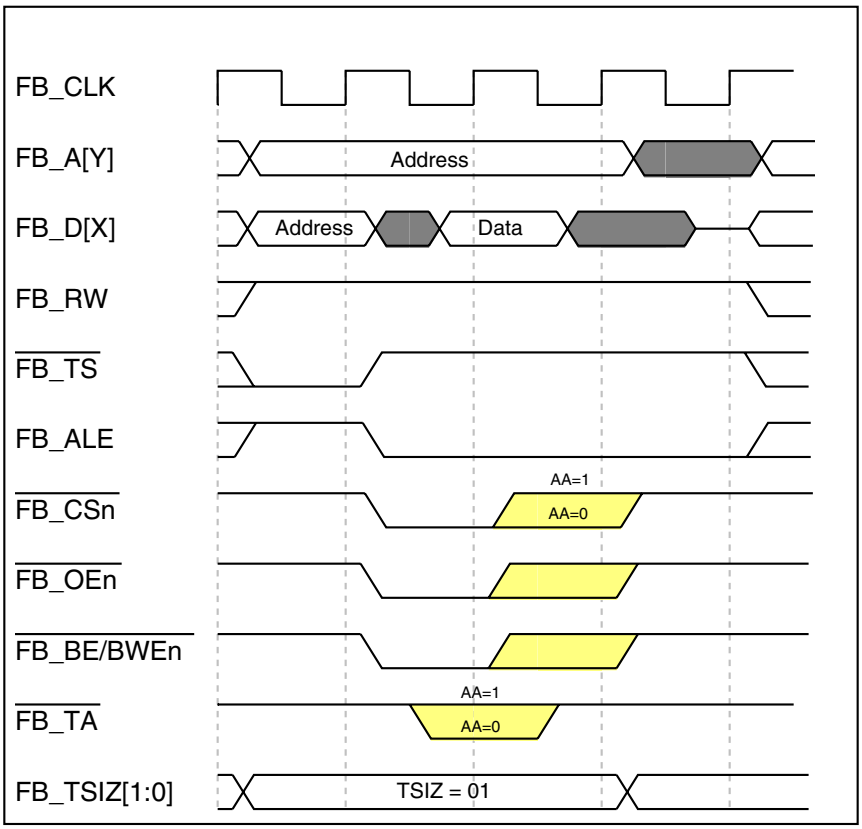



**Figure 31-29. Single Byte-Read Transfer**

The following figure shows the similar configuration for a write transfer. The data is driven from the second clock on FB_AD[31:24].

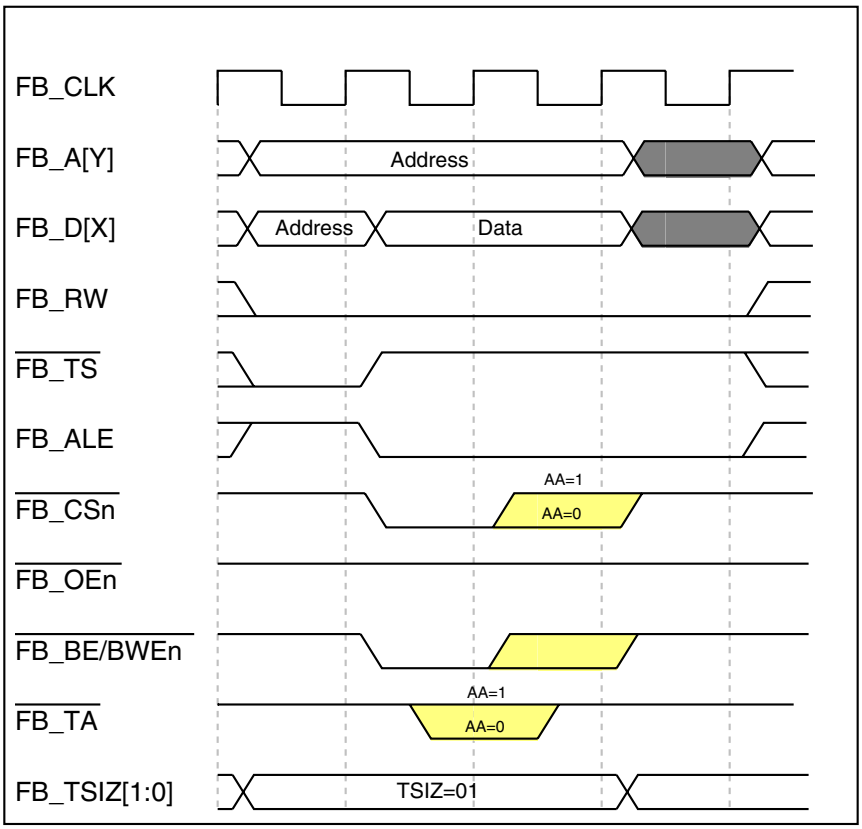



**Figure 31-30. Single Byte-Write Transfer**

### 31.4.11.3.2 Bus Cycle Sizing—Word Transfer, 16-bit Device, No Wait States

The following figure illustrates the basic word read transfer to a 16-bit device with no wait states.

- The address is driven on the full FB_AD[31:8] bus in the first clock.
- The device tristates FB_AD[31:16] on the second clock and continues to drive address on FB_AD[15:0] throughout the bus cycle.
- The external device returns the read data on FB_AD[31:16] and may tristate the data line or continue driving the data one clock after $\overline{\text{FB\_TA}}$ is sampled asserted.

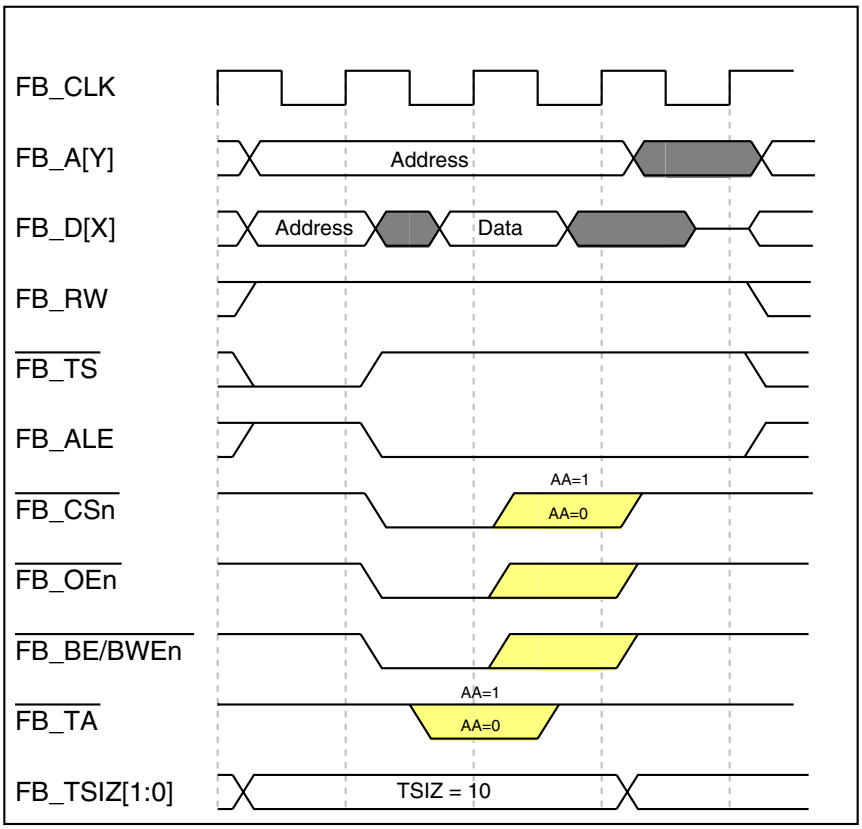



**Figure 31-31. Single Word-Read Transfer**

The following figure shows the similar configuration for a write transfer. The data is driven from the second clock on FB_AD[31:16].

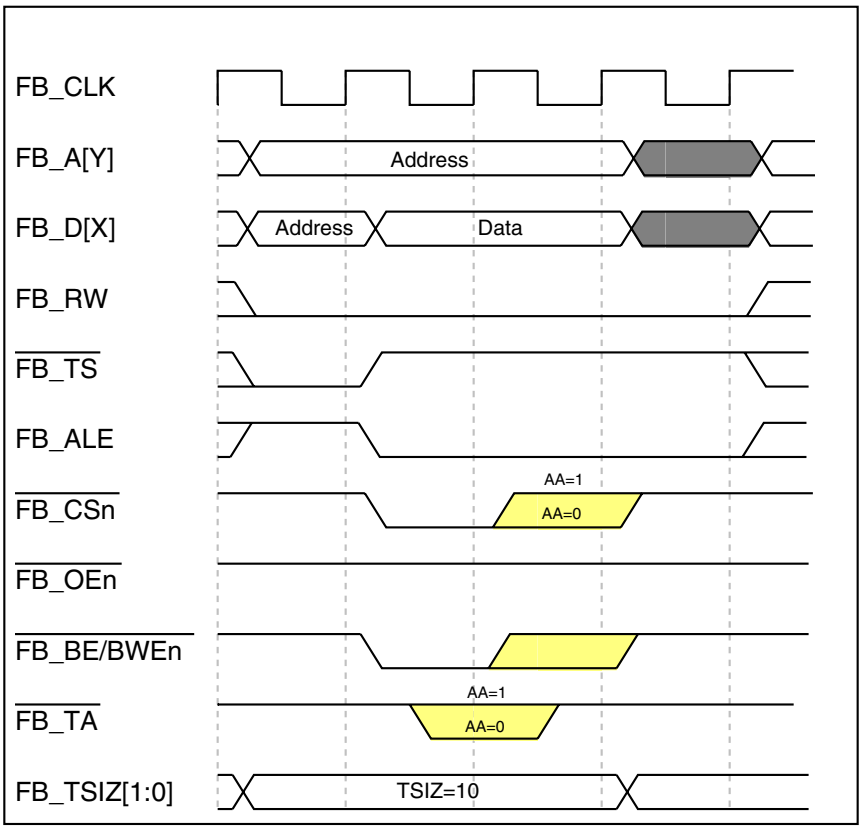



**Figure 31-32. Single Word-Write Transfer**

### 31.4.11.3.3 Bus Cycle Sizing—Longword Transfer, 32-bit Device, No Wait States

The following figure depicts a longword read from a 32-bit device.

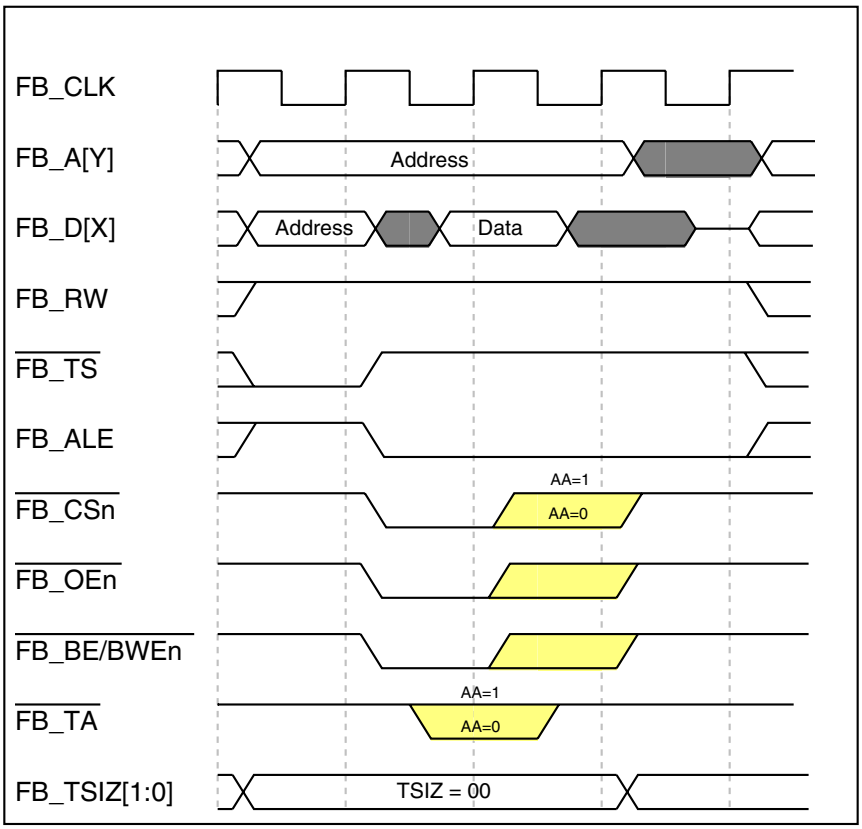



**Figure 31-33. Longword-Read Transfer**

The following figure illustrates the longword write to a 32-bit device.

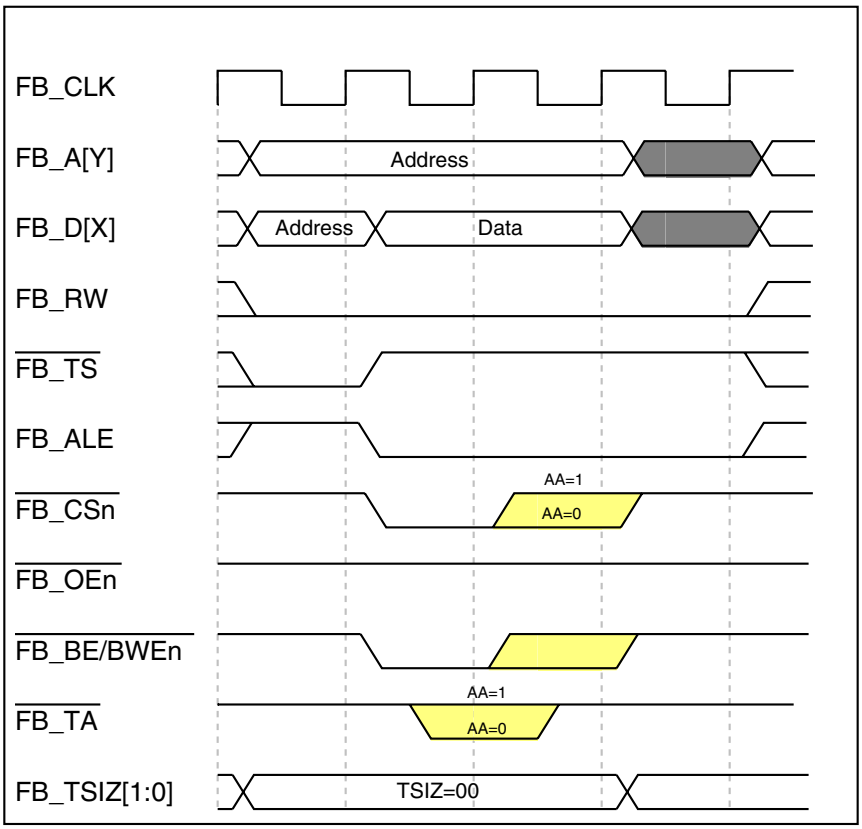



**Figure 31-34. Longword-Write Transfer**

### 31.4.11.4 Timing Variations

The FlexBus module has several features that can change the timing characteristics of a basic read- or write-bus cycle to provide additional address setup, address hold, and time for a device to provide or latch data.

#### 31.4.11.4.1 Wait States

Wait states can be inserted before each beat of a transfer by programming the CSCR*n* registers. Wait states can give the peripheral or memory more time to return read data or sample write data.

The following figures show the basic read and write bus cycles (also shown in Figure 31-26 and Figure 31-31) with the default of no wait states respectively.

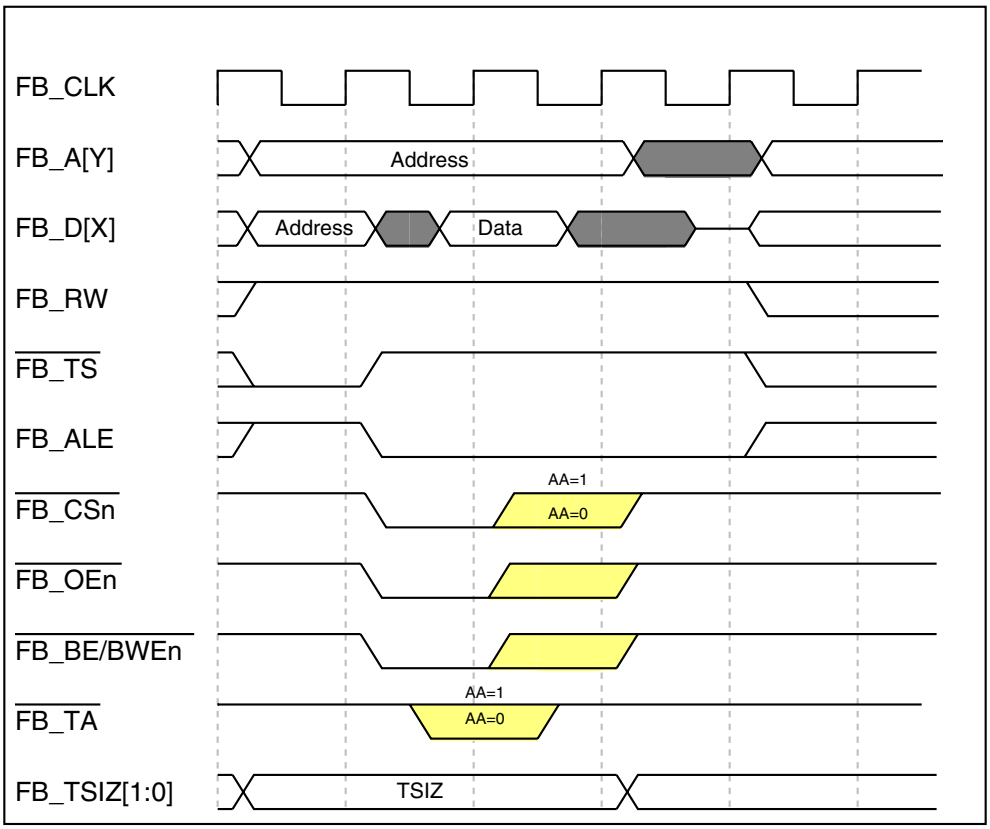



**Figure 31-35. Basic Read-Bus Cycle (No Wait States)**

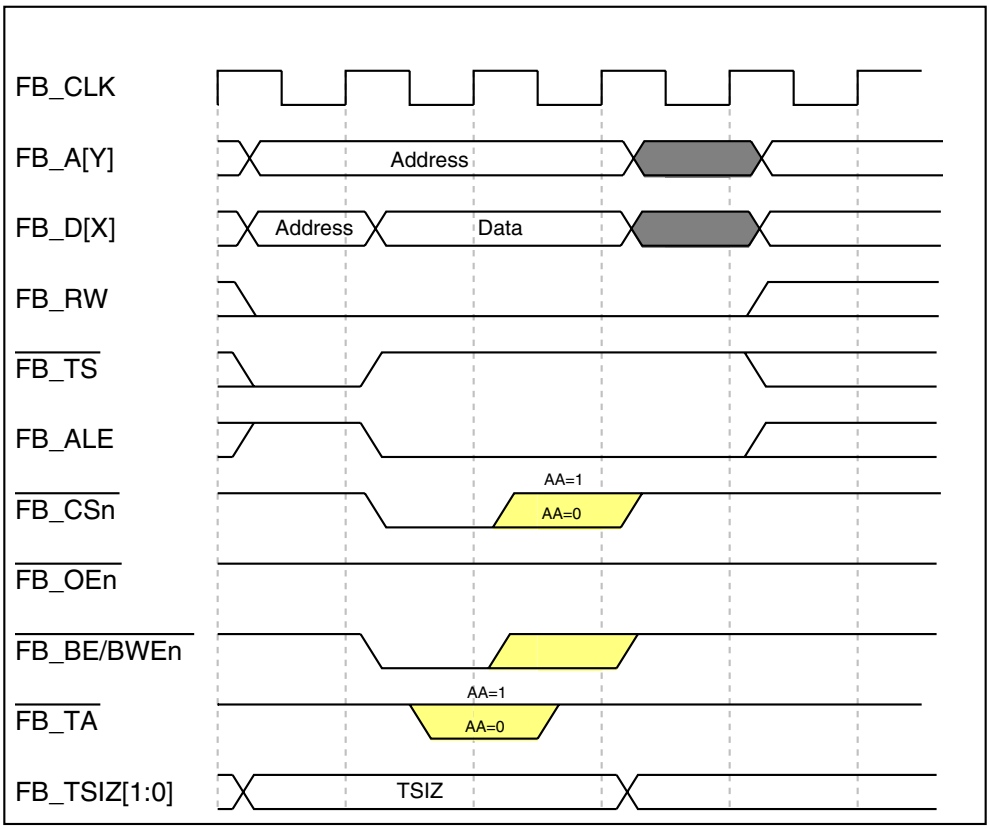



**Figure 31-36. Basic Write-Bus Cycle (No Wait States)**

If wait states are used, the S1 state repeats continuously until the chip-select auto-acknowledge unit asserts internal transfer acknowledge or the external $\overline{\text{FB\_TA}}$ is recognized as asserted. The following figures show a read and write cycle with one wait state respectively.

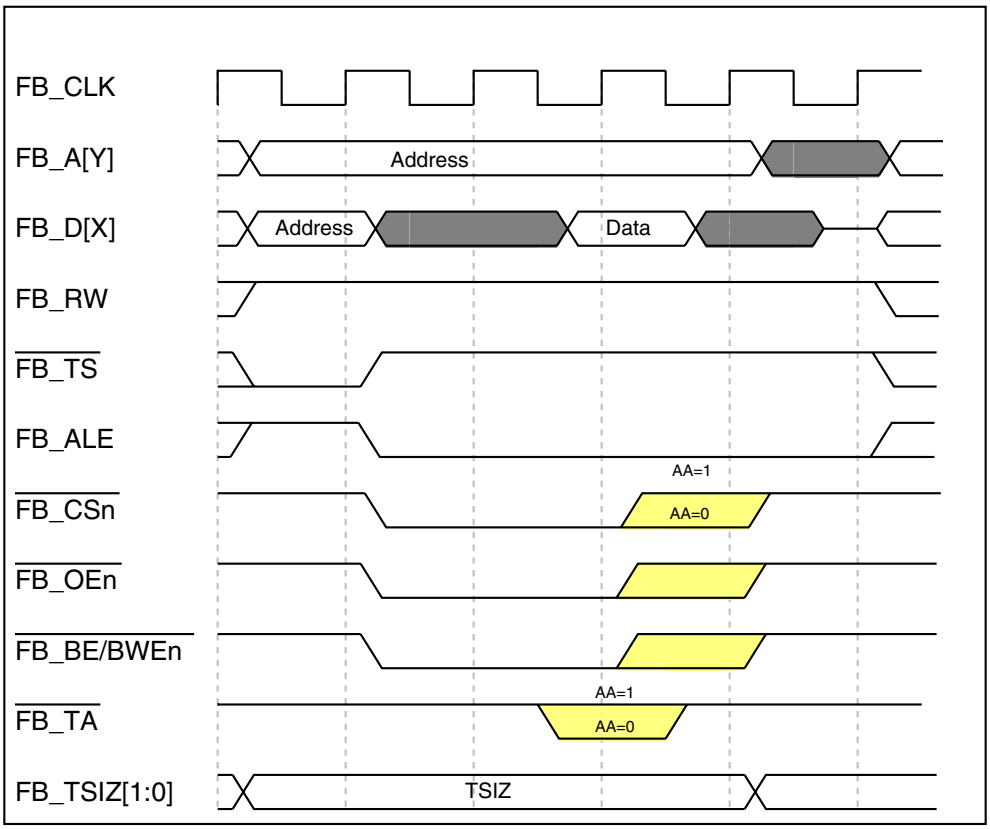



**Figure 31-37. Read-Bus Cycle (One Wait State)**

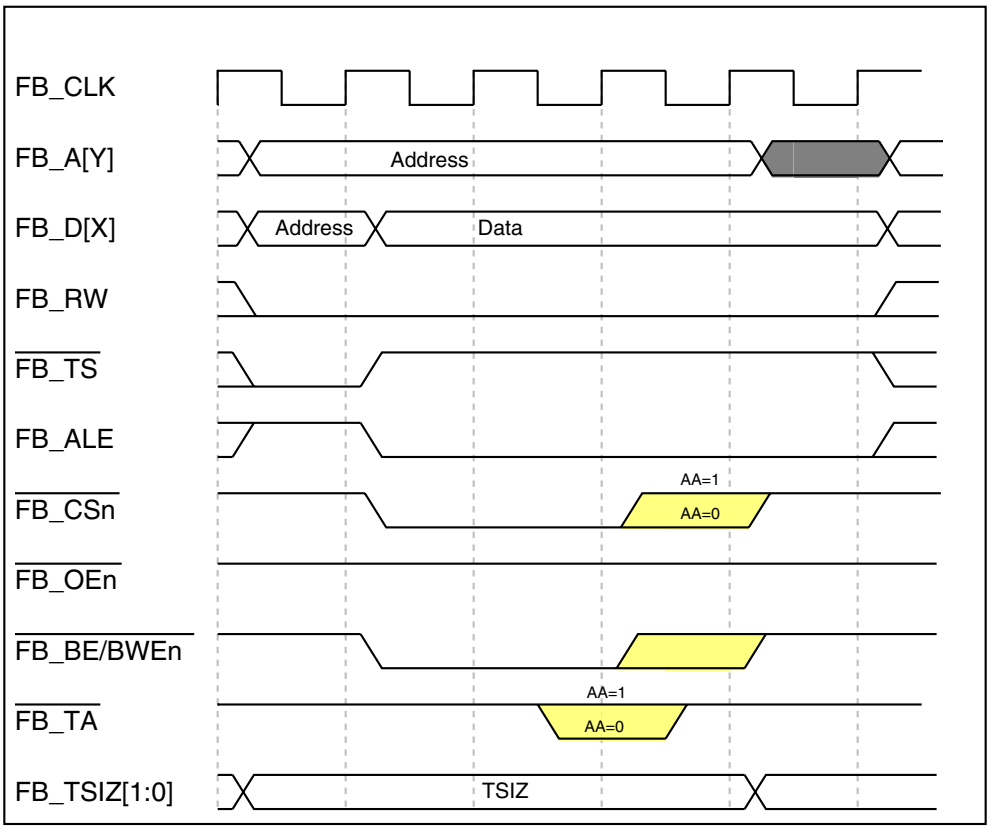



**Figure 31-38. Write-Bus Cycle (One Wait State)**

### 31.4.11.4.2 Address Setup and Hold

The timing of the assertion and negation of the chip selects, byte selects, and output enable can be programmed on a chip-select basis. Each chip-select can be programmed to assert one to four clocks after transfer start/address-latch enable ($\overline{\text{FB\_TS}}$/FB_ALE) is asserted. The following figures show read- and write-bus cycles with two clocks of address setup respectively.

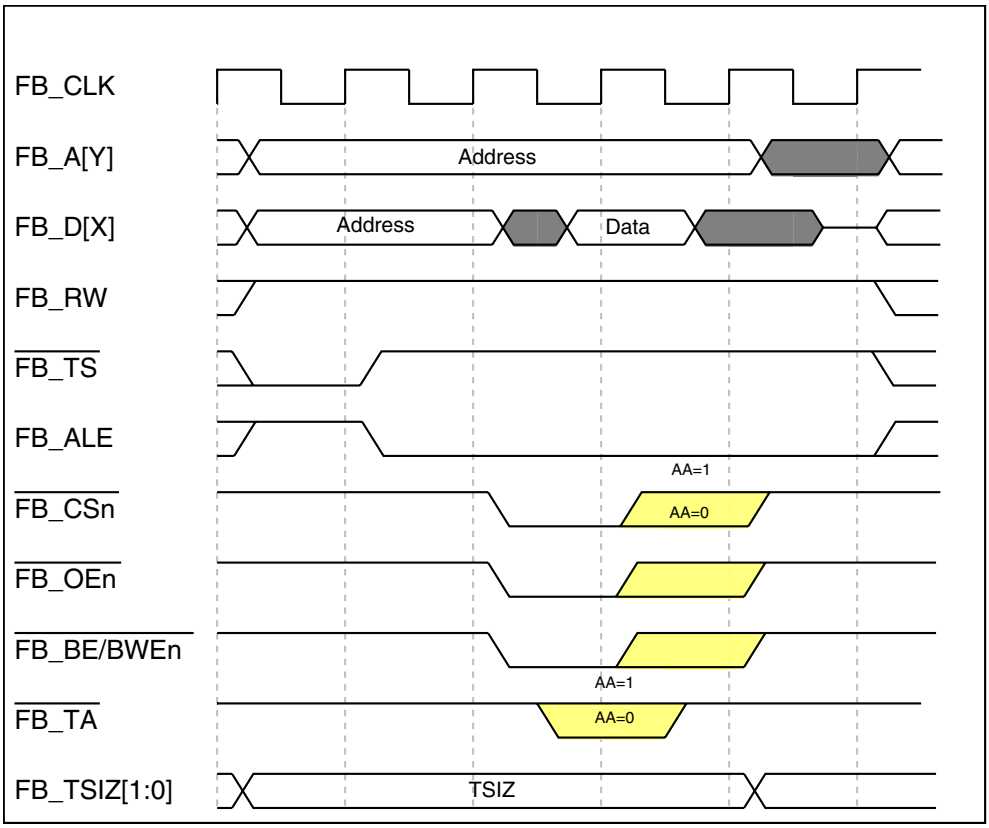



**Figure 31-39. Read-Bus Cycle with Two-Clock Address Setup (No Wait States)**

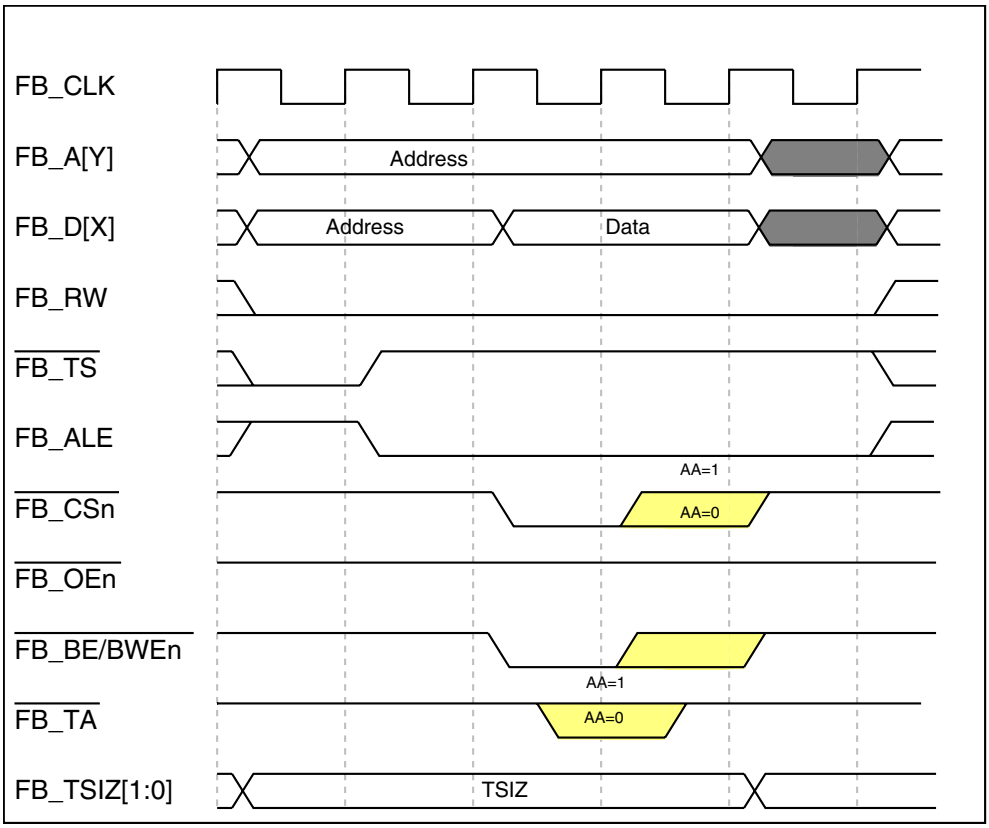



**Figure 31-40. Write-Bus Cycle with Two Clock Address Setup (No Wait States)**

In addition to address setup, a programmable address hold option for each chip select exists. Address and attributes can be held one to four clocks after chip-select, byte-selects, and output-enable negate. The following figures show read and write bus cycles with two clocks of address hold respectively.

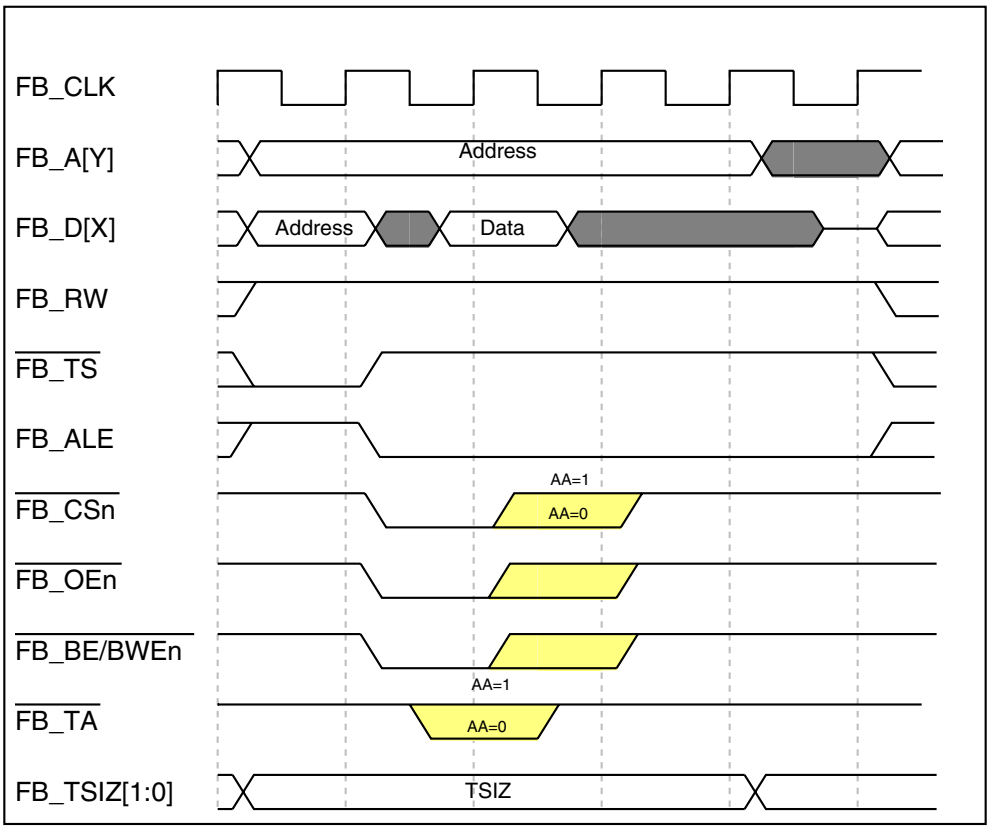



**Figure 31-41. Read Cycle with Two-Clock Address Hold (No Wait States)**

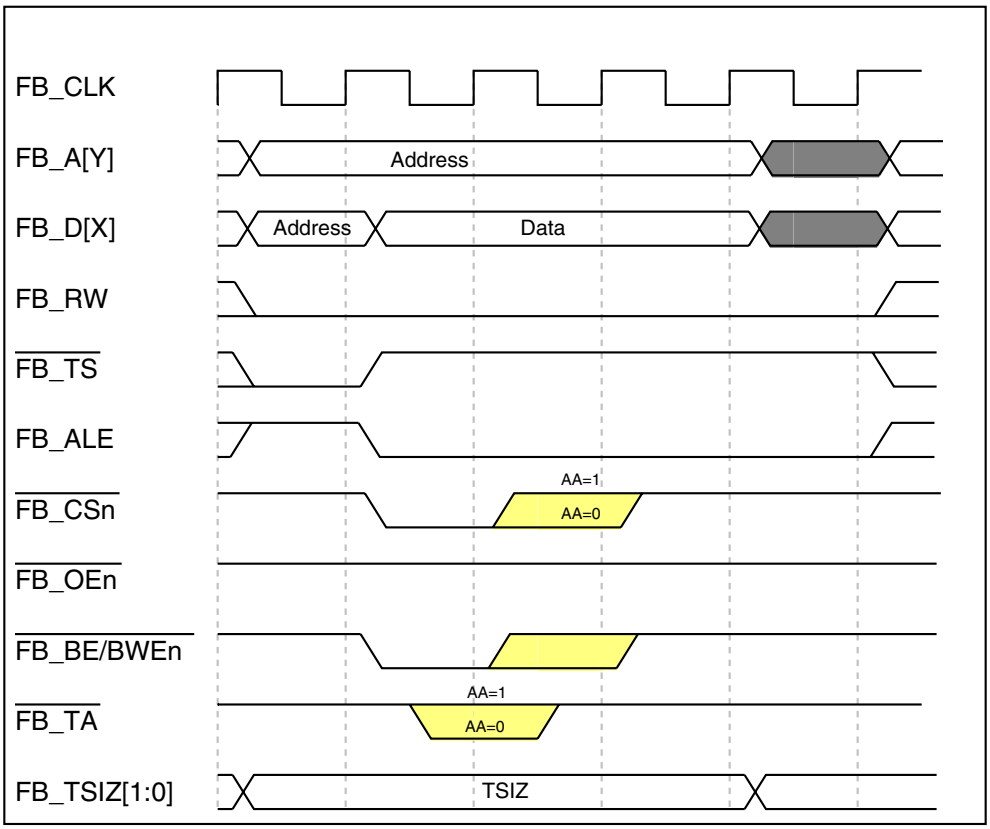



**Figure 31-42. Write Cycle with Two-Clock Address Hold (No Wait States)**

The following figure shows a bus cycle using address setup, wait states, and address hold.

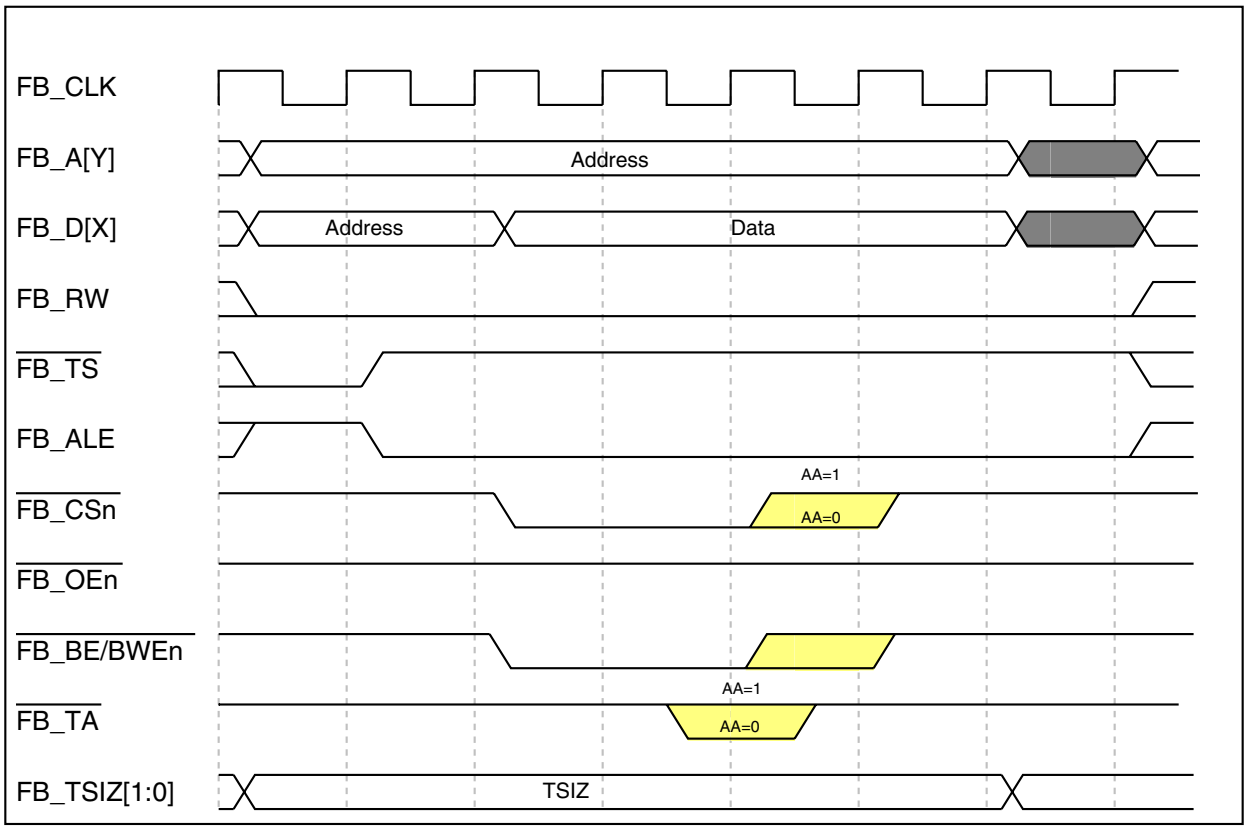



**Figure 31-43. Write Cycle with Two-Clock Address Setup and Two-Clock Hold (One Wait State)**

## 31.4.12 Burst cycles

The chip can be programmed to initiate burst cycles if its transfer size exceeds the port size of the selected destination. The initiation of a burst cycle is encoded on the transfer size pins (FB_TSIZ[1:0]). For burst transfers to smaller port sizes, FB_TSIZ[1:0] indicates the size of the entire transfer. For example, with bursting enabled, a 16-bit transfer to an 8-bit port takes two beats (two byte-sized transfers), for which FB_TSIZ[1:0] equals 10b throughout. A 32-bit transfer to an 8-bit port takes four beats (four byte-sized transfers), for which FB_TSIZ[1:0] equals 00b throughout.

### 31.4.12.1 Enabling and inhibiting burst

The CSCRn registers enable bursting for reads, writes, or both.

Memory spaces can be declared burst-inhibited for reads and writes by writing 0b to the appropriate CSCRn[BSTR] and CSCRn[BSTW] fields.

### 31.4.12.2 Transfer size and port size translation

With bursting disabled, any transfer larger than the port size breaks into multiple individual transfers (e.g. <Addr><Data><Addr+1><Data><Addr+2><Data>). With bursting enabled, any transfer larger than the port size results in a burst cycle of multiple beats (e.g. <Addr><Data><Data><Data>). The following table shows the result of such transfer translations.

| Port size PS[1:0] | Transfer size FB_TSIZ[1:0] | Burst-inhibited: Number of transfers<br>Burst enabled: Number of beats |
|---|---|---|
| 01b (8 bit) | 10b (16 bits) | 2 |
| | 00b (32 bits) | 4 |
| | 11b (16 bytes) | 16 |
| 1Xb (16 bit) | 00b (32 bits) | 2 |
| | 11b (16 bytes) | 8 |
| 00b (32 bit) | 11b (line) | 4 |

The FlexBus can support X-1-1-1 burst cycles to maximize system performance, where X is the primary number of wait states (max 63). Delaying termination of the cycle can add wait states. If internal termination is used, different wait state counters can be used for the first access and the following beats.

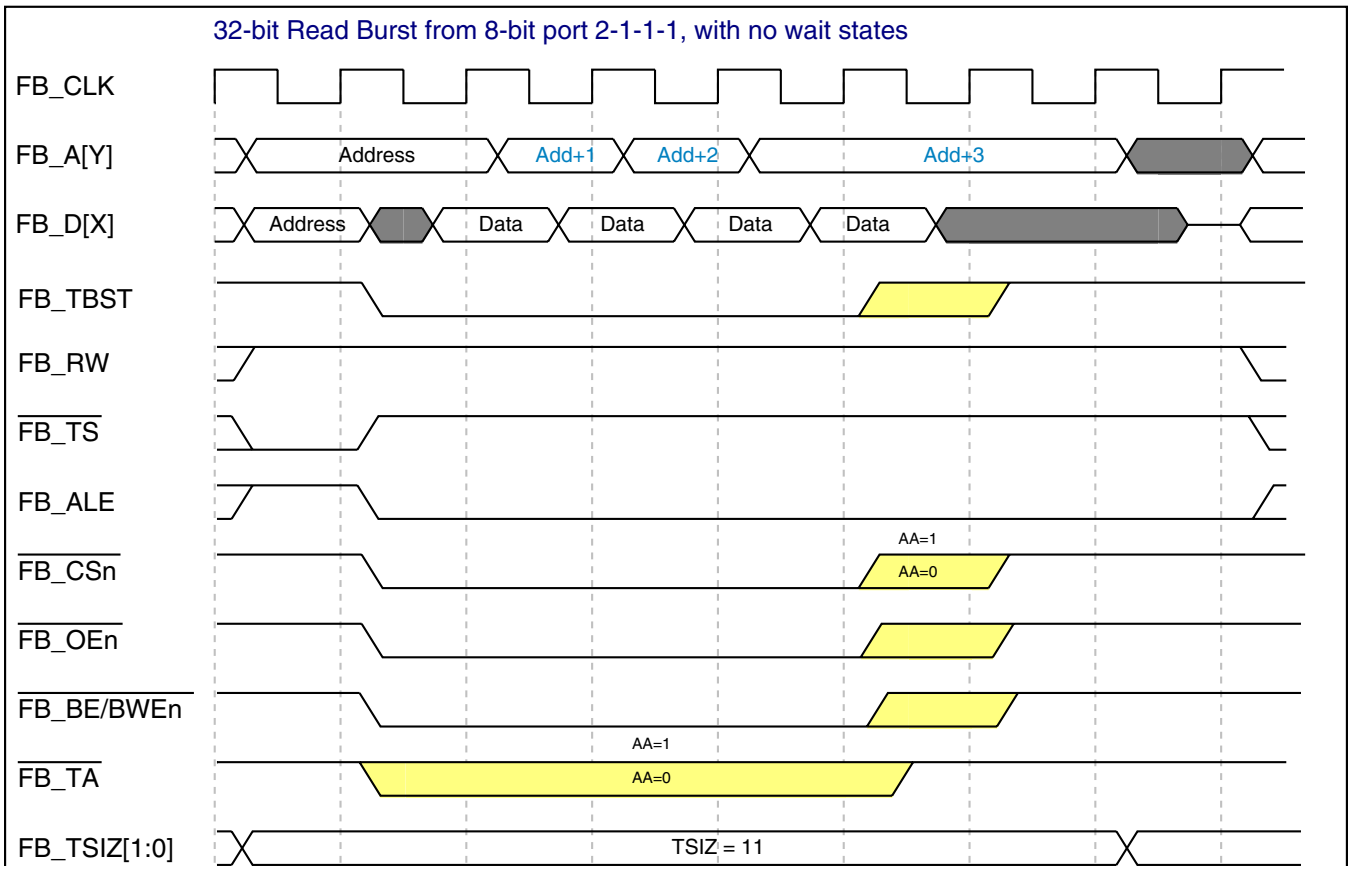
### 31.4.12.3 32-bit-Read burst from 8-Bit port 2-1-1-1 (no wait states)

The following figure shows a 32-bit read to an 8-bit external chip programmed for burst enable. The transfer results in a 4-beat burst and the data is driven on FB_AD[31:24]. The transfer size is driven at 32-bit (00b) throughout the bus cycle.

**Note**

In non-multiplexed address/data mode, the address on FB_A increments only during internally-terminated burst cycles. The first address is driven throughout the entire burst for externally-terminated cycles.

In multiplexed address/data mode, the address is driven on FB_AD only during the first cycle for all terminated cycles.

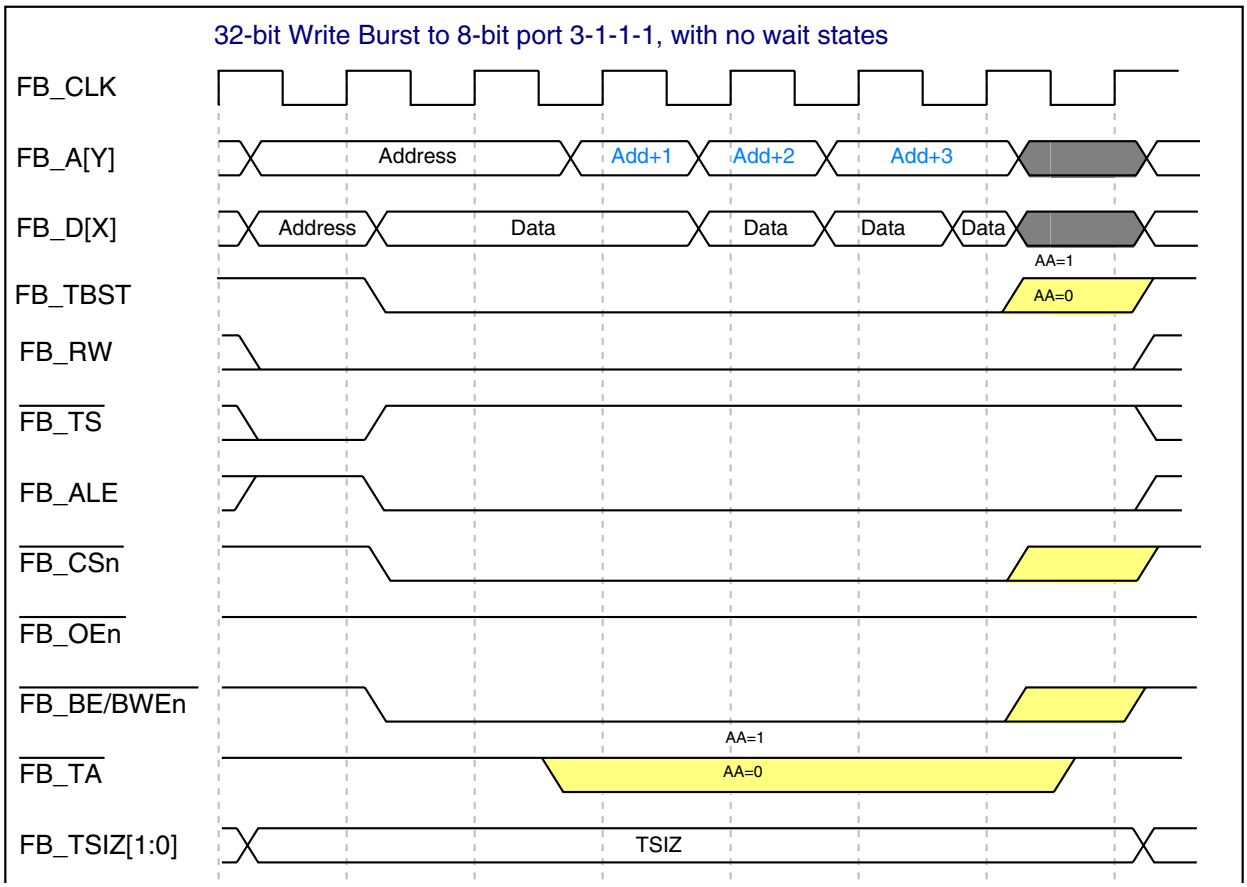



### 31.4.12.4 32-bit-Write burst to 8-Bit port 3-1-1-1 (no wait states)

The following figure shows a 32-bit write to an 8-bit external chip with burst enabled. The transfer results in a 4-beat burst and the data is driven on FB_AD[31:24]. The transfer size is driven at 32-bit (00b) throughout the bus cycle.

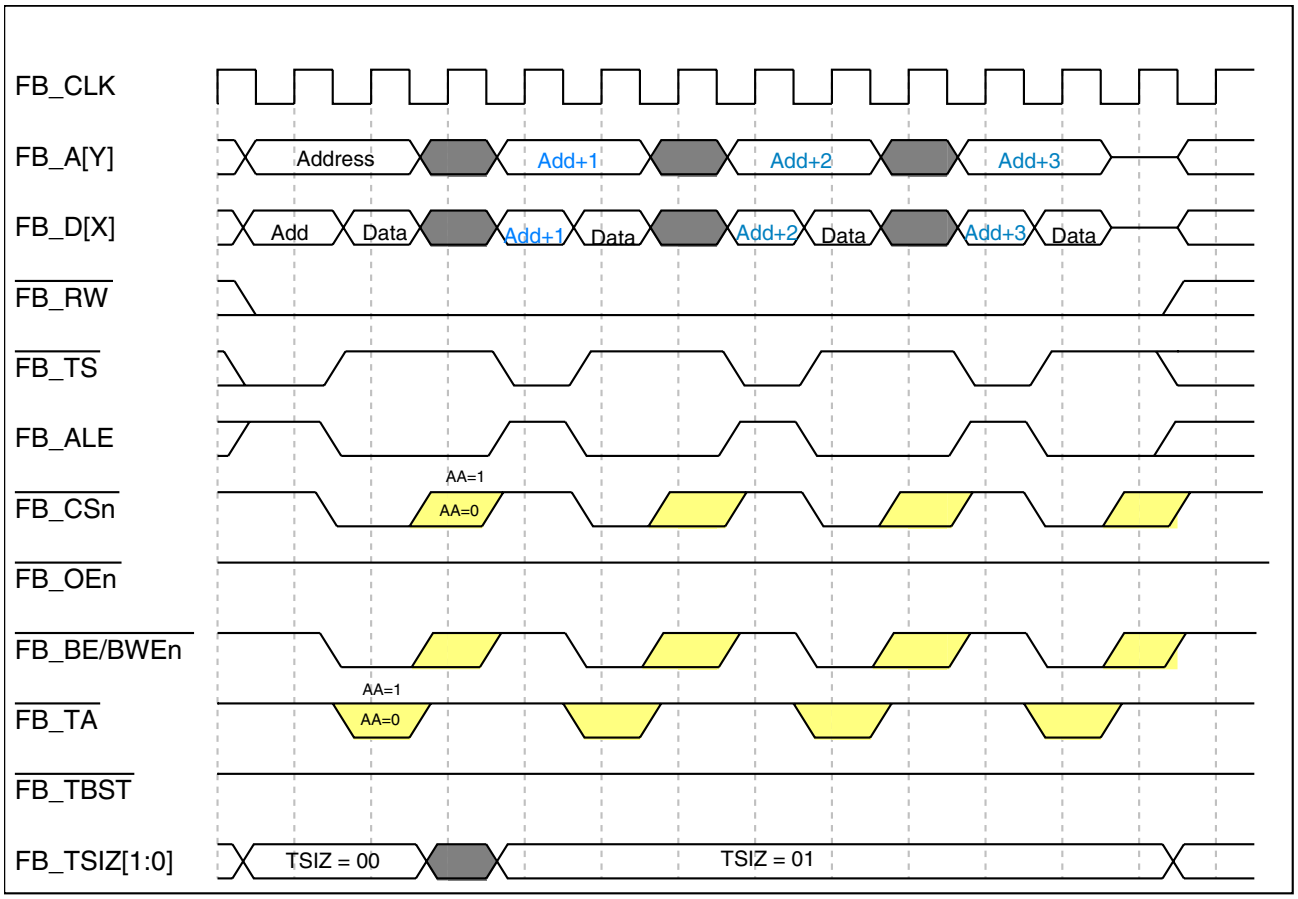
**Note**

The first beat of any write burst cycle has at least one wait state. If the bus cycle is programmed for zero wait states (CSCRn[WS] = 0b), one wait state is added. Otherwise, the programmed number of wait states are used.




### 31.4.12.5 32-bit-write burst-inhibited to 8-bit port (no wait states)

The following figure shows a 32-bit write to an 8-bit device with burst inhibited. The transfer results in four individual transfers. The transfer size is driven at 32-bit (00b) during the first transfer and at byte (01b) during the next three transfers.

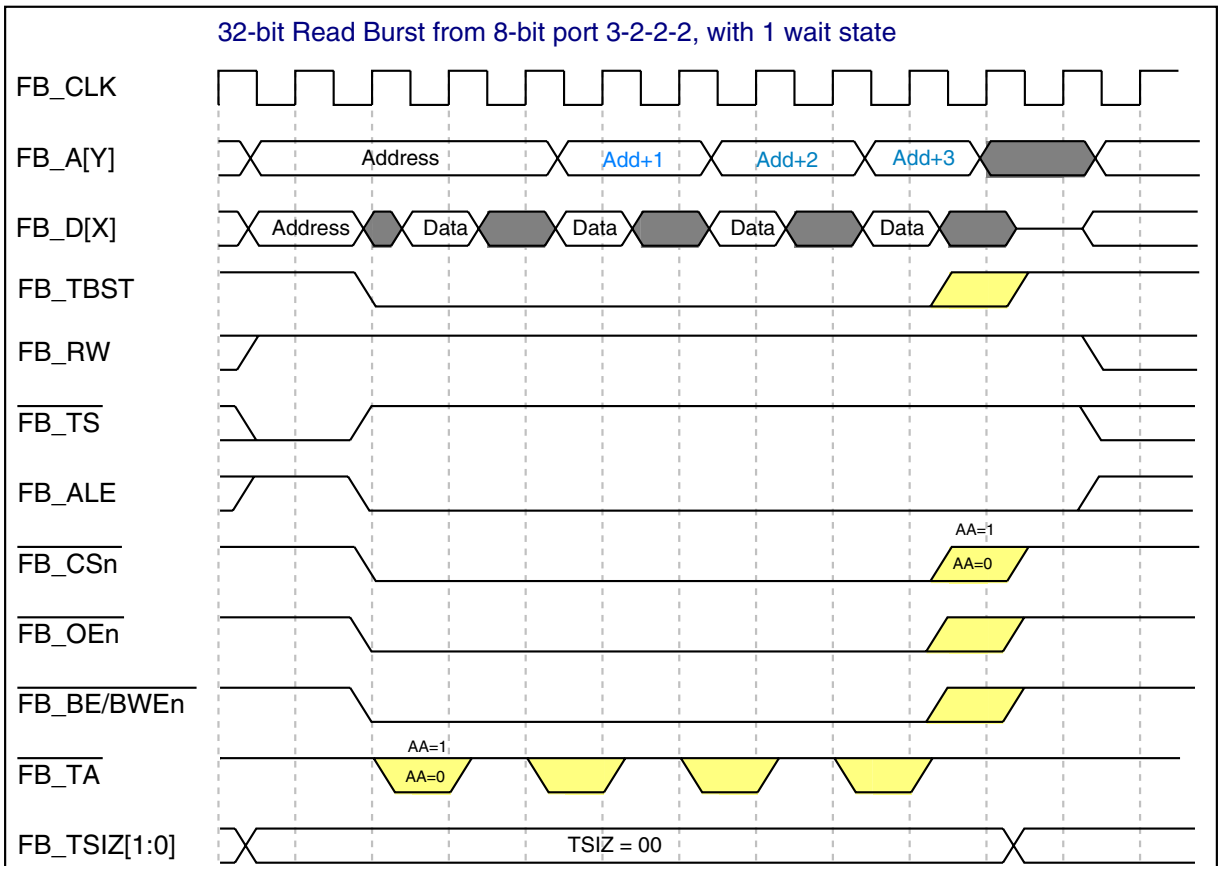



### 31.4.12.6 32-bit-read burst from 8-bit port 3-2-2-2 (one wait state)

The following figure illustrates another read burst transfer, but in this case a wait state is added between individual beats.

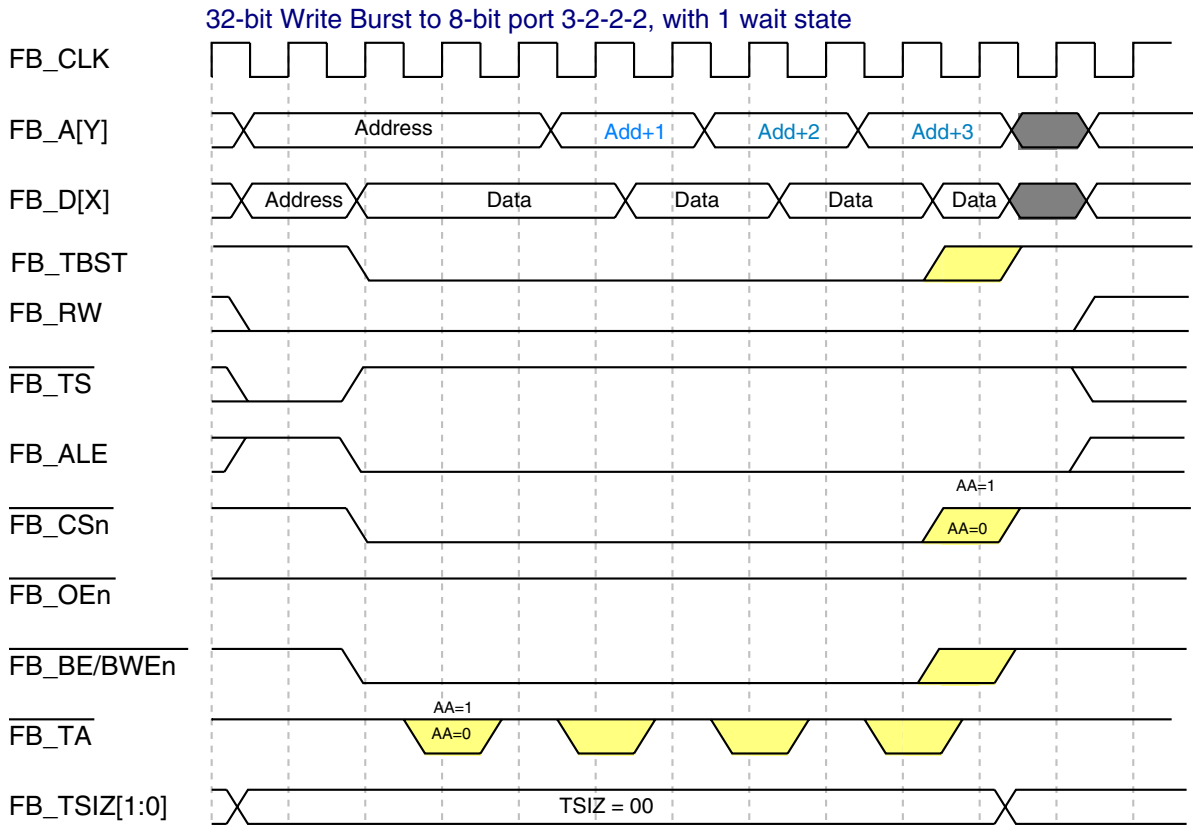
**Note**

CSCRn[WS] determines the number of wait states in the first beat. However, for subsequent beats, the CSCRn[WS] (or CSCRn[SWS] if CSCRn[SWSEN] = 1b) determines the number of wait states.




### 31.4.12.7 32-bit-write burst to 8-bit port 3-2-2-2 (one wait state)

The following figure illustrates a write burst transfer with one wait state.



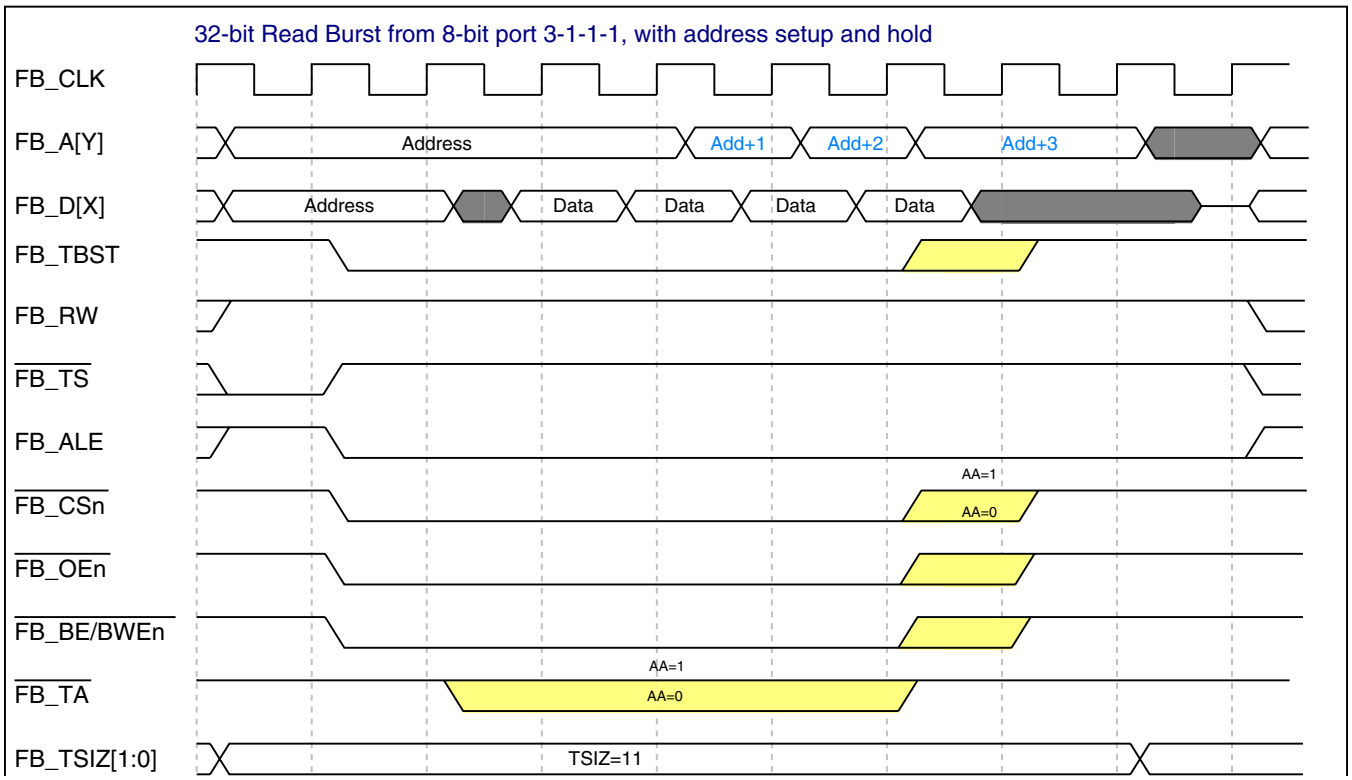

### 31.4.12.8 32-bit-read burst from 8-bit port 3-1-1-1 (address setup and hold)

If address setup and hold are used, only the first and last beat of the burst cycle are affected. The following figure shows a read cycle with one clock of address setup and address hold.

#### Note

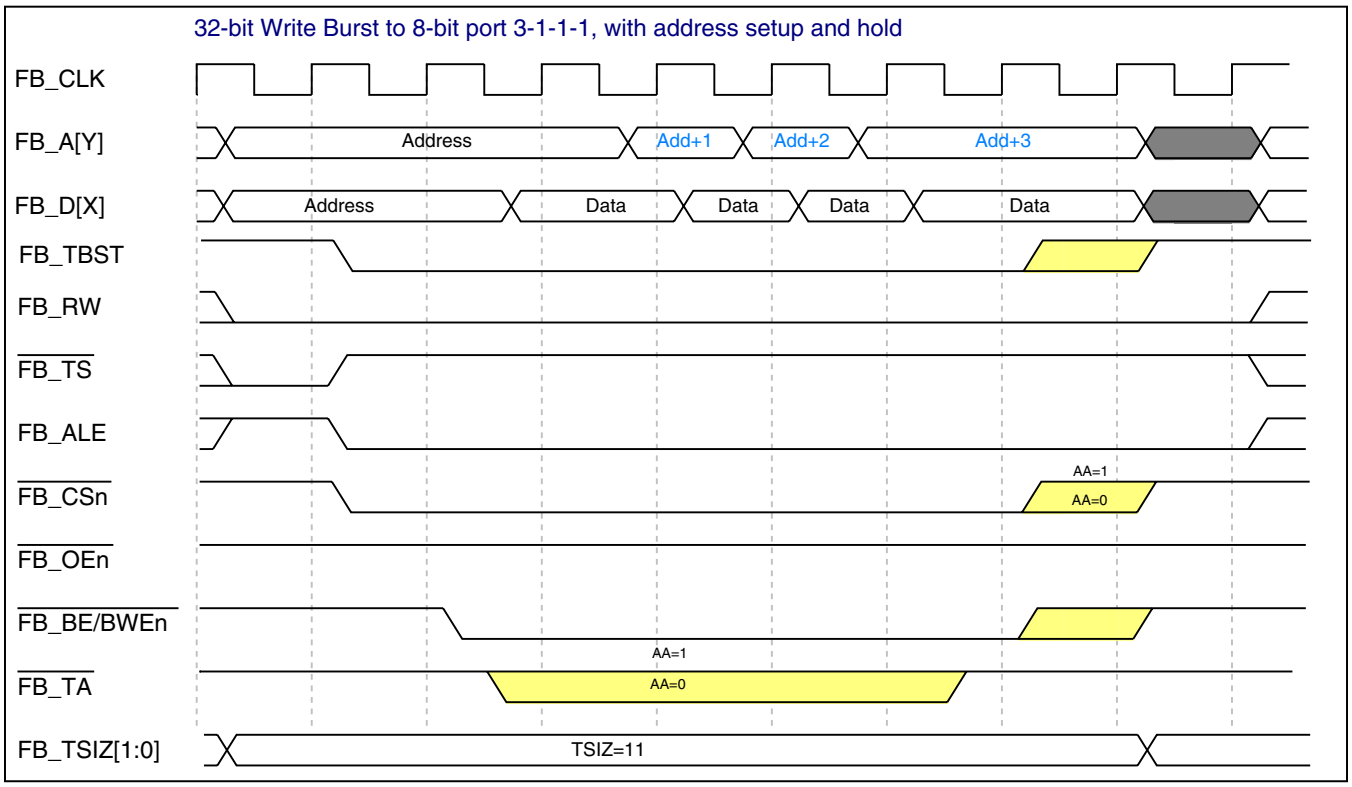
In non-multiplexed address/data mode, the address on FB_A increments only during internally-terminated burst cycles (CSCRn[AA] = 1b). The attached device must be able to account for this, or a wait state must be added. The first address is driven throughout the entire burst for externally-terminated cycles.

In multiplexed address/data mode, the address is driven on FB_AD only during the first cycle for internally- and externally-terminated cycles.




## 31.4.12.9 32-bit-write burst to 8-bit port 3-1-1-1 (address setup and hold)

The following figure shows a write cycle with one clock of address setup and address hold.




## 31.4.13 Extended Transfer Start/Address Latch Enable

The $\overline{\text{FB\_TS}}$/FB_ALE signal indicates that a bus transaction has begun and the address and attributes are valid. By default, the $\overline{\text{FB\_TS}}$/FB_ALE signal asserts for a single bus clock cycle. When CSCR*n*[EXTS] is set, the $\overline{\text{FB\_TS}}$/FB_ALE signal asserts and remain asserted until the first positive clock edge after $\overline{\text{FB\_CS}}$*n* asserts. See the following figure.

**NOTE**

When EXTS is set, CSCR*n*[WS] must be programmed to have at least one primary wait state.

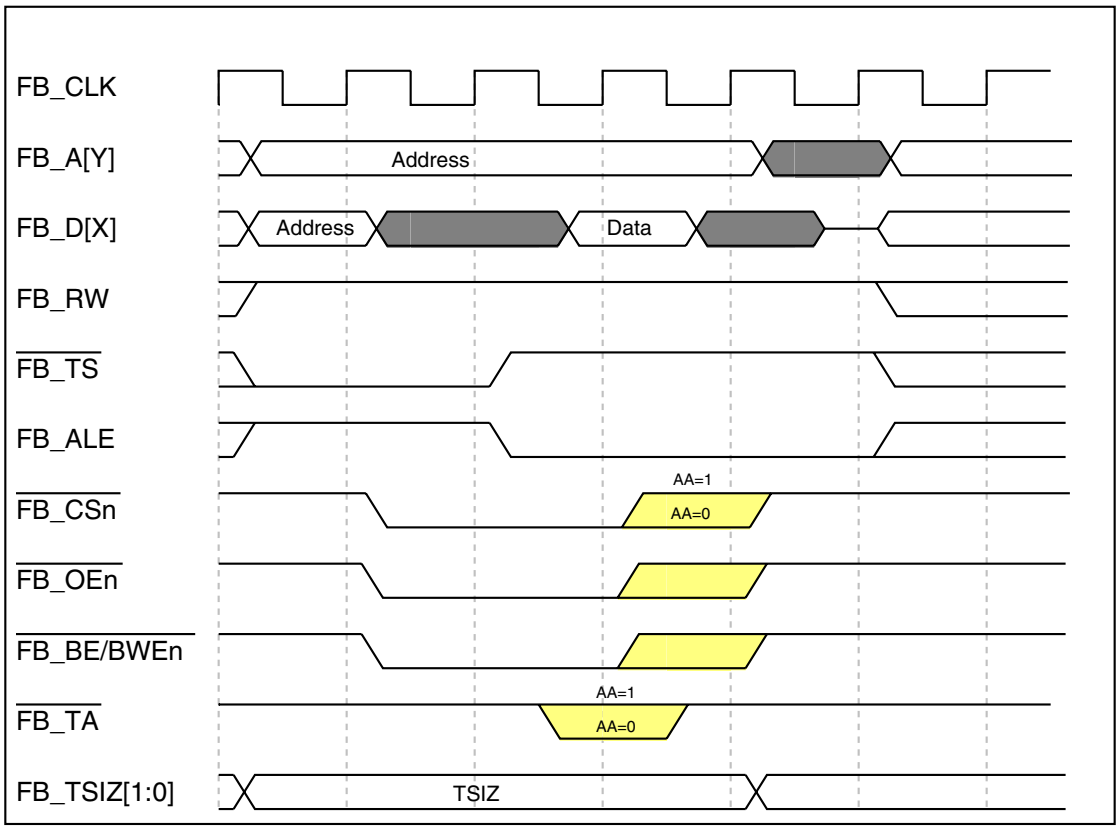



**Figure 31-44. Read-Bus Cycle with CSCR*n*[EXTS] = 1 (One Wait State)**

## 31.4.14 Bus errors

These types of accesses cause a transfer to terminate with a bus error:

- A write to a write-protected address range
- An access whose address is not in a range covered by a chip-select
- An access whose address is in a range covered by more than one chip-selects
- A write to a reserved address in the memory map
- A write to a reserved field in the CSPMCR
- Any FlexBus accesses when FlexBus is secure

If the auto-acknowledge feature is disabled (CSCR[AA] is 0) for an address that generates an error, the transfer can be terminated by asserting $\overline{\text{FB\_TA}}$. If the processor must manage a bus error differently, asserting an interrupt to the core along with $\overline{\text{FB\_TA}}$ when the bus error occurs can invoke an interrupt handler.

The device can hang if FlexBus is configured for external termination and the CSPMCR is not configured for $\overline{\text{FB\_TA}}$.

## 31.5 Initialization/Application Information

### 31.5.1 Initializing a chip-select

To initialize a chip-select:

1. Write to the associated CSAR.
2. Write to the associated CSCR.
3. Write to the associated CSMR, including writing 1b to the Valid field (CSMRn[V]).

### 31.5.2 Reconfiguring a chip-select

To reconfigure a previously-used chip-select:

1. Invalidate the chip-select by writing 0b to the associated CSMR's Valid field (CSMRn[V]).

2. Write to the associated CSAR.
3. Write to the associated CSCR.
4. Write to the associated CSMR, including writing 1b to the Valid field (CSMRn[V]).

# Chapter 32
# Cyclic Redundancy Check (CRC)

## 32.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The cyclic redundancy check (CRC) module generates 16/32-bit CRC code for error detection.

The CRC module provides a programmable polynomial, WAS, 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.

### 32.1.1 Features

Features of the CRC module include:

- 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

## 32.1.2 Block diagram

The following is a block diagram of the CRC.

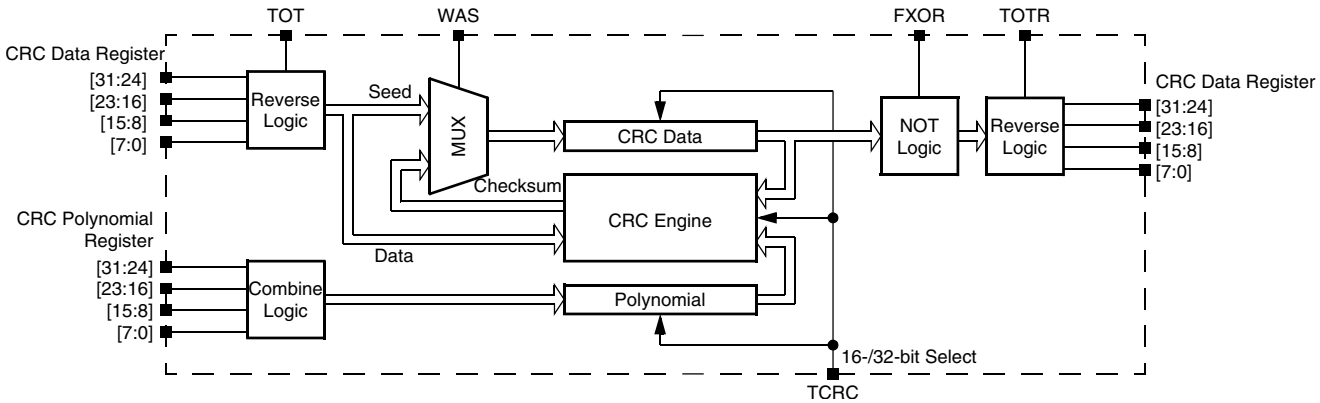



**Figure 32-1. Programmable cyclic redundancy check (CRC) block diagram**

## 32.1.3 Modes of operation

Various MCU modes affect the CRC module's functionality.

### 32.1.3.1 Run mode

This is the basic mode of operation.

### 32.1.3.2 Low-power modes (Wait or Stop)

Any CRC calculation in progress stops when the MCU 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 MCU dependent.

## 32.2 Memory map and register descriptions

**CRC memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4003_2000 | CRC Data register (CRC_DATA) | 32 | R/W | FFFF_FFFFh | 32.2.1/753 |
| 4003_2004 | CRC Polynomial register (CRC_GPOLY) | 32 | R/W | 0000_1021h | 32.2.2/754 |
| 4003_2008 | CRC Control register (CRC_CTRL) | 32 | R/W | 0000_0000h | 32.2.3/754 |

### 32.2.1 CRC Data register (CRC_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.

Address: 4003_2000h base + 0h offset = 4003_2000h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | HU | | | | | | | | HL | | | | | | | | LU | | | | | | | | LL | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

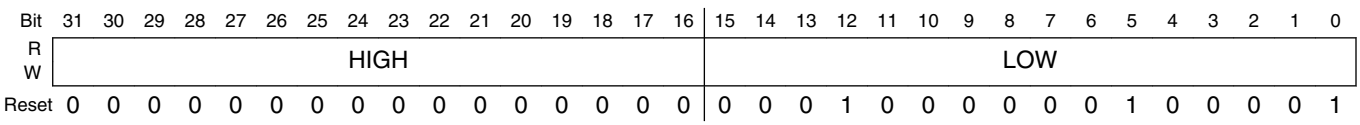
**CRC_DATA field descriptions**

| Field | Description |
|---|---|
| 31–24<br>HU | CRC High Upper Byte<br><br>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<br>HL | CRC High Lower Byte<br><br>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<br>LU | CRC Low Upper Byte<br><br>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<br>LL | CRC Low Lower Byte<br><br>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. |

## 32.2.2 CRC Polynomial register (CRC_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.

Address: 4003_2000h base + 4h offset = 4003_2004h



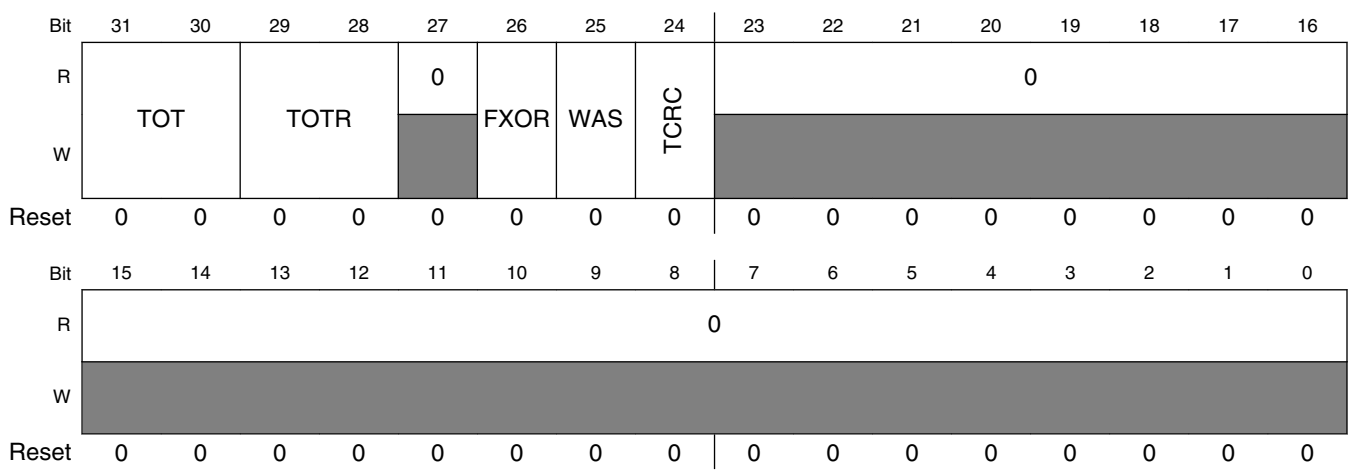

**CRC_GPOLY field descriptions**

| Field | Description |
|---|---|
| 31–16<br>HIGH | High Polynominal Half-word<br><br>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<br>LOW | Low Polynominal Half-word<br><br>Writable and readable in both 32-bit and 16-bit CRC modes. |

## 32.2.3 CRC Control register (CRC_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.

Address: 4003_2000h base + 8h offset = 4003_2008h



### CRC_CTRL field descriptions

| Field | Description |
|---|---|
| 31–30<br>TOT | Type Of Transpose For Writes<br><br>Define the transpose configuration of the data written to the CRC data register. See the description of the transpose feature for the available transpose options.<br><br>00 No transposition.<br>01 Bits in bytes are transposed; bytes are not transposed.<br>10 Both bits in bytes and bytes are transposed.<br>11 Only bytes are transposed; no bits in a byte are transposed. |
| 29–28<br>TOTR | Type Of Transpose For Read<br><br>Identify the transpose configuration of the value read from the CRC Data register. See the description of the transpose feature for the available transpose options.<br><br>00 No transposition.<br>01 Bits in bytes are transposed; bytes are not transposed.<br>10 Both bits in bytes and bytes are transposed.<br>11 Only bytes are transposed; no bits in a byte are transposed. |
| 27<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 26<br>FXOR | Complement Read Of CRC Data Register<br><br>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.<br><br>0 No XOR on reading.<br>1 Invert or complement the read value of the CRC Data register. |
| 25<br>WAS | Write CRC Data Register As Seed<br><br>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.<br><br>0 Writes to the CRC data register are data values.<br>1 Writes to the CRC data register are seed values. |
| 24<br>TCRC | Width of CRC protocol.<br><br>0 16-bit CRC protocol.<br>1 32-bit CRC protocol. |
| 23–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 32.3 Functional description

### 32.3.1 CRC initialization/reinitialization

To enable the CRC calculation, the user must program the WAS, POLYNOMIAL, and necessary parameters for transpose and CRC result inversion in the applicable registers. Asserting CTRL[WAS] enables the programming of the seed value into the CRC data register.

After a completed CRC calculation, reasserting CTRL[WAS] and programming a seed, whether the value is new or a previously used seed value, reinitialize the CRC module for a new CRC computation. All other parameters must be set before programming the seed value and subsequent data values.

### 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

To compute a 16-bit CRC:

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

To compute a 32-bit CRC:

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 00

No transposition occurs.

2. CTRL[TOT] or CTRL[TOTR] is 01

   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]}

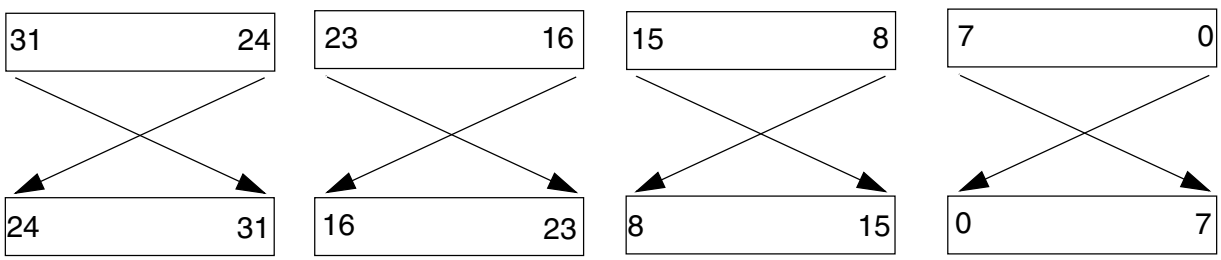



**Figure 32-5. Transpose type 01**

3. CTRL[TOT] or CTRL[TOTR] is 10

   Both bits in bytes and bytes are transposed.

   reg[31:0] becomes = {reg[0:7], reg[8:15],reg[16:23], reg[24:31]}

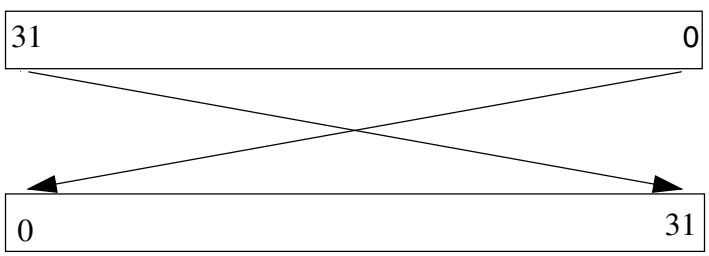



**Figure 32-6. Transpose type 10**

4. CTRL[TOT] or CTRL[TOTR] is 11

   Bytes are transposed, but bits are not transposed.

   reg[31:0] becomes {reg[7:0], reg[15:8], reg[23:16], reg[31:24]}

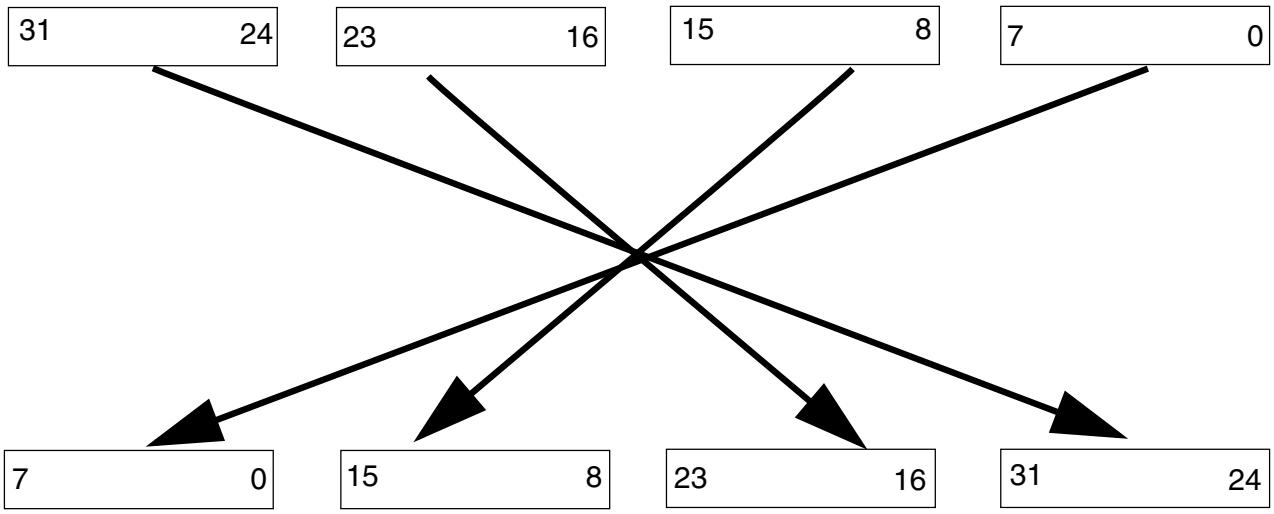



**Figure 32-7. Transpose type 11**

**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 register every time the CRC data register is read. When CTRL[FXOR] is cleared, reading the CRC data register accesses the raw checksum value.

# Chapter 33
# Memory-Mapped Cryptographic Acceleration Unit (MMCAU)

## 33.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The Memory-Mapped Cryptographic Acceleration Unit (MMCAU) is a coprocessor that is connected to the processor's Private Peripheral Bus (PPB). It supports the hardware implementation of a set of specialized operations to improve the throughput of software-based security encryption/decryption operations and message digest functions.

The MMCAU supports acceleration of the DES, 3DES, AES, MD5, SHA-1, and SHA-256 algorithms. Freecsale provides an optimized, callable C-function library that provides the appropriate software building blocks to implement higher-level security functions.

## 33.2 MMCAU Block Diagram

A simplified block diagram is given below that illustrates the MMCAU and a table to show its parts.

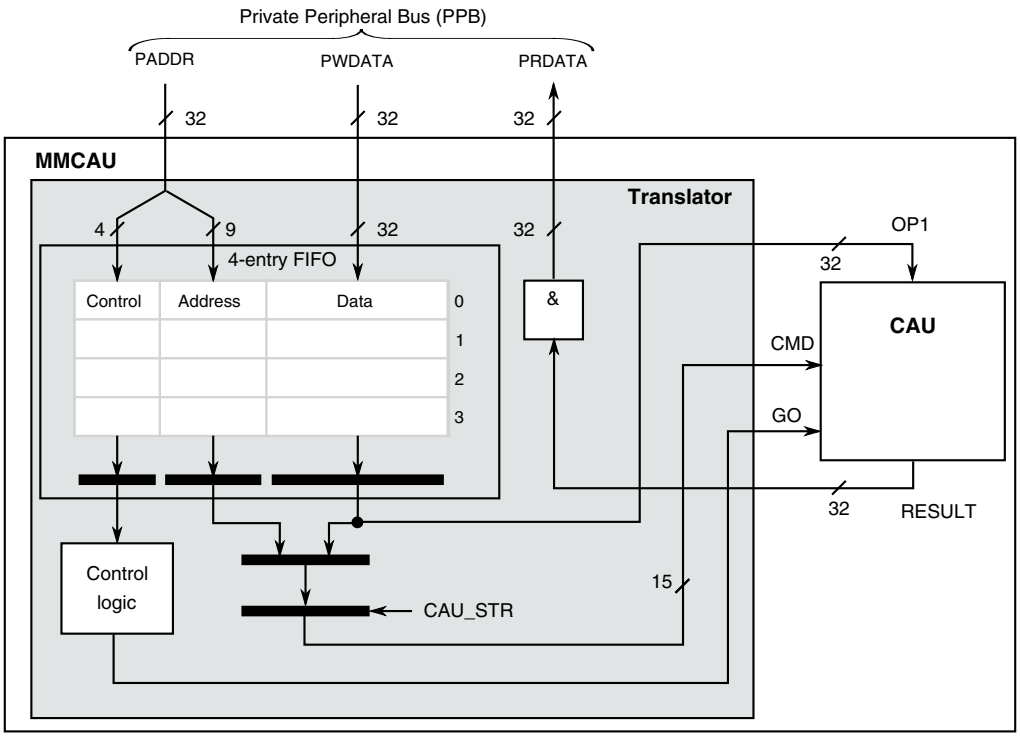



**Figure 33-1. MMCAU block diagram**

**Table 33-1. MMCAU parts table**

| Item | Description |
|---|---|
| Translator submodule | Provides the bridge between the private APB interface and the CAU module. Passes memory-mapped commands and data on the APB to/from the CAU |
| 4-entry FIFO | Contains commands and input operands and the associated control captured from the PPB and sent to the CAU |
| CAU | 3-terminal block with a command and optional input operand and a result bus. More details in following figure. |

The following figure shows the CAU block in more detail.

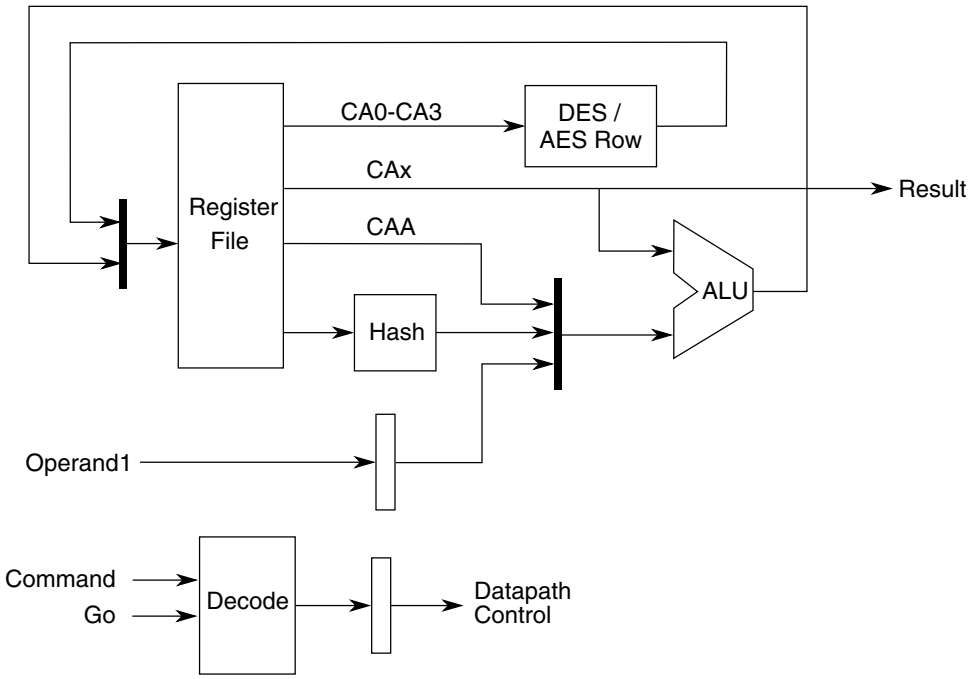



**Figure 33-2. Top-level CAU block diagram**

## 33.3 Overview

As the name suggests, the MMCAU provides a mechanism for memory-mapped register reads and writes to be transformed into specific commands and operands sent to the CAU coprocessor.

The MMCAU translator module performs the following functions:

- All the required functions affecting the transmission of commands to the CAU module.
- If needed, stalling the PPB transactions based on the state of the 4-entry command/ data FIFO.
- Some basic integrity checks on PPB operations.

The set of implemented algorithms provides excellent support for network security standards, such as SSL and IPsec. Additionally, using the MMCAU efficiently permits the implementation of any higher level functions or modes of operation, such as HMAC, CBC, and so on based on the supported algorithms.

The cryptographic algorithms are implemented partially in software with only functions critical to increasing performance implemented in hardware. The MMCAU allows for efficient, fine-grained partitioning of functions between hardware and software:

- Implement the innermost security kernel functions using the coprocessor instructions.
- Implement higher level functions in software by using the standard processor instructions.

This partitioning of functions is key to minimizing size of the MMCAU while maintaining a high level of throughput. Using software for some functions also simplifies the MMCAU design. The CAU implements a set of coprocessor commands that operate on a register file of 32-bit registers.

## 33.4 Features

The MMCAU includes the following distinctive features:

- Supports DES, 3DES, AES, MD5, SHA-1, and SHA-256 algorithms
- Simple, flexible programming model
- Ability to send up to three commands in one data write operation

## 33.5 Memory map/Register definition

The CAU contains multiple registers used by each of the supported algorithms. The following table shows registers that are applicable to each supported algorithm and indicates the corresponding letter designations for each algorithm. For more information on these letter designations, see the algorithm specifications.

| Code | Register | DES | AES | MD5 | SHA-1 | SHA-256 |
|---|---|---|---|---|---|---|
| 0 | CAU Status Register (CASR) | — | — | — | — | — |
| 1 | CAU Accumulator (CAA) | — | — | a | T | T |
| 2 | General-Purpose Register 0 (CA0) | C | W0 | — | A | A |
| 3 | General-Purpose Register 1 (CA1) | D | W1 | b | B | B |

*Table continues on the next page...*

| Code | Register | DES | AES | MD5 | SHA-1 | SHA-256 |
|---|---|---|---|---|---|---|
| 4 | General-Purpose Register 2 (CA2) | L | W2 | c | C | C |
| 5 | General-Purpose Register 3 (CA3) | R | W3 | d | D | D |
| 6 | General-Purpose Register 4 (CA4) | — | — | — | E | E |
| 7 | General-Purpose Register 5 (CA5) | — | — | — | W | F |
| 8 | General-Purpose Register 6 (CA6) | — | — | — | — | G |
| 9 | General-Purpose Register 7 (CA7) | — | — | — | — | H |
| 10 | General-Purpose Register 8 (CA8) | — | — | — | — | $W/T_1$ |

The CAU supports only 32-bit operations and register accesses. All registers support read, write, and ALU operations. However, only bits 1–0 of the CASR are writeable. Bits 31–2 of the CASR must be written as 0 for compatibility with future versions of the CAU.

The codes listed in this section are used in the memory-mapped commands. For more details on this, see MMCAU programming model.

**NOTE**

In the following table, the "address" or "offset" refers to the command code value for the CAU registers.

**CAU memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| E008_1000 | Status Register (CAU_CASR) | 32 | R/W | 2000_0000h | 33.5.1/766 |
| E008_1001 | Accumulator (CAU_CAA) | 32 | R/W | 0000_0000h | 33.5.2/767 |

*Table continues on the next page...*

**CAU memory map (continued)**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| E008_1002 | General Purpose Register (CAU_CA0) | 32 | R/W | 0000_0000h | 33.5.3/767 |
| E008_1003 | General Purpose Register (CAU_CA1) | 32 | R/W | 0000_0000h | 33.5.3/767 |
| E008_1004 | General Purpose Register (CAU_CA2) | 32 | R/W | 0000_0000h | 33.5.3/767 |
| E008_1005 | General Purpose Register (CAU_CA3) | 32 | R/W | 0000_0000h | 33.5.3/767 |
| E008_1006 | General Purpose Register (CAU_CA4) | 32 | R/W | 0000_0000h | 33.5.3/767 |
| E008_1007 | General Purpose Register (CAU_CA5) | 32 | R/W | 0000_0000h | 33.5.3/767 |
| E008_1008 | General Purpose Register (CAU_CA6) | 32 | R/W | 0000_0000h | 33.5.3/767 |
| E008_1009 | General Purpose Register (CAU_CA7) | 32 | R/W | 0000_0000h | 33.5.3/767 |
| E008_100A | General Purpose Register (CAU_CA8) | 32 | R/W | 0000_0000h | 33.5.3/767 |

## 33.5.1 Status Register (CAU_CASR)

CASR contains the status and configuration for the CAU.

Address: E008_1000h base + 0h offset = E008_1000h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VER | | | | 0 | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**CAU_CASR field descriptions**

| Field | Description |
|---|---|
| 31–28<br>VER | CAU Version<br><br>Indicates CAU version.<br><br>0x1 Initial CAU version.<br>0x2 Second version, added support for SHA-256 algorithm (This is the value on this device). |
| 27–2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 1<br>DPE | DES Parity Error<br><br>Indicates whether the DES parity error is detected.<br><br>0 No error detected.<br>1 DES key parity error detected. |

*Table continues on the next page...*

**CAU_CASR field descriptions (continued)**

| Field | Description |
|---|---|
| 0<br>IC | Illegal Command<br><br>Indicates an illegal instruction has been executed.<br><br>0 No illegal commands issued.<br>1 Illegal command issued. |

## 33.5.2 Accumulator (CAU_CAA)

Commands use the CAU accumulator for storage of results and as an operand for the cryptographic algorithms.

Address: E008_1000h base + 1h offset = E008_1001h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ACC | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CAU_CAA field descriptions**

| Field | Description |
|---|---|
| 31–0<br>ACC | Accumulator<br><br>Stores results of various CAU commands. |

## 33.5.3 General Purpose Register (CAU_CA*n*)

The General Purpose Register is used in the CAU commands for storage of results and as operands for various cryptographic algorithms.

Address: E008_1000h base + 2h offset + (1d × i), where i=0d to 8d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CAn | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CAU_CA*n* field descriptions**

| Field | Description |
|---|---|
| 31–0<br>CAn | General Purpose Registers<br><br>Used by the CAU commands. Some cryptographic operations work with specific registers. |

## 33.6 Functional description

This section discusses the programming model and operation of the MMCAU.

### 33.6.1 MMCAU programming model

The 4-entry FIFO is indirectly mapped into a 4-KB address space associated with the MMCAU located at byte addresses 0xE008_1000 – 0xE008_1FFF on this device. This address space is effectively split into two equal regions:

- one used to directly write commands for CAU load operations
- the other used to send commands and input operands for CAU loads

Data writes on the PPB are loaded into this FIFO and automatically converted into CAU load operands by the MMCAU translator. Data reads on the PPB are converted into CAU store register operations where the result is returned to the processor as the read data value.

The CAU requires a 15-bit command, and optionally, a 32-bit input operand, for each CAU load, PPB write. The 15-bit command includes the 9-bit opcode and other bits statically formed by the MMCAU translator logic controlling the CAU.

The following figure shows the 4-KB address space and the mapping of the CAU commands into this space.

**NOTE**

- Although the indirect store/load portion of the address space in the figure below shows only the indirect load/store commands, direct load commands can also be used in this space. However, it is more efficient to use the direct load portion of the address space.
- Accesses to the reserved space in the direct load space are terminated with an error, while accesses to the reserved space in the indirect load/store space are detected as an illegal CAU command. See MMCAU integrity checks for details.

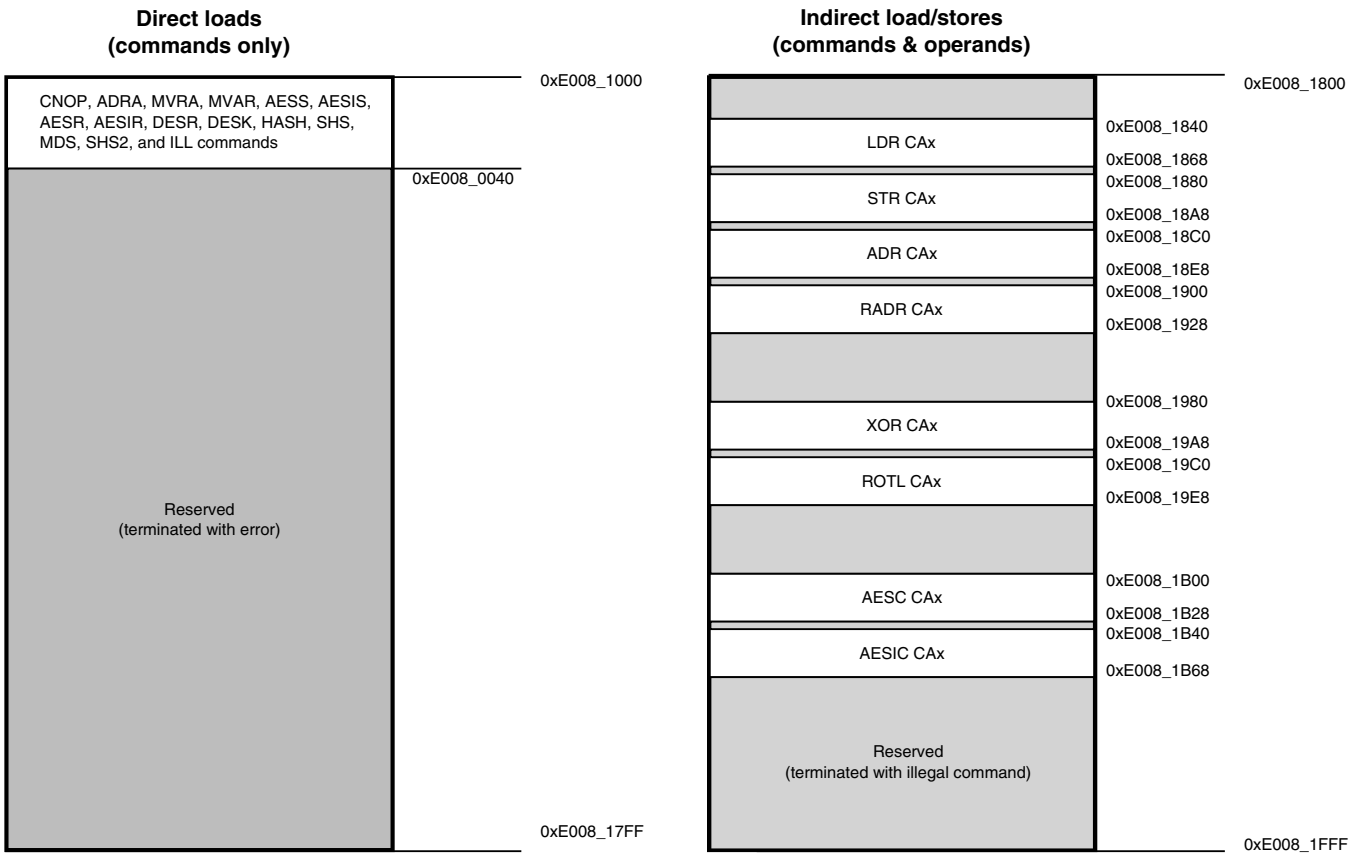



**Figure 33-15. MMCAU memory map**

### 33.6.1.1 Direct loads

The MMCAU supports writing multiple commands in each 32-bit direct write operation. Each 9-bit opcode also includes a valid bit. Therefore, one, two, or three commands can be transmitted in a single 32-bit PPB write. The following figure illustrates the accepted formats for the 32-bit MMCAU write data value:

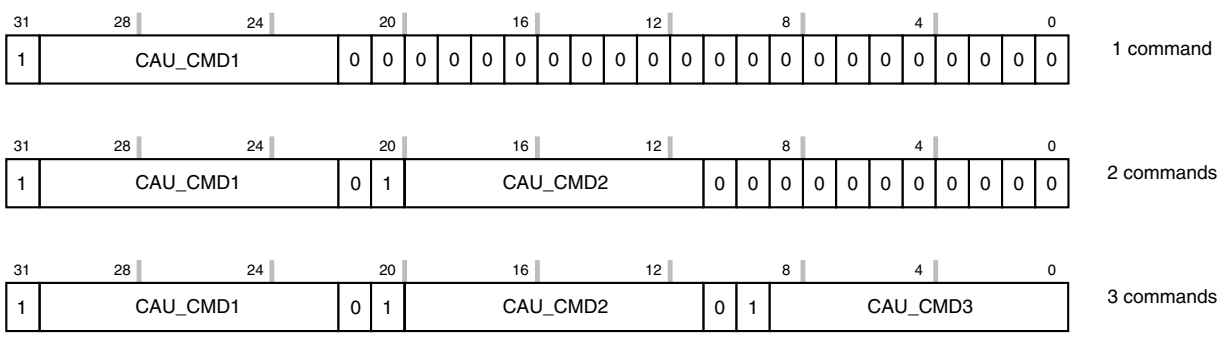



**Figure 33-16. Direct loads**

### 33.6.1.2 Indirect loads

For CAU load operations requiring a 32-bit input operand, the address contains the 9-bit opcode to be passed to the MMCAU while the data is the 32-bit operand. Specifically, the MMCAU address and data for these indirect writes is shown in the figure below.

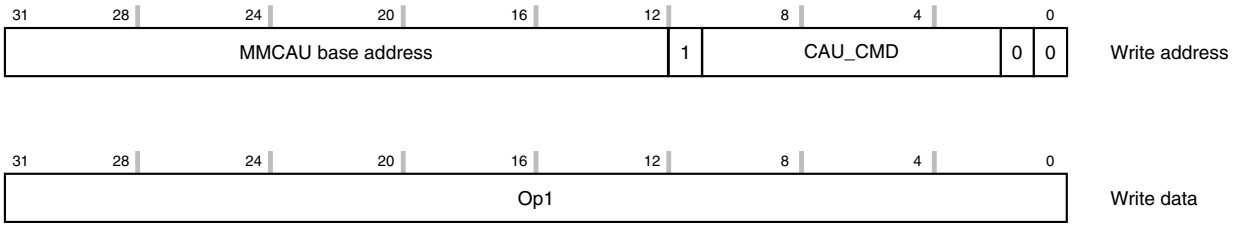



**Figure 33-17. Indirect loads**

### 33.6.1.3 Indirect stores

For CAU store operations, a PPB read is performed with the appropriate CAU store register opcode embedded in the address. This appears as another indirect command. The detail of Indirect stores is shown in the figure below.

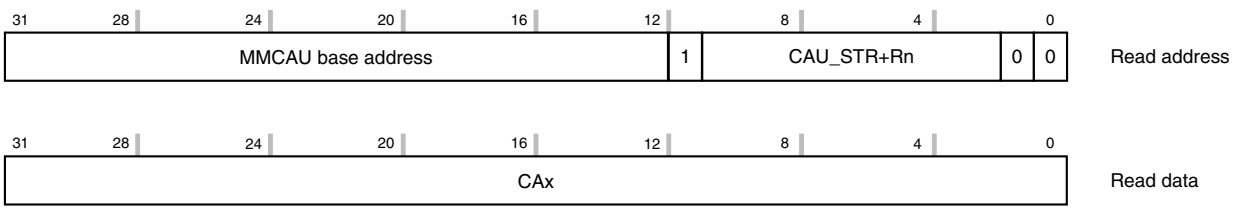



**Figure 33-18. Indirect store**

## 33.6.2 MMCAU integrity checks

If an illegal operation or access is attempted, the PPB bus cycle is terminated with an error response and the operation is aborted and not sent to the CAU.

The MMCAU performs a series of address and data integrity checks as described in the following sections. The results of these checks are logically summed together and, if appropriate, a PPB error termination is generated.

### 33.6.2.1 Address integrity checks

The MMCAU address checking includes the following. See Figure 33-15 for the MMCAU memory map details.

- Any MMCAU reference using a non-0-modulo-4 byte address (addr[1:0] ≠ 00) generates an error termination.
- For MMCAU writes:
  - Only the first 64 bytes of the 2-KB direct write address space can be referenced. Attempting to access regions beyond the first 64 bytes terminates with an error.
  - The second 2-KB space defines the indirect address-as-command region and any reference in this space is allowed by the MMCAU.

**NOTE**

The CAU contains error logic to detect any illegal command sent to it. Accordingly, there are address values in this upper 2-KB region of the address space that are passed to the CAU, and then detected as illegal commands. If the CAU detects an illegal command, it sets the CASR[IC] flag and performs no operation.

- For MMCAU reads:
  - Any attempted read from the first 2-KB region of the address space (an attempted direct read) is illegal and produces an error termination.
  - Within the second 2-KB region of the address space, i.e., addr[11] = 1, only a 64-byte space is treated as a legal CAU store operation. The allowable addresses are defined as:

    addr[11:0] = 1000_10xx_xx_00

    where the 4-bit xxxx value specifies the CAU register number. The CAU supports a subset of the allowable register numbers, 0x0 - 0xA. Attempting a store of a reserved register produces an undefined result.

### 33.6.2.2 Data integrity checks

Direct writes can send 1, 2, or 3 commands to the CAU in a single 32-bit transfer. As shown in Figure 33-16, the commands include a valid bit located at bits 31, 20, and 9 of the write data where:

- Bit 31 is the valid bit for the first command
- Bit 20 is the valid bit for the second command
- Bit 9 is the valid bit for the third command

The direct write data check validates the combination of these three valid bits. The following table presents the three legal states associated with these bits:

| Value of bits 31, 20, and 9 | Number of commands included |
|---|---|
| 100 | 1 |
| 110 | 2 |
| 111 | 3 |

All other combinations of bits 31, 20, and 9 are illegal and generate an error termination.

## 33.6.3 CAU commands

The CAU supports the commands shown in the following table. All other encodings are reserved. The CASR[IC] bit is set if an undefined command is issued. A specific illegal command (ILL) is defined to allow software self-checking. Reserved commands must not be issued so as to ensure compatibility with future implementations.

The CMD field specifies the 9-bit CAU opcode for the operation command.

See Assembler equate values for a set of assembly constants used in the command descriptions here. If supported by the assembler, macros can also be created for each instruction. The value CA*x* should be interpreted as any CAU register (CASR, CAA, and CA*n*).

**Table 33-15. CAU commands**

| Type | Command name | Description | CMD | | | | | | | | | Operation |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Direct load | CNOP | No Operation | 0x000 | | | | | | | | | — |
| Indirect load | LDR | Load Reg | 0x01 | | | | | CAx | | | | Op1 → CAx |
| Indirect store | STR | Store Reg | 0x02 | | | | | CAx | | | | CAx → Result |
| Indirect load | ADR | Add | 0x03 | | | | | CAx | | | | CAx + Op1 → CAx |
| Indirect load | RADR | Reverse and Add | 0x04 | | | | | CAx | | | | CAx + ByteRev(Op1) → CAx |
| Direct load | ADRA | Add Reg to Acc | 0x05 | | | | | CAx | | | | CAx + CAA → CAA |
| Indirect load | XOR | Exclusive Or | 0x06 | | | | | CAx | | | | CAx ^ Op1 → CAx |
| Indirect load | ROTL | Rotate Left | 0x07 | | | | | CAx | | | | (CAx <<< (Op1 % 32)) \| (CAx >>> (32 - (Op1 % 32))) → CAx |
| Direct load | MVRA | Move Reg to Acc | 0x08 | | | | | CAx | | | | CAx → CAA |
| Direct load | MVAR | Move Acc to Reg | 0x09 | | | | | CAx | | | | CAA → CAx |
| Direct load | AESS | AES Sub Bytes | 0x0A | | | | | CAx | | | | SubBytes(CAx) → CAx |
| Direct load | AESIS | AES Inv Sub Bytes | 0x0B | | | | | CAx | | | | InvSubBytes(CAx) → CAx |
| Indirect load | AESC | AES Column Op | 0x0C | | | | | CAx | | | | MixColumns(CAx)^Op1→ CAx |
| Indirect load | AESIC | AES Inv Column Op | 0x0D | | | | | CAx | | | | InvMixColumns(CAx^Op1) → CAx |

*Table continues on the next page...*

**Table 33-15. CAU commands (continued)**

| Type | Command name | Description | CMD | | | | | | | | | Operation |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Direct load | AESR | AES Shift Rows | 0x0E0 | | | | | | | | | ShiftRows(CA0-CA3) → CA0-CA3 |
| Direct load | AESIR | AES Inv Shift Rows | 0x0F0 | | | | | | | | | InvShiftRows(CA0-CA3)→ CA0-CA3 |
| Direct load | DESR | DES Round | 0x10 | | | | | IP | FP | KS[1:0] | | DES Round(CA0-CA3) →CA0-CA3 |
| Direct load | DESK | DES Key Setup | 0x11 | | | | | 0 | 0 | CP | DC | DES Key Op(CA0-CA1)→ CA0-CA1<br>Key Parity Error & CP → CASR[1] |
| Direct load | HASH | Hash Function | 0x12 | | | | | 0 | HF[2:0] | | | Hash Func(CA1-CA3)+CAA→ CAA |
| Direct load | SHS | Secure Hash Shift | 0x130 | | | | | | | | | CAA <<< 5→ CAA,<br>CAA→CA0, CA0→CA1,<br>CA1 <<< 30 → CA2,<br>CA2→CA3, CA3→CA4 |
| Direct load | MDS | Message Digest Shift | 0x140 | | | | | | | | | CA3→CAA, CAA→CA1,<br>CA1→CA2, CA2→CA3, |
| Direct load | SHS2 | Secure Hash Shift 2 | 0x150 | | | | | | | | | CAA→CA0, CA0→CA1,<br>CA1 → CA2, CA2→CA3,<br>CA3 + CA8 →CA4,<br>CA4 → CA5, CA5 → CA6,<br>CA6 → CA7 |
| Direct load | ILL | Illegal Command | 0x1F0 | | | | | | | | | 0x1→CASR[IC] |

### 33.6.3.1 Coprocessor No Operation (CNOP)

The CNOP command is the coprocessor no-op. It is issued by the MMCAU and consumes a location in the MMCAU FIFO, but has no effect on any CAU register.

### 33.6.3.2 Load Register (LDR)

The LDR command loads CAx with the source data specified by the write data.

### 33.6.3.3 Store Register (STR)

The STR command returns the value of CAx specified in the read address to the destination specified as read data.

### 33.6.3.4 Add to Register (ADR)

The ADR command adds the source operand specified by the write data to CAx and stores the result in CAx.

### 33.6.3.5 Reverse and Add to Register (RADR)

The RADR command performs a byte reverse on the source operand specified by the write data, adds that value to CAx, and stores the result in CAx. The table below shows an example.

**Table 33-16. RADR command example**

| Operand | CAx before | CAx after |
|---|---|---|
| 0x0102_0304 | 0xA0B0_C0D0 | 0xA4B3_C2D1 |

### 33.6.3.6 Add Register to Accumulator (ADRA)

The ADRA command adds CAx to CAA and stores the result in CAA.

### 33.6.3.7 Exclusive Or (XOR)

The XOR command performs an exclusive-or of the source operand specified by the write data with CAx and stores the result in CAx.

### 33.6.3.8 Rotate Left (ROTL)

ROTL rotates the CAx bits to the left with the result stored back to CAx. The number of bits to rotate is the value specified by the write data modulo 32.

#### 33.6.3.9 Move Register to Accumulator (MVRA)

The MVRA command moves the value from the source register CAx to the destination register CAA.

#### 33.6.3.10 Move Accumulator to Register (MVAR)

The MVAR command moves the value from source register CAA to the destination register CAx.

#### 33.6.3.11 AES Substitution (AESS)

The AESS command performs the AES byte substitution operation on CAx and stores the result back to CAx.

#### 33.6.3.12 AES Inverse Substitution (AESIS)

The AESIS command performs the AES inverse byte substitution operation on CAx and stores the result back to CAx.

#### 33.6.3.13 AES Column Operation (AESC)

The AESC command performs the AES column operation on the contents of CAx. It then performs an exclusive-or of that result with the source operand specified by the write data and stores the result in CAx.

#### 33.6.3.14 AES Inverse Column Operation (AESIC)

The AESIC command performs an exclusive-or operation of the source operand specified by the write data on the contents of CAx followed by the AES inverse mix column operation on that result and stores the result back in CAx.

#### 33.6.3.15 AES Shift Rows (AESR)

The AESR command performs the AES shift rows operation on registers CA0, CA1, CA2, and CA3. The table below shows an example.

**Table 33-17. AESR command example**

| Register | Before | After |
|---|---|---|
| CA0 | 0x0102_0304 | 0x0106_0B00 |
| CA1 | 0x0506_0708 | 0x050A_0F04 |
| CA2 | 0x090A_0B0C | 0x090E_0308 |
| CA3 | 0x0D0E_0F00 | 0x0D02_070C |

### 33.6.3.16 AES Inverse Shift Rows (AESIR)

The AESIR command performs the AES inverse shift rows operation on registers CA0, CA1, CA2, and CA3. The table below shows an example.

**Table 33-18. AESIR command example**

| Register | Before | After |
|---|---|---|
| CA0 | 0x0106_0B00 | 0x0102_0304 |
| CA1 | 0x050A_0F04 | 0x0506_0708 |
| CA2 | 0x090E_0308 | 0x090A_0B0C |
| CA3 | 0x0D02_070C | 0x0D0E_0F00 |

### 33.6.3.17 DES Round (DESR)

The DESR command performs a round of the DES algorithm and a key schedule update with the following source and destination designations: CA0=C, CA1=D, CA2=L, CA3=R. If the IP bit is set, DES initial permutation performs on CA2 and CA3 before the round operation. If the FP bit is set, DES final permutation, that is, inverse initial permutation, performs on CA2 and CA3 after the round operation. The round operation uses the source values from registers CA0 and CA1 for the key addition operation. The KSx field specifies the shift for the key schedule operation to update the values in CA0 and CA1. The following table defines the specific shift function performed based on the KSx field.

**Table 33-19. Key shift function codes**

| KSx code | KSx define | Shift function |
|---|---|---|
| 0 | KSL1 | Left 1 |
| 1 | KSL2 | Left 2 |
| 2 | KSR1 | Right 1 |
| 3 | KSR2 | Right 2 |

### 33.6.3.18 DES Key setup (DESK)

The DESK command performs the initial key transformation, permuted choice 1, defined by the DES algorithm on CA0 and CA1 with CA0 containing bits 1–32 of the key and CA1 containing bits 33–64 of the key[1] . If the DC bit is set, no shift operation performs and the values $C_0$ and $D_0$ store back to CA0 and CA1, respectively. The DC bit must be set for decrypt operations. If the DC bit is not set, a left shift by one also occurs and the values $C_1$ and $D_1$ store back to CA0 and CA1, respectively. The DC bit should be cleared for encrypt operations. If the CP bit is set and a key parity error is detected, CASR[DPE] bit is set; otherwise, it is cleared.

### 33.6.3.19 Hash Function (HASH)

The HASH command performs a hashing operation on a set of registers and adds that result to the value in CAA and stores the result in CAA. The specific hash function performed is based on the HFx field as defined in the table below.

This table uses the following terms:

- ROTR$^n$(CAx): rotate CAx register right *n* times
- SHR$^n$(CAx): shift CAx register right *n* times

**Table 33-20. Hash Function codes**

| HFx code | HFx define | Hash Function | Hash logic |
|---|---|---|---|
| 0 | HFF | MD5 F() | (CA1 & CA2) \| ($\overline{CA1}$ & CA3) |
| 1 | HFG | MD5 G() | (CA1 & CA3) \| (CA2 & $\overline{CA3}$) |
| 2 | HFH | MD5 H(), SHA Parity() | CA1 ^ CA2 ^ CA3 |
| 3 | HFI | MD5 I() | CA2 ^ (CA1 \| $\overline{CA3}$) |
| 4 | HFC | SHA Ch() | (CA1 & CA2) ^ ($\overline{CA1}$ & CA3) |
| 5 | HFM | SHA Maj() | (CA1 & CA2) ^ (CA1 & CA3) ^ (CA2 & CA3) |
| 6 | HF2C | SHA-256 Ch() | (CA4 & CA5) ^ ($\overline{CA1}$ & CA6) |
| 7 | HF2M | SHA-256 Maj() | (CA0 & CA1) ^ (CA0 & CA2) ^ (CA1 & CA2) |
| 8 | HF2S | SHA-256 Sigma 0 | ROTR$^2$(CA0) ^ ROTR$^{13}$(CA0) ^ ROTR$^{22}$(CA0) |
| 9 | HF2T | SHA-256 Sigma 1 | ROTR$^6$(CA4) ^ ROTR$^{11}$(CA4) ^ ROTR$^{25}$(CA4) |
| A | HF2U | SHA-256 Sigma 0 | ROTR$^7$(CA8) ^ ROTR$^{18}$(CA8) ^ SHR$^3$(CA8) |
| B | HF2V | SHA-256 Sigma 1 | ROTR$^{17}$(CA8) ^ ROTR$^{19}$(CA8) ^ SHR$^{10}$(CA8) |

1. The DES algorithm numbers the most significant bit of a block as bit 1 and the least significant as bit 64.

### 33.6.3.20 Secure Hash Shift (SHS)

The SHS command does a set of parallel register-to-register move and shift operations for implementing SHA-1. The following source and destination assignments are made:

| Register | Value prior to command | Value after command executes |
|---|---|---|
| CA4 | CA4 | CA3 |
| CA3 | CA3 | CA2 |
| CA2 | CA2 | CA1<<<30 |
| CA1 | CA1 | CA0 |
| CA0 | CA0 | CAA |
| CAA | CAA | CAA<<<5 |

### 33.6.3.21 Message Digest Shift (MDS)

The MDS command does a set of parallel register-to-register move operations for implementing MD5. The following source and destination assignments are made:

| Register | Value prior to command | Value after command executes |
|---|---|---|
| CA3 | CA3 | CA2 |
| CA2 | CA2 | CA1 |
| CA1 | CA1 | CAA |
| CAA | CAA | CA3 |

### 33.6.3.22 Secure Hash Shift 2 (SHS2)

The SHS2 command does an addition and a set of register to register moves in parallel for implementing SHA-256. The following source and destination assignments are made:

| Register | Value prior to command | Value after command executes |
|---|---|---|
| CA7 | CA7 | CA6 |
| CA6 | CA6 | CA5 |
| CA5 | CA5 | CA4 |
| CA4 | CA4 | CA3+CA8 |
| CA3 | CA3 | CA2 |
| CA2 | CA2 | CA1 |
| CA1 | CA1 | CA0 |
| CA0 | CA0 | CAA |

### 33.6.3.23 Illegal command (ILL)

The ILL command is a specific illegal command that sets CASR[IC]. All other illegal commands are reserved for use in future implementations.

## 33.7 Application/initialization information

This section discusses how to initialize and use the MMCAU.

### 33.7.1 Code example

A code fragment is shown below as an example of how the MMCAU is used. This example shows the round function of the AES algorithm. Core registers are defined as follows:

- R1 points to the key schedule
- R3 contains three direct MMCAU commands
- R8 contains two direct MMCAU commands
- R9 contains an indirect MMCAU command
- FP points to the MMCAU indirect command address space
- IP points to the MMCAU direct command space

```
    movw    fp, #:lower16:MMCAU_PPB_INDIRECT              @ fp -> MMCAU_PPB_INDIRECT
    movt    fp, #:upper16:MMCAU_PPB_INDIRECT
    movw    ip, #:lower16:MMCAU_PPB_DIRECT                @ ip -> MMCAU_PPB_DIRECT
    movt    ip, #:upper16:MMCAU_PPB_DIRECT

# r3 = mmcau_3_cmds(AESS+CA0,AESS+CA1,AESS+CA2)
    movw    r3, #:lower16:(0x80100200+(AESS+CA0)<<22+(AESS+CA1)<<11+AESS+CA2)
    movt    r3, #:upper16:(0x80100200+(AESS+CA0)<<22+(AESS+CA1)<<11+AESS+CA2)

# r8 = mmcau_2_cmds(AESS+CA3,AESR)
    movw    r8, #:lower16:(0x80100000+(AESS+CA3)<<22+(AESR)<<11)
    movt    r8, #:upper16:(0x80100000+(AESS+CA3)<<22+(AESR)<<11)

    add     r9, fp, $((AESC+CA0)<<2)                        @ r9 = mmcau_cmd(AESC+CA0)

    str     r3, [ip]                                      @ sub bytes w0, w1, w2
    str     r8, [ip]                                      @ sub bytes w3, shift rows
    ldmia   r1!, {r4-r7}                                  @ get next 4 keys; r1++
    stmia   r9, {r4-r7}                                   @ mix columns, add keys
```

### 33.7.2 Assembler equate values

The following equates ease programming of the MMCAU.

```
; CAU Registers (CAx)
```

```
      .set CASR,0x0
      .set CAA,0x1
      .set CA0,0x2
      .set CA1,0x3
      .set CA2,0x4
      .set CA3,0x5
      .set CA4,0x6
      .set CA5,0x7
      .set CA6,0x8
      .set CA7,0x9
      .set CA8,0xA
; CAU Commands
      .set CNOP,0x000
      .set LDR,0x010
      .set STR,0x020
      .set ADR,0x030
      .set RADR,0x040
      .set ADRA,0x050
      .set XOR,0x060
      .set ROTL,0x070
      .set MVRA,0x080
      .set MVAR,0x090
      .set AESS,0x0A0
      .set AESIS,0x0B0
      .set AESC,0x0C0
      .set AESIC,0x0D0
      .set AESR,0x0E0
      .set AESIR,0x0F0
      .set DESR,0x100
      .set DESK,0x110
      .set HASH,0x120
      .set SHS,0x130
      .set MDS,0x140
      .set SHS2,0x150
      .set ILL,0x1F0
; DESR  Fields
      .set IP,0x08          ; initial permutation
      .set FP,0x04          ; final permutation
      .set KSL1,0x00        ; key schedule left 1 bit
      .set KSL2,0x01        ; key schedule left 2 bits
      .set KSR1,0x02        ; key schedule right 1 bit
      .set KSR2,0x03        ; key schedule right 2 bits
; DESK Field
      .set DC,0x01          ; decrypt key schedule
      .set CP,0x02          ; check parity
; HASH Functions Codes
      .set HFF,0x0          ; MD5 F() CA1&CA2 | ~CA1&CA3
      .set HFG,0x1          ; MD5 G() CA1&CA3 | CA2&~CA3
      .set HFH,0x2          ; MD5 H(), SHA Parity() CA1^CA2^CA3
      .set HFI,0x3          ; MD5 I()  CA2^(CA1|~CA3)
      .set HFC,0x4          ; SHA Ch() CA1&CA2 ^ ~CA1&CA3
      .set HFM,0x5          ; SHA Maj() CA1&CA2 ^ CA1&CA3 ^ CA2&CA3
      .set HF2C,0x6         ; SHA-256 Ch() CA4&CA5 ^ ~CA4&CA6
      .set HF2M,0x7         ; SHA-256 Maj() CA0&CA1 ^ CA0&CA2 ^ CA1&CA2
      .set HF2S,0x8         ; SHA-256 Sigma 0 ROTR2(CA0)^ROTR13(CA0)^ROTR22(CA0)
      .set HF2T,0x9         ; SHA-256 Sigma 1 ROTR6(CA4)^ROTR11(CA4)^ROTR25(CA4)
      .set HF2U,0xA         ; SHA-256 sigma 0 ROTR7(CA8)^ROTR18(CA8)^SHR3(CA8)
      .set HF2V,0xB         ; SHA-256 sigma 1 ROTR17(CA8)^ROTR19(CA8)^SHR10(CA8)
```

# Chapter 34
# Random Number Generator Accelerator (RNGA)

## 34.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

This chapter describes the Random Number Generator Accelerator (RNGA), including a programming model, functional description, and application information.

### 34.1.1 Overview

The RNGA module is a digital integrated circuit capable of generating 32-bit random numbers. The random bits are generated by clocking shift registers with clocks derived from ring oscillators. The configuration of the shift registers ensures statistically good data, that is, data that looks random. The oscillators with their unknown frequencies provide the required entropy needed to create random data.

It is important to note that there is no known cryptographic proof showing that this is a secure method of generating random data. In fact, there may be an attack against the random number generator described in this document if its output is used directly in a cryptographic application (the attack is based on the linearity of the internal shift registers). In light of this, it is highly recommended that the random data produced by this module be used as an input seed to a NIST approved (based on DES or SHA-1) Pseudo Random Number Generator as defined in *NIST FIPS PUB 186-2 Appendix 3* and *NIST FIPS PUB SP 800-90*. Other sources of entropy be used along with the RNGA to generate the seed to the pseudorandom algorithm. The more random sources combined to create the seed, the better. The following is a list of sources that could be easily combined with the output of this module:

- Current time using highest precision possible.

- Mouse and keyboard motions, or equivalent if being used on a cell phone or PDA.
- Other entropy supplied directly by the user.

## 34.2 Modes of operation

Although the RNGA has several modes, only one is intended for use during normal operation. The Sleep mode is entered by setting the appropriate bit in the RNGA Control Register.

- Normal mode

  In this mode, the RNGA generates random data. Because this is the default mode of operation, the user is not required to change the mode before requesting random data.

- Sleep mode

  In this mode the RNGA's oscillator clocks are shut off. The mode is entered by writing to the Sleep (SLP) bit in the RNGA Control Register. When in this mode, the RNGA Output Register will not be loaded.

## 34.3 Memory map and register definition

This section describes the RNGA registers.

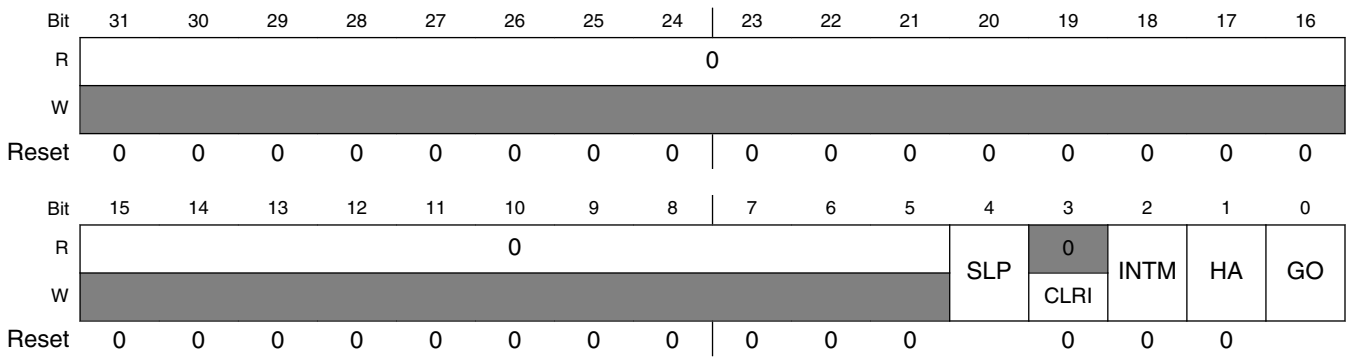
**RNG memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4002_9000 | RNGA Control Register (RNG_CR) | 32 | R/W | 0000_0000h | 34.3.1/783 |
| 4002_9004 | RNGA Status Register (RNG_SR) | 32 | R | 0001_0000h | 34.3.2/784 |
| 4002_9008 | RNGA Entropy Register (RNG_ER) | 32 | W (always reads 0) | 0000_0000h | 34.3.3/786 |
| 4002_900C | RNGA Output Register (RNG_OR) | 32 | R | 0000_0000h | 34.3.4/787 |

## 34.3.1 RNGA Control Register (RNG_CR)

Immediately after reset, the RNGA begins generating entropy in its internal shift registers. Random data is not pushed to the RNGA Output Register until the GO bit in the RNGA Control register is set. After this, a random 32-bit word is pushed to the RNGA Output Register every 256 system clock cycles. If the RNGA Output Register is full, then no push will occur. In this way, the RNGA Output Register is kept as close to full as possible.

Address: 4002_9000h base + 0h offset = 4002_9000h




**RNG_CR field descriptions**

| Field | Description |
|---|---|
| 31–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>SLP | Sleep<br>The RNGA can be placed in low-power mode by either asserting the module input, or by setting this bit. If either of these conditions are met, the oscillators are disabled. Clearing this bit causes the RNGA to exit Sleep mode. The RNGA Output Register is not pushed while the RNGA is in Sleep mode.<br>0 RNGA is not in Sleep mode.<br>1 RNGA is in Sleep mode. |
| 3<br>CLRI | Clear Interrupt<br>Writing a one to this bit clears the error interrupt as well as the error status bit (ERRI) in the RNGA Status Register. The bit is self clearing.<br>0 Do not clear the interrupt.<br>1 Clear the interrupt. |
| 2<br>INTM | Interrupt Mask<br>This bit masks the error interrupt.<br>0 Interrupt is enabled.<br>1 Interrupt is masked. |

*Table continues on the next page...*

**RNG_CR field descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>HA | High Assurance<br><br>While this bit is high, a read of the RNGA Output Register while empty causes a security violation. This bit enables security violation bit (SECV) in the RNGA Status Register. This bit is sticky and can only be cleared through a hardware reset.<br><br>0 Notification of security violations is enabled.<br>1 Notification of security violations is masked. |
| 0<br>GO | This bit must be set before the RNGA begins loading data into the RNGA Output Register. This bit is sticky and can only be cleared by a hardware reset . Setting this bit does not bring the RNGA out of Sleep mode. Furthermore, this bit does not need to be reset after exiting Sleep mode.<br><br>0 RNGA Output Register is not loaded with random data.<br>1 RNGA Output Register is loaded with random data. |

## 34.3.2 RNGA Status Register (RNG_SR)

The RNGA Status Register is a read only register that reflects the internal status of the RNGA.

Address: 4002_9000h base + 4h offset = 4002_9004h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | OREG_SIZE | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OREG_LVL | | | | | | | | 0 | | | SLP | ERRI | ORU | LRS | SECV |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 |

## RNG_SR field descriptions

| Field | Description |
|---|---|
| 31–24<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 23–16<br>OREG_SIZE | Output Register Size<br><br>This bit signals the actual size of the RNGA Output Register. In other words, this is the maximum possible RNGA Output Register Level. The bits should be interpreted as an integer. This value is set to 0x01. |
| 15–8<br>OREG_LVL | Output Register Level<br><br>This bit signals how many random words are currently resident in the RNGA Output Register. Only two values are possible. The bits should be interpreted as an integer (the value 0b00000001 indicates that 0x01 random word is in the RNGA Output Register, while the value 0b00000000 indicates that no random data is in the RNGA Output Register). |
| 7–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>SLP | Sleep<br><br>This bit reflects whether the RNGA is in Sleep mode that is, either the Sleep bit (SLP) in the RNGA Control Register is set or the “ipg_doze” input is asserted. When this bit is set, the RNGA is in Sleep mode and the oscillator clocks are inactive. While in this mode, the RNGA Output Register will not be loaded and the RNGA Output Register Level does not increase.<br><br>0 The RNGA is not in Sleep mode.<br>1 The RNGA is in Sleep mode. |
| 3<br>ERRI | Error Interrupt<br><br>This bit is always enabled and signals an RNGA Output Register underflow condition. This bit is different from the previous two bits in that it is only reset by a write to the clear interrupt bit (CLRI) in the RNGA Control Register. This bit is not masked by the Interrupt Mask bit of the RNGA Control Register.<br><br>0 The RNGA Output Register has not been read while empty.<br>1 The RNGA Output Register has been read while empty. |
| 2<br>ORU | Output Register Underflow<br><br>This bit is always enabled and signals an RNGA Output Register underflow condition. The bit is reset by reading the RNGA Status Register.<br><br>0 The RNGA Output Register has not been read while empty since last read of the RNGA Status Register.<br>1 The RNGA Output Register has been read while empty since last read of the RNGA Status Register. |
| 1<br>LRS | Last Read Status<br><br>This bit is always enabled and reflects the status of the most recent read of the RNGA Output Register.<br><br>0 Last read was performed while the RNGA Output Register was not empty.<br>1 Last read was performed while the RNGA Output Register was empty (underflow condition). |
| 0<br>SECV | Security Violation<br><br>When enabled by the High Assurance bit (HA) in the RNGA Control Register, this bit signals that a security violation has occurred. Currently, RNGA Output Register underflow is the only condition that is considered to be a security violation. The bit is sticky and can be cleared only by a hardware reset. |

*Table continues on the next page...*

**RNG_SR field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 No security violations have occured or the High Assurance bit (HA) in the RNGA Control Register is not set.<br>1 A security violation has occurred. |

## 34.3.3 RNGA Entropy Register (RNG_ER)

The RNGA Entropy Register is a write-only register that allows the user to insert entropy into the RNGA. This register allows an external user to continually seed the RNGA with externally generated random data. Although the use of this register is recommended, it is also optional. The RNGA Entropy Register can be written at any time during operation.

Address: 4002_9000h base + 8h offset = 4002_9008h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | EXT_ENT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RNG_ER field descriptions**

| Field | Description |
|---|---|
| 31–0<br>EXT_ENT | External Entropy<br><br>A write to this register allows the user to introduce 32 bits of entropy to the internal state of RNGA. Eight writes are required to introduce 256 bits of external entropy. |

### 34.3.4 RNGA Output Register (RNG_OR)

The RNGA Output Register provides temporary storage for random data generated by the RNGA. As long as the RNGA Output Register is not empty, a read of this address will return 32 bits of random data. If the RNGA Output Register is read when it is empty, Error Interrupt (ERRI), Output Register Underflow (ORU), and Last Read Staus (LRS) bits in the RNGA Status Register are set. If the interrupt is enabled in the RNGA Control Register, an interrupt is triggered to the interrupt controller The RNGA Output Register Level field in the RNGA Status Register can be polled to monitor how many 32-bit words currently resides in the RNGA Output Register. When in Normal mode, a new random word is pushed into the RNGA Output Register every 256 system clock cycles (as long as the RNGA Output Register is not full). Polling the RNGA Status Register is very important to make sure random values are present before reading from the RNGA Output Register.

Address: 4002_9000h base + Ch offset = 4002_900Ch

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RANDOUT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RNG_OR field descriptions**

| Field | Description |
|---|---|
| 31–0<br>RANDOUT | Random Output<br><br>32-bits of Random Data |

## 34.4 Functional description

The following figure shows the RNGA has three functional blocks: output FIFO, internal bus interface, and the RNGA core/control logic blocks. The following sections describe these blocks in more detail.

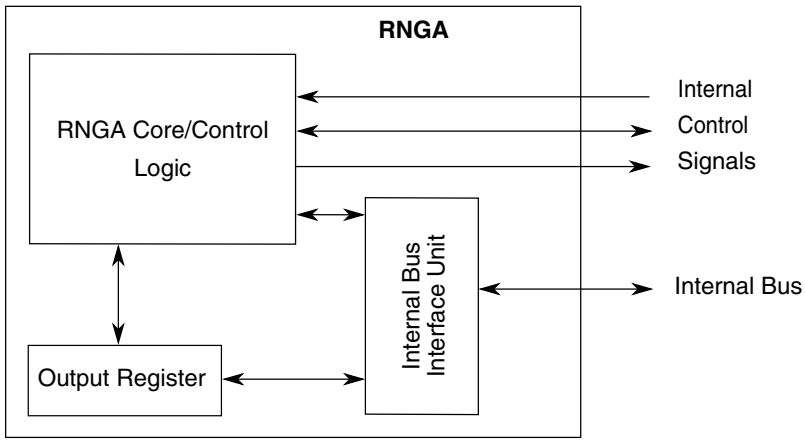



**Figure 34-5. RNGA block diagram**

## 34.4.1 RNGA Output Register

The RNGA Output Register provides temporary storage for random data generated by the RNGA Core/Control Logic block. The RNGA status register allows the user to monitor the number of random words in the RNGA Output Register through the RNGA Output Register Level field. If the user reads from the RNGA Output Register when it is empty and the interrupt is enabled, the RNGA drives an interrupt request to the interrupt controller . Polling the RNGA Status Register is very important to make sure random values are present before reading from the RNGA Output Register.

## 34.4.2 RNGA Core/Control Logic Block

This block contains the RNGA's Control Logic as well as its Core Engine used to generate random data.

### 34.4.2.1 RNGA Control Block

The Control Block contains the address decoder, all addressable registers, and control state machines for the RNGA. This block is responsible for communication with both the peripheral interface and the RNGA Output Register interface. The block also controls the Core Engine to generate random data. The general functionality of the block is as follows. After reset, entropy is generated and stored in the RNGA's shift registers. After the GO bit is set in the RNGA Control Register, the RNGA Output Register is loaded with a random word every 256 system clock cycles. The process of loading the RNGA Output Register continues as long as the RNGA Output Register is not full.

#### 34.4.2.2 Core Engine

The Core Engine Block contains the logic used to generate random data. The logic within the Core Engine contains the internal shift registers as well as the logic used to generate the two oscillator-based clocks. The Control Block controls how the shift registers are configured as well as when the oscillator clocks are turned on.

## 34.5 Initialization/application information

The intended general operation of the RNGA is as follows:

1. Reset/initialize
2. Write to the RNGA Control Register and set the Interrupt Mask (INTM), High Assurance (HA), and GO bits.
3. Poll the RNGA Status Register for RNGA Output Register Level
4. Read available random data from the RNGA Output Register
5. Repeat steps 3 and 4 as needed

# Chapter 35
# Analog-to-Digital Converter (ADC)

## 35.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The 16-bit analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip.

**NOTE**

For the chip specific modes of operation, see the power management information of the device.

### 35.1.1 Features

Features of the ADC module include:

- Linear successive approximation algorithm with up to 16-bit resolution
- Up to four pairs of differential and 24 single-ended external analog inputs
- Output modes:
  - differential 16-bit, 13-bit, 11-bit, and 9-bit modes
  - single-ended 16-bit, 12-bit, 10-bit, and 8-bit modes
- Output format in 2's complement 16-bit sign extended for differential modes
- Output in right-justified unsigned format for single-ended
- Single or continuous conversion, that is, automatic return to idle after single conversion

- Configurable sample time and conversion speed/power
- Conversion complete/hardware average complete flag and interrupt
- Input clock selectable from up to four sources
- Operation in Low-Power modes for lower noise
- Asynchronous clock source for lower noise operation with option to output the clock
- Selectable hardware conversion trigger with hardware channel select
- Automatic compare with interrupt for less-than, greater-than or equal-to, within range, or out-of-range, programmable value
- Temperature sensor
- Hardware average function
- Selectable voltage reference: external or alternate
- Self-Calibration mode

### 35.1.2 Block diagram

The following figure is the ADC module block diagram.

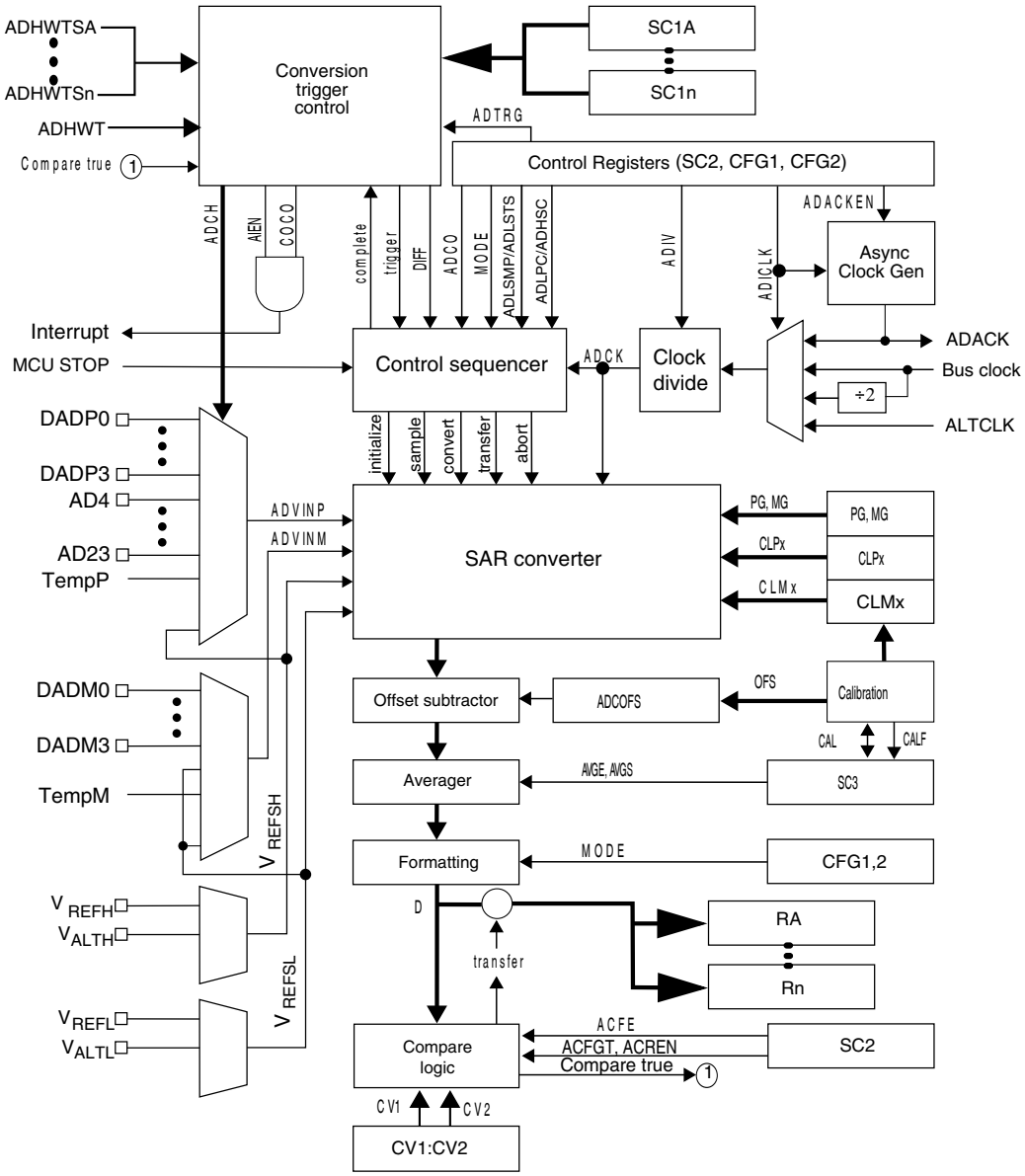



**Figure 35-1. ADC block diagram**

## 35.2 ADC Signal Descriptions

The ADC module supports up to 4 pairs of differential inputs and up to 24 single-ended inputs. Each differential pair requires two inputs, DADPx and DADMx. The ADC also requires four supply/reference/ground connections.

**NOTE**

Refer to ADC configuration section in chip configuration chapter for the number of channels supported on this device.

**Table 35-1. ADC Signal Descriptions**

| Signal | Description | I/O |
|---|---|---|
| DADP3–DADP0 | Differential Analog Channel Inputs | I |
| DADM3–DADM0 | Differential Analog Channel Inputs | I |
| AD*n* | Single-Ended Analog Channel Inputs | I |
| $V_{REFSH}$ | Voltage Reference Select High | I |
| $V_{REFSL}$ | Voltage Reference Select Low | I |
| $V_{DDA}$ | Analog Power Supply | I |
| $V_{SSA}$ | Analog Ground | I |

## 35.2.1 Analog Power ($V_{DDA}$)

The ADC analog portion uses $V_{DDA}$ as its power connection. In some packages, $V_{DDA}$ is connected internally to $V_{DD}$. If externally available, connect the $V_{DDA}$ pin to the same voltage potential as $V_{DD}$. External filtering may be necessary to ensure clean $V_{DDA}$ for good results.

## 35.2.2 Analog Ground ($V_{SSA}$)

The ADC analog portion uses $V_{SSA}$ as its ground connection. In some packages, $V_{SSA}$ is connected internally to $V_{SS}$. If externally available, connect the $V_{SSA}$ pin to the same voltage potential as $V_{SS}$.

## 35.2.3 Voltage Reference Select

$V_{REFSH}$ and $V_{REFSL}$ are the high and low reference voltages for the ADC module.

The ADC can be configured to accept one of two voltage reference pairs for $V_{REFSH}$ and $V_{REFSL}$. Each pair contains a positive reference that must be between the minimum Ref Voltage High and $V_{DDA}$, and a ground reference that must be at the same potential as $V_{SSA}$. The two pairs are external ($V_{REFH}$ and $V_{REFL}$) and alternate ($V_{ALTH}$ and $V_{ALTL}$). These voltage references are selected using SC2[REFSEL]. The alternate $V_{ALTH}$ and $V_{ALTL}$ voltage reference pair may select additional external pins or internal sources depending on MCU configuration. See the chip configuration information on the Voltage References specific to this MCU.

In some packages, $V_{REFH}$ is connected in the package to $V_{DDA}$ and $V_{REFL}$ to $V_{SSA}$. If externally available, the positive reference(s) may be connected to the same potential as $V_{DDA}$ or may be driven by an external source to a level between the minimum Ref Voltage High and the $V_{DDA}$ potential. $V_{REFH}$ must never exceed $V_{DDA}$. Connect the ground references to the same voltage potential as $V_{SSA}$.

### 35.2.4 Analog Channel Inputs (ADx)

The ADC module supports up to 24 single-ended analog inputs. A single-ended input is selected for conversion through the SC1[ADCH] channel select bits when SC1n[DIFF] is low.

### 35.2.5 Differential Analog Channel Inputs (DADx)

The ADC module supports up to four differential analog channel inputs. Each differential analog input is a pair of external pins, DADPx and DADMx, referenced to each other to provide the most accurate analog to digital readings. A differential input is selected for conversion through SC1[ADCH] when SC1n[DIFF] is high. All DADPx inputs may be used as single-ended inputs if SC1n[DIFF] is low. In certain MCU configurations, some DADMx inputs may also be used as single-ended inputs if SC1n[DIFF] is low. See the chip configuration chapter for ADC connections specific to this MCU.

## 35.3 Register definition

This section describes the ADC registers.

**ADC memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4003_B000 | ADC Status and Control Registers 1 (ADC0_SC1A) | 32 | R/W | 0000_001Fh | 35.3.1/797 |
| 4003_B004 | ADC Status and Control Registers 1 (ADC0_SC1B) | 32 | R/W | 0000_001Fh | 35.3.1/797 |
| 4003_B008 | ADC Configuration Register 1 (ADC0_CFG1) | 32 | R/W | 0000_0000h | 35.3.2/800 |
| 4003_B00C | ADC Configuration Register 2 (ADC0_CFG2) | 32 | R/W | 0000_0000h | 35.3.3/802 |
| 4003_B010 | ADC Data Result Register (ADC0_RA) | 32 | R | 0000_0000h | 35.3.4/803 |
| 4003_B014 | ADC Data Result Register (ADC0_RB) | 32 | R | 0000_0000h | 35.3.4/803 |
| 4003_B018 | Compare Value Registers (ADC0_CV1) | 32 | R/W | 0000_0000h | 35.3.5/804 |
| 4003_B01C | Compare Value Registers (ADC0_CV2) | 32 | R/W | 0000_0000h | 35.3.5/804 |

*Table continues on the next page...*

## ADC memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4003_B020 | Status and Control Register 2 (ADC0_SC2) | 32 | R/W | 0000_0000h | 35.3.6/805 |
| 4003_B024 | Status and Control Register 3 (ADC0_SC3) | 32 | R/W | 0000_0000h | 35.3.7/807 |
| 4003_B028 | ADC Offset Correction Register (ADC0_OFS) | 32 | R/W | 0000_0004h | 35.3.8/809 |
| 4003_B02C | ADC Plus-Side Gain Register (ADC0_PG) | 32 | R/W | 0000_8200h | 35.3.9/809 |
| 4003_B030 | ADC Minus-Side Gain Register (ADC0_MG) | 32 | R/W | 0000_8200h | 35.3.10/ 810 |
| 4003_B034 | ADC Plus-Side General Calibration Value Register (ADC0_CLPD) | 32 | R/W | 0000_000Ah | 35.3.11/ 810 |
| 4003_B038 | ADC Plus-Side General Calibration Value Register (ADC0_CLPS) | 32 | R/W | 0000_0020h | 35.3.12/ 811 |
| 4003_B03C | ADC Plus-Side General Calibration Value Register (ADC0_CLP4) | 32 | R/W | 0000_0200h | 35.3.13/ 811 |
| 4003_B040 | ADC Plus-Side General Calibration Value Register (ADC0_CLP3) | 32 | R/W | 0000_0100h | 35.3.14/ 812 |
| 4003_B044 | ADC Plus-Side General Calibration Value Register (ADC0_CLP2) | 32 | R/W | 0000_0080h | 35.3.15/ 812 |
| 4003_B048 | ADC Plus-Side General Calibration Value Register (ADC0_CLP1) | 32 | R/W | 0000_0040h | 35.3.16/ 813 |
| 4003_B04C | ADC Plus-Side General Calibration Value Register (ADC0_CLP0) | 32 | R/W | 0000_0020h | 35.3.17/ 813 |
| 4003_B054 | ADC Minus-Side General Calibration Value Register (ADC0_CLMD) | 32 | R/W | 0000_000Ah | 35.3.18/ 814 |
| 4003_B058 | ADC Minus-Side General Calibration Value Register (ADC0_CLMS) | 32 | R/W | 0000_0020h | 35.3.19/ 814 |
| 4003_B05C | ADC Minus-Side General Calibration Value Register (ADC0_CLM4) | 32 | R/W | 0000_0200h | 35.3.20/ 815 |
| 4003_B060 | ADC Minus-Side General Calibration Value Register (ADC0_CLM3) | 32 | R/W | 0000_0100h | 35.3.21/ 815 |
| 4003_B064 | ADC Minus-Side General Calibration Value Register (ADC0_CLM2) | 32 | R/W | 0000_0080h | 35.3.22/ 816 |
| 4003_B068 | ADC Minus-Side General Calibration Value Register (ADC0_CLM1) | 32 | R/W | 0000_0040h | 35.3.23/ 816 |
| 4003_B06C | ADC Minus-Side General Calibration Value Register (ADC0_CLM0) | 32 | R/W | 0000_0020h | 35.3.24/ 817 |
| 400B_B000 | ADC Status and Control Registers 1 (ADC1_SC1A) | 32 | R/W | 0000_001Fh | 35.3.1/797 |
| 400B_B004 | ADC Status and Control Registers 1 (ADC1_SC1B) | 32 | R/W | 0000_001Fh | 35.3.1/797 |
| 400B_B008 | ADC Configuration Register 1 (ADC1_CFG1) | 32 | R/W | 0000_0000h | 35.3.2/800 |
| 400B_B00C | ADC Configuration Register 2 (ADC1_CFG2) | 32 | R/W | 0000_0000h | 35.3.3/802 |
| 400B_B010 | ADC Data Result Register (ADC1_RA) | 32 | R | 0000_0000h | 35.3.4/803 |
| 400B_B014 | ADC Data Result Register (ADC1_RB) | 32 | R | 0000_0000h | 35.3.4/803 |
| 400B_B018 | Compare Value Registers (ADC1_CV1) | 32 | R/W | 0000_0000h | 35.3.5/804 |
| 400B_B01C | Compare Value Registers (ADC1_CV2) | 32 | R/W | 0000_0000h | 35.3.5/804 |

*Table continues on the next page...*

**ADC memory map (continued)**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 400B_B020 | Status and Control Register 2 (ADC1_SC2) | 32 | R/W | 0000_0000h | 35.3.6/805 |
| 400B_B024 | Status and Control Register 3 (ADC1_SC3) | 32 | R/W | 0000_0000h | 35.3.7/807 |
| 400B_B028 | ADC Offset Correction Register (ADC1_OFS) | 32 | R/W | 0000_0004h | 35.3.8/809 |
| 400B_B02C | ADC Plus-Side Gain Register (ADC1_PG) | 32 | R/W | 0000_8200h | 35.3.9/809 |
| 400B_B030 | ADC Minus-Side Gain Register (ADC1_MG) | 32 | R/W | 0000_8200h | 35.3.10/ 810 |
| 400B_B034 | ADC Plus-Side General Calibration Value Register (ADC1_CLPD) | 32 | R/W | 0000_000Ah | 35.3.11/ 810 |
| 400B_B038 | ADC Plus-Side General Calibration Value Register (ADC1_CLPS) | 32 | R/W | 0000_0020h | 35.3.12/ 811 |
| 400B_B03C | ADC Plus-Side General Calibration Value Register (ADC1_CLP4) | 32 | R/W | 0000_0200h | 35.3.13/ 811 |
| 400B_B040 | ADC Plus-Side General Calibration Value Register (ADC1_CLP3) | 32 | R/W | 0000_0100h | 35.3.14/ 812 |
| 400B_B044 | ADC Plus-Side General Calibration Value Register (ADC1_CLP2) | 32 | R/W | 0000_0080h | 35.3.15/ 812 |
| 400B_B048 | ADC Plus-Side General Calibration Value Register (ADC1_CLP1) | 32 | R/W | 0000_0040h | 35.3.16/ 813 |
| 400B_B04C | ADC Plus-Side General Calibration Value Register (ADC1_CLP0) | 32 | R/W | 0000_0020h | 35.3.17/ 813 |
| 400B_B054 | ADC Minus-Side General Calibration Value Register (ADC1_CLMD) | 32 | R/W | 0000_000Ah | 35.3.18/ 814 |
| 400B_B058 | ADC Minus-Side General Calibration Value Register (ADC1_CLMS) | 32 | R/W | 0000_0020h | 35.3.19/ 814 |
| 400B_B05C | ADC Minus-Side General Calibration Value Register (ADC1_CLM4) | 32 | R/W | 0000_0200h | 35.3.20/ 815 |
| 400B_B060 | ADC Minus-Side General Calibration Value Register (ADC1_CLM3) | 32 | R/W | 0000_0100h | 35.3.21/ 815 |
| 400B_B064 | ADC Minus-Side General Calibration Value Register (ADC1_CLM2) | 32 | R/W | 0000_0080h | 35.3.22/ 816 |
| 400B_B068 | ADC Minus-Side General Calibration Value Register (ADC1_CLM1) | 32 | R/W | 0000_0040h | 35.3.23/ 816 |
| 400B_B06C | ADC Minus-Side General Calibration Value Register (ADC1_CLM0) | 32 | R/W | 0000_0020h | 35.3.24/ 817 |

## 35.3.1 ADC Status and Control Registers 1 (ADC*x*_SC1*n*)

SC1A is used for both software and hardware trigger modes of operation.

To allow sequential conversions of the ADC to be triggered by internal peripherals, the ADC can have more then one status and control register: one for each conversion. The SC1B–SC1n registers indicate potentially multiple SC1 registers for use only in hardware

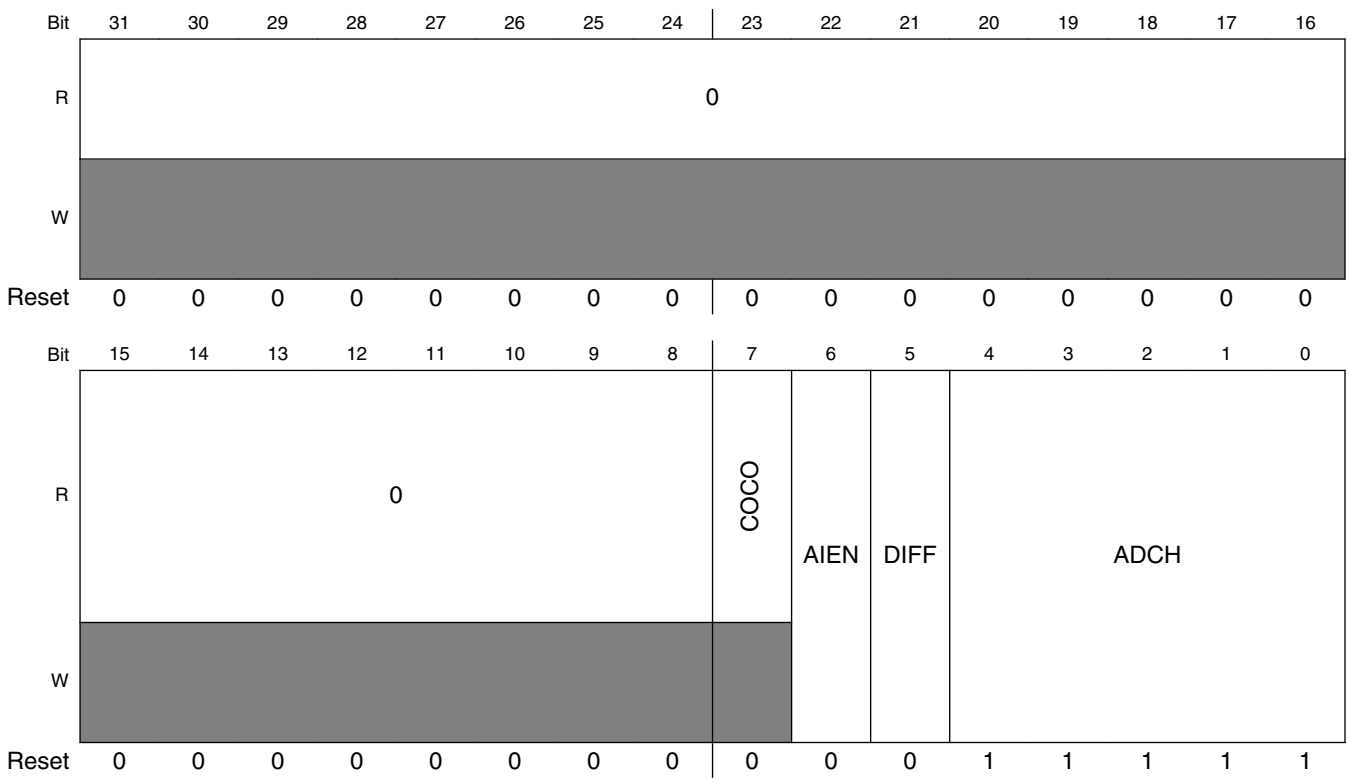
trigger mode. See the chip configuration information about the number of SC1n registers specific to this device. The SC1n registers have identical fields, and are used in a "ping-pong" approach to control ADC operation.

At any one point in time, only one of the SC1n registers is actively controlling ADC conversions. Updating SC1A while SC1n is actively controlling a conversion is allowed, and vice-versa for any of the SC1n registers specific to this MCU.

Writing SC1A while SC1A is actively controlling a conversion aborts the current conversion. In Software Trigger mode, when SC2[ADTRG]=0, writes to SC1A subsequently initiate a new conversion, if SC1[ADCH] contains a value other than all 1s.

Writing any of the SC1n registers while that specific SC1n register is actively controlling a conversion aborts the current conversion. None of the SC1B-SC1n registers are used for software trigger operation and therefore writes to the SC1B–SC1n registers do not initiate a new conversion.

Address: Base address + 0h offset + (4d × i), where i=0d to 1d




**ADCx_SC1n field descriptions**

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>COCO | Conversion Complete Flag |

*Table continues on the next page...*

## ADCx_SC1*n* field descriptions (continued)

| Field | Description |
|---|---|
| | This is a read-only field that is set each time a conversion is completed when the compare function is disabled, or SC2[ACFE]=0 and the hardware average function is disabled, or SC3[AVGE]=0. When the compare function is enabled, or SC2[ACFE]=1, COCO is set upon completion of a conversion only if the compare result is true. When the hardware average function is enabled, or SC3[AVGE]=1, COCO is set upon completion of the selected number of conversions (determined by AVGS). COCO in SC1A is also set at the completion of a calibration sequence. COCO is cleared when the respective SC1n register is written or when the respective Rn register is read.<br><br>0 Conversion is not completed.<br>1 Conversion is completed. |
| 6<br>AIEN | Interrupt Enable<br><br>Enables conversion complete interrupts. When COCO becomes set while the respective AIEN is high, an interrupt is asserted.<br><br>0 Conversion complete interrupt is disabled.<br>1 Conversion complete interrupt is enabled. |
| 5<br>DIFF | Differential Mode Enable<br><br>Configures the ADC to operate in differential mode. When enabled, this mode automatically selects from the differential channels, and changes the conversion algorithm and the number of cycles to complete a conversion.<br><br>0 Single-ended conversions and input channels are selected.<br>1 Differential conversions and input channels are selected. |
| 4–0<br>ADCH | Input channel select<br><br>Selects one of the input channels. The input channel decode depends on the value of DIFF. DAD0-DAD3 are associated with the input pin pairs DADPx and DADMx.<br><br>**NOTE:** Some of the input channel options in the bitfield-setting descriptions might not be available for your device. For the actual ADC channel assignments for your device, see the Chip Configuration details.<br><br>The successive approximation converter subsystem is turned off when the channel select bits are all set, that is, ADCH = 11111. This feature allows explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way prevents an additional single conversion from being performed. It is not necessary to set ADCH to all 1s to place the ADC in a low-power state when continuous conversions are not enabled because the module automatically enters a low-power state when a conversion completes.<br><br>00000 When DIFF=0, DADP0 is selected as input; when DIFF=1, DAD0 is selected as input.<br>00001 When DIFF=0, DADP1 is selected as input; when DIFF=1, DAD1 is selected as input.<br>00010 When DIFF=0, DADP2 is selected as input; when DIFF=1, DAD2 is selected as input.<br>00011 When DIFF=0, DADP3 is selected as input; when DIFF=1, DAD3 is selected as input.<br>00100 When DIFF=0, AD4 is selected as input; when DIFF=1, it is reserved.<br>00101 When DIFF=0, AD5 is selected as input; when DIFF=1, it is reserved.<br>00110 When DIFF=0, AD6 is selected as input; when DIFF=1, it is reserved.<br>00111 When DIFF=0, AD7 is selected as input; when DIFF=1, it is reserved.<br>01000 When DIFF=0, AD8 is selected as input; when DIFF=1, it is reserved.<br>01001 When DIFF=0, AD9 is selected as input; when DIFF=1, it is reserved.<br>01010 When DIFF=0, AD10 is selected as input; when DIFF=1, it is reserved.<br>01011 When DIFF=0, AD11 is selected as input; when DIFF=1, it is reserved. |

*Table continues on the next page...*

**ADCx_SC1*n* field descriptions (continued)**

| Field | Description |
|---|---|
| | 01100 When DIFF=0, AD12 is selected as input; when DIFF=1, it is reserved. |
| | 01101 When DIFF=0, AD13 is selected as input; when DIFF=1, it is reserved. |
| | 01110 When DIFF=0, AD14 is selected as input; when DIFF=1, it is reserved. |
| | 01111 When DIFF=0, AD15 is selected as input; when DIFF=1, it is reserved. |
| | 10000 When DIFF=0, AD16 is selected as input; when DIFF=1, it is reserved. |
| | 10001 When DIFF=0, AD17 is selected as input; when DIFF=1, it is reserved. |
| | 10010 When DIFF=0, AD18 is selected as input; when DIFF=1, it is reserved. |
| | 10011 When DIFF=0, AD19 is selected as input; when DIFF=1, it is reserved. |
| | 10100 When DIFF=0, AD20 is selected as input; when DIFF=1, it is reserved. |
| | 10101 When DIFF=0, AD21 is selected as input; when DIFF=1, it is reserved. |
| | 10110 When DIFF=0, AD22 is selected as input; when DIFF=1, it is reserved. |
| | 10111 When DIFF=0, AD23 is selected as input; when DIFF=1, it is reserved. |
| | 11000 Reserved. |
| | 11001 Reserved. |
| | 11010 When DIFF=0, Temp Sensor (single-ended) is selected as input; when DIFF=1, Temp Sensor (differential) is selected as input. |
| | 11011 When DIFF=0,Bandgap (single-ended) is selected as input; when DIFF=1, Bandgap (differential) is selected as input. |
| | 11100 Reserved. |
| | 11101 When DIFF=0,$V_{REFSH}$ is selected as input; when DIFF=1, -$V_{REFSH}$ (differential) is selected as input. Voltage reference selected is determined by SC2[REFSEL]. |
| | 11110 When DIFF=0,$V_{REFSL}$ is selected as input; when DIFF=1, it is reserved. Voltage reference selected is determined by SC2[REFSEL]. |
| | 11111 Module is disabled. |

## 35.3.2 ADC Configuration Register 1 (ADCx_CFG1)

The configuration Register 1 (CFG1) selects the mode of operation, clock source, clock divide, and configuration for low power or long sample time.

Address: Base address + 8h offset

| Bit | 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | ADLPC | ADIV | | ADLSMP | MODE | | ADICLK | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

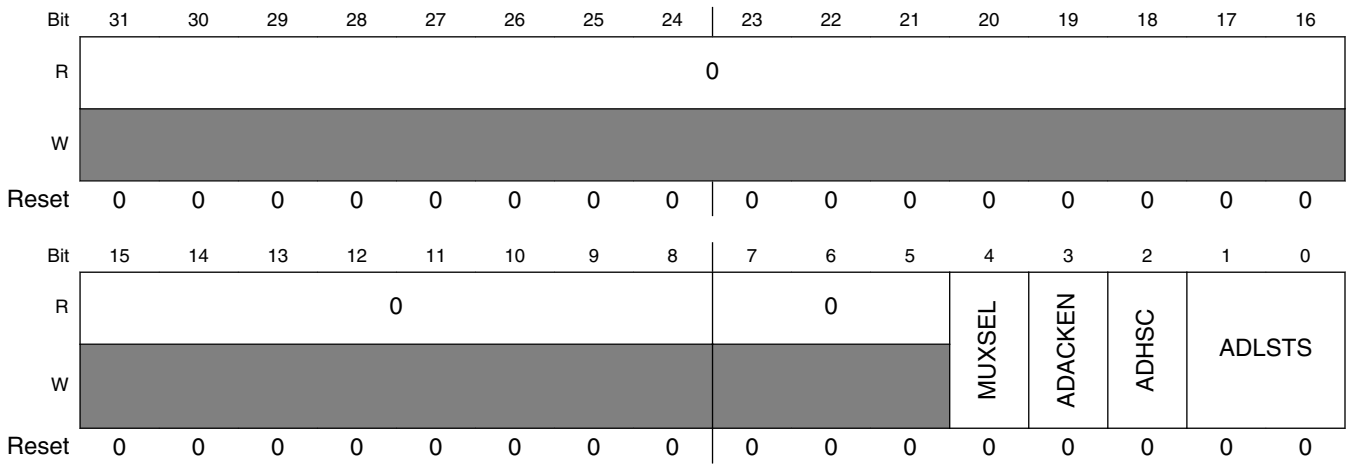
## ADCx_CFG1 field descriptions

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>ADLPC | Low-Power Configuration<br><br>Controls the power configuration of the successive approximation converter. This optimizes power consumption when higher sample rates are not required.<br><br>0 Normal power configuration.<br>1 Low-power configuration. The power is reduced at the expense of maximum clock speed. |
| 6–5<br>ADIV | Clock Divide Select<br><br>ADIV selects the divide ratio used by the ADC to generate the internal clock ADCK.<br><br>00 The divide ratio is 1 and the clock rate is input clock.<br>01 The divide ratio is 2 and the clock rate is (input clock)/2.<br>10 The divide ratio is 4 and the clock rate is (input clock)/4.<br>11 The divide ratio is 8 and the clock rate is (input clock)/8. |
| 4<br>ADLSMP | Sample time configuration<br><br>ADLSMP selects between different sample times based on the conversion mode selected. This bit adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption if continuous conversions are enabled and high conversion rates are not required. When ADLSMP=1, the long sample time select bits, (ADLSTS[1:0]), can select the extent of the long sample time.<br><br>0 Short sample time.<br>1 Long sample time. |
| 3–2<br>MODE | Conversion mode selection<br><br>Selects the ADC resolution mode.<br><br>00 When DIFF=0:It is single-ended 8-bit conversion; when DIFF=1, it is differential 9-bit conversion with 2's complement output.<br>01 When DIFF=0:It is single-ended 12-bit conversion ; when DIFF=1, it is differential 13-bit conversion with 2's complement output.<br>10 When DIFF=0:It is single-ended 10-bit conversion ; when DIFF=1, it is differential 11-bit conversion with 2's complement output.<br>11 When DIFF=0:It is single-ended 16-bit conversion; when DIFF=1, it is differential 16-bit conversion with 2's complement output. |
| 1–0<br>ADICLK | Input Clock Select<br><br>Selects the input clock source to generate the internal clock, ADCK. Note that when the ADACK clock source is selected, it is not required to be active prior to conversion start. When it is selected and it is not active prior to a conversion start, when CFG2[ADACKEN]=0, the asynchronous clock is activated at the start of a conversion and deactivated when conversions are terminated. In this case, there is an associated clock startup delay each time the clock source is re-activated.<br><br>00 Bus clock<br>01 (Bus clock)/2<br>10 Alternate clock (ALTCLK)<br>11 Asynchronous clock (ADACK) |

## 35.3.3 ADC Configuration Register 2 (ADCx_CFG2)

Configuration Register 2 (CFG2) selects the special high-speed configuration for very high speed conversions and selects the long sample time duration during long sample mode.

Address: Base address + Ch offset




**ADCx_CFG2 field descriptions**

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>MUXSEL | ADC Mux Select<br><br>Changes the ADC mux setting to select between alternate sets of ADC channels.<br><br>0 ADxxa channels are selected.<br>1 ADxxb channels are selected. |
| 3<br>ADACKEN | Asynchronous Clock Output Enable<br><br>Enables the asynchronous clock source and the clock source output regardless of the conversion and status of CFG1[ADICLK]. Based on MCU configuration, the asynchronous clock may be used by other modules. See chip configuration information. Setting this field allows the clock to be used even while the ADC is idle or operating from a different clock source. Also, latency of initiating a single or first-continuous conversion with the asynchronous clock selected is reduced because the ADACK clock is already operational.<br><br>0 Asynchronous clock output disabled; Asynchronous clock is enabled only if selected by ADICLK and a conversion is active.<br>1 Asynchronous clock and clock output is enabled regardless of the state of the ADC. |
| 2<br>ADHSC | High-Speed Configuration<br><br>Configures the ADC for very high-speed operation. The conversion sequence is altered with 2 ADCK cycles added to the conversion time to allow higher speed conversion clocks. |

*Table continues on the next page...*

**ADCx_CFG2 field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Normal conversion sequence selected.<br>1 High-speed conversion sequence selected with 2 additional ADCK cycles to total conversion time. |
| 1–0<br>ADLSTS | Long Sample Time Select<br><br>Selects between the extended sample times when long sample time is selected, that is, when CFG1[ADLSMP]=1. This allows higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption when continuous conversions are enabled if high conversion rates are not required.<br><br>00 Default longest sample time; 20 extra ADCK cycles; 24 ADCK cycles total.<br>01 12 extra ADCK cycles; 16 ADCK cycles total sample time.<br>10 6 extra ADCK cycles; 10 ADCK cycles total sample time.<br>11 2 extra ADCK cycles; 6 ADCK cycles total sample time. |

## 35.3.4 ADC Data Result Register (ADCx_R*n*)

The data result registers (Rn) contain the result of an ADC conversion of the channel selected by the corresponding status and channel control register (SC1A:SC1n). For every status and channel control register, there is a corresponding data result register.

Unused bits in R n are cleared in unsigned right-justified modes and carry the sign bit (MSB) in sign-extended 2's complement modes. For example, when configured for 10-bit single-ended mode, D[15:10] are cleared. When configured for 11-bit differential mode, D[15:10] carry the sign bit, that is, bit 10 extended through bit 15.

The following table describes the behavior of the data result registers in the different modes of operation.

**Table 35-43. Data result register 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 right justified |
| 13-bit differential | S | S | S | S | D | D | D | D | D | D | D | D | D | D | D | D | Sign-extended 2's complement |
| 12-bit single-ended | 0 | 0 | 0 | 0 | D | D | D | D | D | D | D | D | D | D | D | D | Unsigned right-justified |
| 11-bit differential | S | S | S | S | S | S | D | D | D | D | D | D | D | D | D | D | Sign-extended 2's complement |
| 10-bit single-ended | 0 | 0 | 0 | 0 | 0 | 0 | D | D | D | D | D | D | D | D | D | D | Unsigned right-justified |

*Table continues on the next page...*

**Table 35-43. Data result register description (continued)**

| Conversion mode | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Format |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 9-bit differential | S | S | S | S | S | S | S | S | D | D | D | D | D | D | D | D | Sign-extended 2's complement |
| 8-bit single-ended | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | D | D | D | D | D | D | D | D | Unsigned right-justified |

**NOTE**

S: Sign bit or sign bit extension;

D: Data, which is 2's complement data if indicated

Address: Base address + 10h offset + (4d × i), where i=0d to 1d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | D | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ADCx_Rn field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–0<br>D | Data result |

## 35.3.5 Compare Value Registers (ADCx_CVn)

The compare value registers (CV1 and CV2) contain a compare value used to compare the conversion result when the compare function is enabled, that is, SC2[ACFE]=1. This register is formatted in the same way as the Rn registers in different modes of operation for both bit position definition and value format using unsigned or sign-extended 2's complement. Therefore, the compare function uses only the CVn fields that are related to the ADC mode of operation.

The compare value 2 register (CV2) is used only when the compare range function is enabled, that is, SC2[ACREN]=1.

Address: Base address + 18h offset + (4d × i), where i=0d to 1d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | CV | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

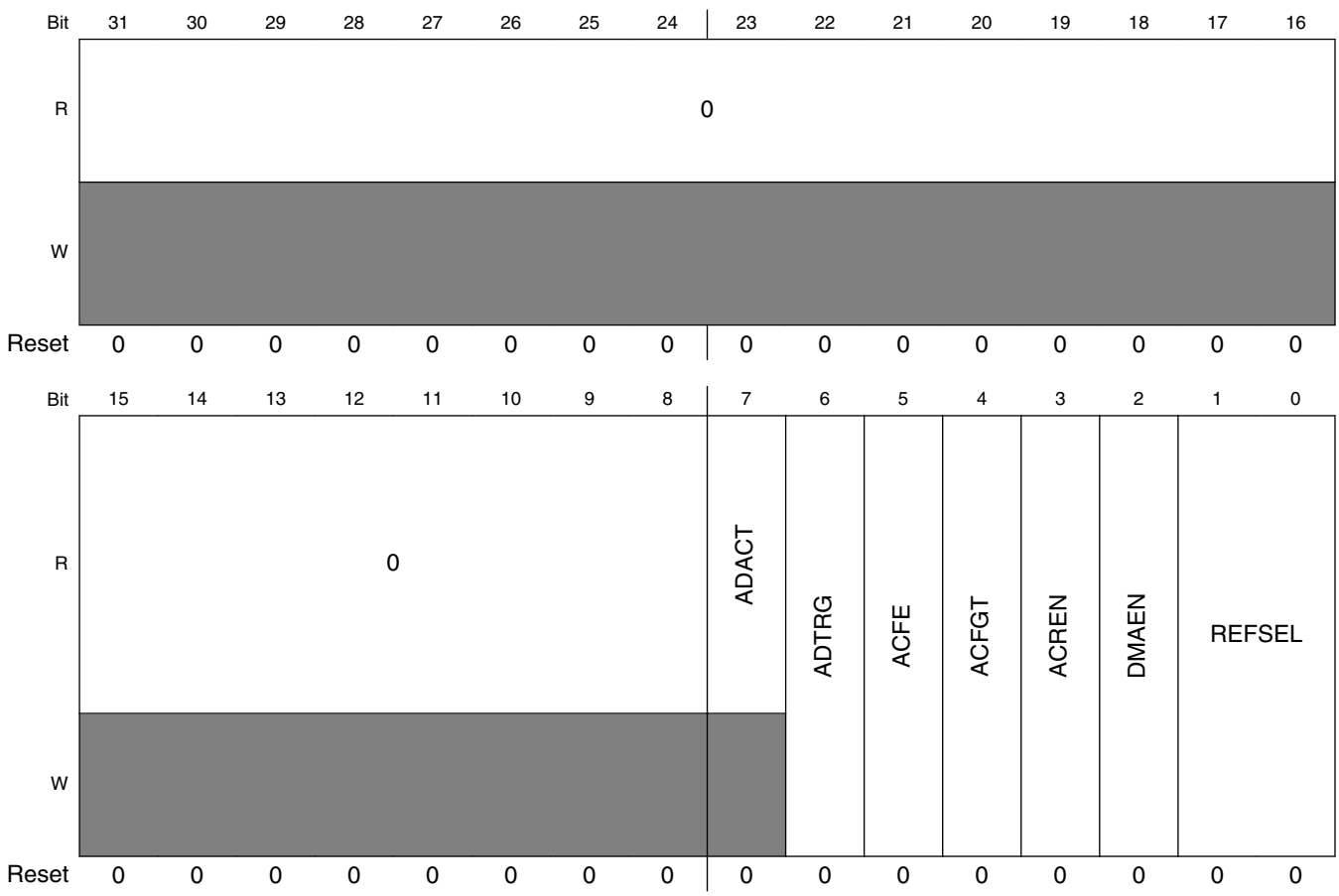
**ADCx_CV*n* field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–0<br>CV | Compare Value. |

## 35.3.6 Status and Control Register 2 (ADCx_SC2)

The status and control register 2 (SC2) contains the conversion active, hardware/software trigger select, compare function, and voltage reference select of the ADC module.

Address: Base address + 20h offset




**ADCx_SC2 field descriptions**

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>ADACT | Conversion Active |

*Table continues on the next page...*

## ADCx_SC2 field descriptions (continued)

| Field | Description |
|---|---|
| | Indicates that a conversion or hardware averaging is in progress. ADACT is set when a conversion is initiated and cleared when a conversion is completed or aborted.<br><br>0 Conversion not in progress.<br>1 Conversion in progress. |
| 6<br>ADTRG | Conversion Trigger Select<br><br>Selects the type of trigger used for initiating a conversion. Two types of trigger are selectable:<br>• Software trigger: When software trigger is selected, a conversion is initiated following a write to SC1A.<br>• Hardware trigger: When hardware trigger is selected, a conversion is initiated following the assertion of the ADHWT input after a pulse of the ADHWTSn input.<br><br>0 Software trigger selected.<br>1 Hardware trigger selected. |
| 5<br>ACFE | Compare Function Enable<br><br>Enables the compare function.<br><br>0 Compare function disabled.<br>1 Compare function enabled. |
| 4<br>ACFGT | Compare Function Greater Than Enable<br><br>Configures the compare function to check the conversion result relative to the CV1 and CV2 based upon the value of ACREN. ACFE must be set for ACFGT to have any effect.<br><br>0 Configures less than threshold, outside range not inclusive and inside range not inclusive; functionality based on the values placed in CV1 and CV2.<br>1 Configures greater than or equal to threshold, outside and inside ranges inclusive; functionality based on the values placed in CV1 and CV2. |
| 3<br>ACREN | Compare Function Range Enable<br><br>Configures the compare function to check if the conversion result of the input being monitored is either between or outside the range formed by CV1 and CV2 determined by the value of ACFGT. ACFE must be set for ACFGT to have any effect.<br><br>0 Range function disabled. Only CV1 is compared.<br>1 Range function enabled. Both CV1 and CV2 are compared. |
| 2<br>DMAEN | DMA Enable<br><br>0 DMA is disabled.<br>1 DMA is enabled and will assert the ADC DMA request during an ADC conversion complete event noted when any of the SC1n[COCO] flags is asserted. |
| 1–0<br>REFSEL | Voltage Reference Selection<br><br>Selects the voltage reference source used for conversions.<br><br>00 Default voltage reference pin pair, that is, external pins $V_{REFH}$ and $V_{REFL}$<br>01 Alternate reference pair, that is, $V_{ALTH}$ and $V_{ALTL}$ . This pair may be additional external pins or internal sources depending on the MCU configuration. See the chip configuration information for details specific to this MCU |

*Table continues on the next page...*

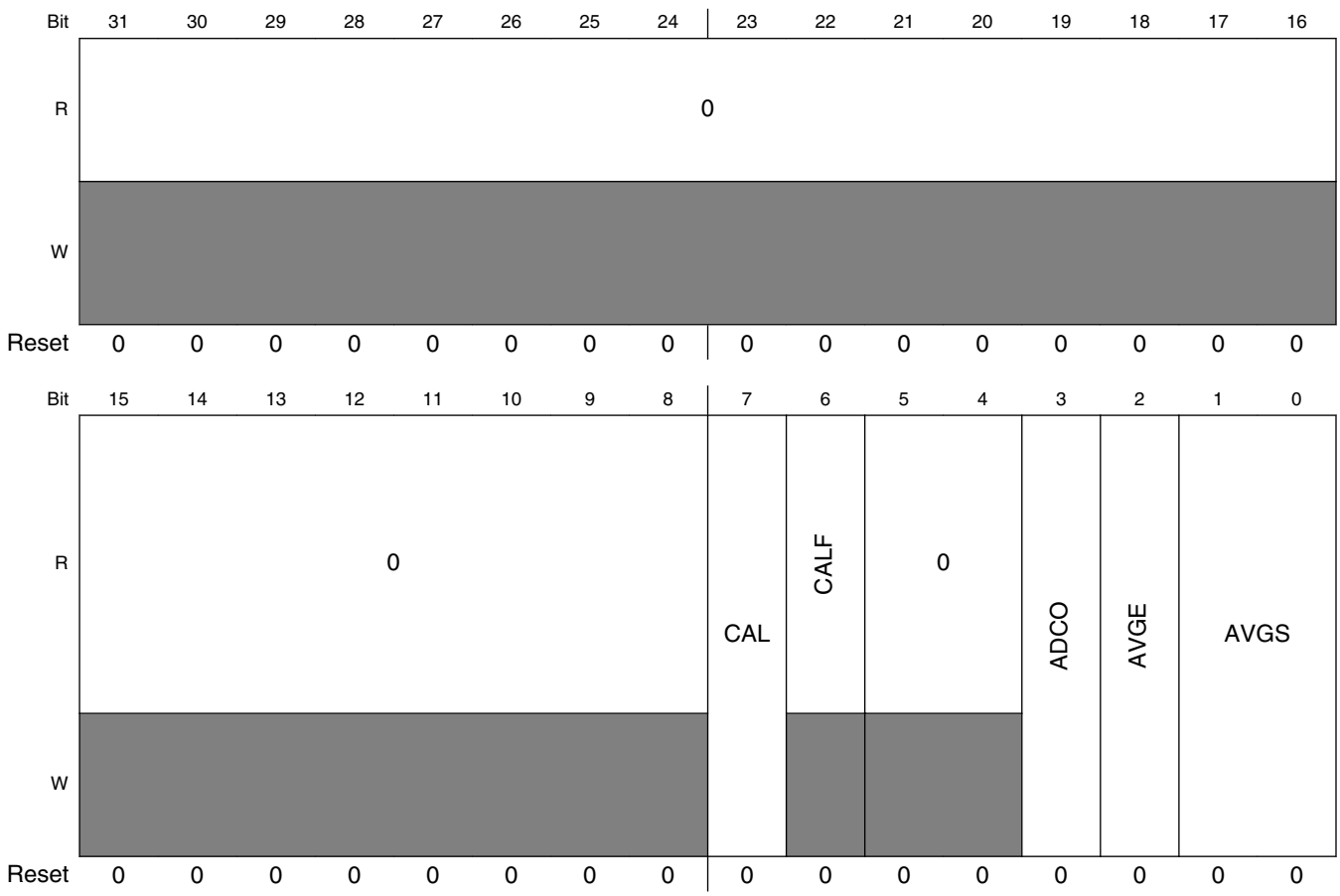
**ADCx_SC2 field descriptions (continued)**

| Field | Description |
|---|---|
| | 10 Reserved<br>11 Reserved |

## 35.3.7 Status and Control Register 3 (ADCx_SC3)

The Status and Control Register 3 (SC3) controls the calibration, continuous convert, and hardware averaging functions of the ADC module.

Address: Base address + 24h offset




**ADCx_SC3 field descriptions**

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>CAL | Calibration<br><br>Begins the calibration sequence when set. This field stays set while the calibration is in progress and is cleared when the calibration sequence is completed. CALF must be checked to determine the result of the |

*Table continues on the next page...*

## ADCx_SC3 field descriptions (continued)

| Field | Description |
|---|---|
| | calibration sequence. Once started, the calibration routine cannot be interrupted by writes to the ADC registers or the results will be invalid and CALF will set. Setting CAL will abort any current conversion. |
| 6<br>CALF | Calibration Failed Flag<br><br>Displays the result of the calibration sequence. The calibration sequence will fail if SC2[ADTRG] = 1, any ADC register is written, or any stop mode is entered before the calibration sequence completes. Writing 1 to CALF clears it.<br><br>0 Calibration completed normally.<br>1 Calibration failed. ADC accuracy specifications are not guaranteed. |
| 5–4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3<br>ADCO | Continuous Conversion Enable<br><br>Enables continuous conversions.<br><br>0 One conversion or one set of conversions if the hardware average function is enabled, that is, AVGE=1, after initiating a conversion.<br>1 Continuous conversions or sets of conversions if the hardware average function is enabled, that is, AVGE=1, after initiating a conversion. |
| 2<br>AVGE | Hardware Average Enable<br><br>Enables the hardware average function of the ADC.<br><br>0 Hardware average function disabled.<br>1 Hardware average function enabled. |
| 1–0<br>AVGS | Hardware Average Select<br><br>Determines how many ADC conversions will be averaged to create the ADC average result.<br><br>00 4 samples averaged.<br>01 8 samples averaged.<br>10 16 samples averaged.<br>11 32 samples averaged. |

## 35.3.8 ADC Offset Correction Register (ADCx_OFS)

The ADC Offset Correction Register (OFS) contains the user-selected or calibration-generated offset error correction value. This register is a 2's complement, left-justified, 16-bit value . The value in OFS is subtracted from the conversion and the result is transferred into the result registers, Rn. If the result is greater than the maximum or less than the minimum result value, it is forced to the appropriate limit for the current mode of operation.

Address: Base address + 28h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | OFS | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**ADCx_OFS field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–0<br>OFS | Offset Error Correction Value |

## 35.3.9 ADC Plus-Side Gain Register (ADCx_PG)

The Plus-Side Gain Register (PG) contains the gain error correction for the plus-side input in differential mode or the overall conversion in single-ended mode. PG, a 16-bit real number in binary format, is the gain adjustment factor, with the radix point fixed between ADPG15 and ADPG14. This register must be written by the user with the value described in the calibration procedure. Otherwise, the gain error specifications may not be met.

Address: Base address + 2Ch offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | PG | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ADCx_PG field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–0<br>PG | Plus-Side Gain |

## 35.3.10 ADC Minus-Side Gain Register (ADCx_MG)

The Minus-Side Gain Register (MG) contains the gain error correction for the minus-side input in differential mode. This register is ignored in single-ended mode. MG, a 16-bit real number in binary format, is the gain adjustment factor, with the radix point fixed between ADMG15 and ADMG14. This register must be written by the user with the value described in the calibration procedure. Otherwise, the gain error specifications may not be met.

Address: Base address + 30h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | MG | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

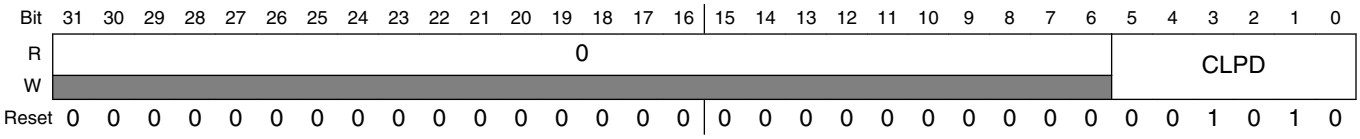
**ADCx_MG field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–0<br>MG | Minus-Side Gain |

## 35.3.11 ADC Plus-Side General Calibration Value Register (ADCx_CLPD)

The Plus-Side General Calibration Value Registers (CLPx) contain calibration information that is generated by the calibration function. These registers contain seven calibration values of varying widths: CLP0[5:0], CLP1[6:0], CLP2[7:0], CLP3[8:0], CLP4[9:0], CLPS[5:0], and CLPD[5:0]. CLPx are automatically set when the self-calibration sequence is done, that is, CAL is cleared. If these registers are written by the user after calibration, the linearity error specifications may not be met.

Address: Base address + 34h offset



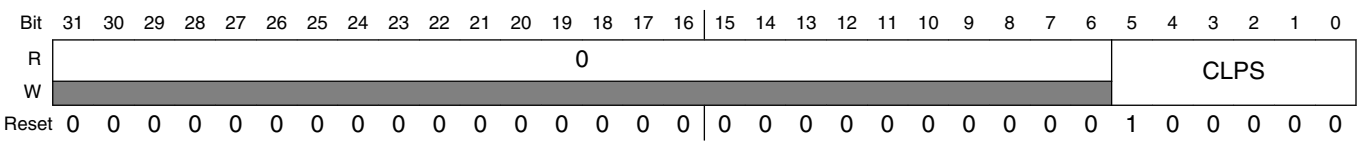

**ADCx_CLPD field descriptions**

| Field | Description |
|---|---|
| 31–6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5–0<br>CLPD | Calibration Value |

## 35.3.12 ADC Plus-Side General Calibration Value Register (ADCx_CLPS)

For more information, see CLPD register description.

Address: Base address + 38h offset



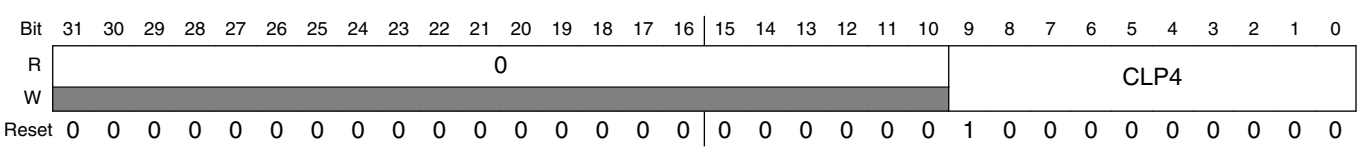

**ADCx_CLPS field descriptions**

| Field | Description |
|---|---|
| 31–6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5–0<br>CLPS | Calibration Value |

## 35.3.13 ADC Plus-Side General Calibration Value Register (ADCx_CLP4)

For more information, see CLPD register description.

Address: Base address + 3Ch offset



**ADCx_CLP4 field descriptions**

| Field | Description |
|---|---|
| 31–10<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 9–0<br>CLP4 | Calibration Value |

## 35.3.14 ADC Plus-Side General Calibration Value Register (ADCx_CLP3)

For more information, see CLPD register description.

Address: Base address + 40h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | CLP3 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ADCx_CLP3 field descriptions**

| Field | Description |
|---|---|
| 31–9<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 8–0<br>CLP3 | Calibration Value |

## 35.3.15 ADC Plus-Side General Calibration Value Register (ADCx_CLP2)

For more information, see CLPD register description.

Address: Base address + 44h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | | | | | | | | | CLP2 | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ADCx_CLP2 field descriptions**

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7–0<br>CLP2 | Calibration Value |

## 35.3.16 ADC Plus-Side General Calibration Value Register (ADCx_CLP1)

For more information, see CLPD register description.

Address: Base address + 48h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | CLP1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

**ADCx_CLP1 field descriptions**

| Field | Description |
|---|---|
| 31–7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6–0<br>CLP1 | Calibration Value |

## 35.3.17 ADC Plus-Side General Calibration Value Register (ADCx_CLP0)

For more information, see CLPD register description.

Address: Base address + 4Ch offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | CLP0 | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

**ADCx_CLP0 field descriptions**

| Field | Description |
|---|---|
| 31–6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5–0<br>CLP0 | Calibration Value |

## 35.3.18 ADC Minus-Side General Calibration Value Register (ADCx_CLMD)

The Minus-Side General Calibration Value (CLMx) registers contain calibration information that is generated by the calibration function. These registers contain seven calibration values of varying widths: CLM0[5:0], CLM1[6:0], CLM2[7:0], CLM3[8:0], CLM4[9:0], CLMS[5:0], and CLMD[5:0]. CLMx are automatically set when the self-calibration sequence is done, that is, CAL is cleared. If these registers are written by the user after calibration, the linearity error specifications may not be met.

Address: Base address + 54h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | CLMD | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

**ADCx_CLMD field descriptions**

| Field | Description |
|---|---|
| 31–6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5–0<br>CLMD | Calibration Value |

## 35.3.19 ADC Minus-Side General Calibration Value Register (ADCx_CLMS)

For more information, see CLMD register description.

Address: Base address + 58h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | CLMS | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

**ADCx_CLMS field descriptions**

| Field | Description |
|---|---|
| 31–6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5–0<br>CLMS | Calibration Value |

## 35.3.20 ADC Minus-Side General Calibration Value Register (ADCx_CLM4)

For more information, see CLMD register description.

Address: Base address + 5Ch offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | CLM4 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ADCx_CLM4 field descriptions**

| Field | Description |
|---|---|
| 31–10<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 9–0<br>CLM4 | Calibration Value |

## 35.3.21 ADC Minus-Side General Calibration Value Register (ADCx_CLM3)

For more information, see CLMD register description.

Address: Base address + 60h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | | | | | | | | CLM3 | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ADCx_CLM3 field descriptions**

| Field | Description |
|---|---|
| 31–9<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 8–0<br>CLM3 | Calibration Value |

## 35.3.22 ADC Minus-Side General Calibration Value Register (ADCx_CLM2)

For more information, see CLMD register description.

Address: Base address + 64h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | | | | | | | | | CLM2 | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ADCx_CLM2 field descriptions**

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7–0<br>CLM2 | Calibration Value |

## 35.3.23 ADC Minus-Side General Calibration Value Register (ADCx_CLM1)

For more information, see CLMD register description.

Address: Base address + 68h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | | | | | | | | | | CLM1 | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

**ADCx_CLM1 field descriptions**

| Field | Description |
|---|---|
| 31–7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6–0<br>CLM1 | Calibration Value |

### 35.3.24 ADC Minus-Side General Calibration Value Register (ADCx_CLM0)

For more information, see CLMD register description.

Address: Base address + 6Ch offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | CLM0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

**ADCx_CLM0 field descriptions**

| Field | Description |
|---|---|
| 31–6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5–0<br>CLM0 | Calibration Value |

## 35.4 Functional description

The ADC module is disabled during reset, in Low-Power Stop mode, or when SC1n[ADCH] are all high; see the power management information for details. The module is idle when a conversion has completed and another conversion has not been initiated. When it is idle and the asynchronous clock output enable is disabled, or CFG2[ADACKEN]= 0, the module is in its lowest power state. The ADC can perform an analog-to-digital conversion on any of the software selectable channels. All modes perform conversion by a successive approximation algorithm.

To meet accuracy specifications, the ADC module must be calibrated using the on-chip calibration function. See Calibration function for details on how to perform calibration.

When the conversion is completed, the result is placed in the Rn data registers. The respective SC1n[COCO] is then set and an interrupt is generated if the respective conversion complete interrupt has been enabled, or, when SC1n[AIEN]=1.

The ADC module has the capability of automatically comparing the result of a conversion with the contents of the CV1 and CV2 registers. The compare function is enabled by setting SC2[ACFE] and operates in any of the conversion modes and configurations.

The ADC module has the capability of automatically averaging the result of multiple conversions. The hardware average function is enabled by setting SC3[AVGE] and operates in any of the conversion modes and configurations.

**NOTE**

For the chip specific modes of operation, see the power management information of this MCU.

### 35.4.1 Clock select and divide control

One of four clock sources can be selected as the clock source for the ADC module. This clock source is then divided by a configurable value to generate the input clock ADCK, to the module. The clock is selected from one of the following sources by means of CFG1[ADICLK].

- Bus clock. This is the default selection following reset.
- Bus clock divided by two. For higher bus clock rates, this allows a maximum divide-by-16 of the bus clock using CFG1[ADIV].
- ALTCLK: As defined for this MCU. See the chip configuration information.
- Asynchronous clock (ADACK): This clock is generated from a clock source within the ADC module. When the ADACK clock source is selected, it is not required to be active prior to conversion start. When it is selected and it is not active prior to a conversion start CFG2[ADACKEN]=0, ADACK is activated at the start of a conversion and deactivated when conversions are terminated. In this case, there is an associated clock startup delay each time the clock source is re-activated. To avoid the conversion time variability and latency associated with the ADACK clock startup, set CFG2[ADACKEN]=1 and wait the worst-case startup time of 5 µs prior to initiating any conversions using the ADACK clock source. Conversions are possible using ADACK as the input clock source while the MCU is in Normal Stop mode. See Power Control for more information.

Whichever clock is selected, its frequency must fall within the specified frequency range for ADCK. If the available clocks are too slow, the ADC may not perform according to specifications. If the available clocks are too fast, the clock must be divided to the appropriate frequency. This divider is specified by CFG1[ADIV] and can be divide-by 1, 2, 4, or 8.

### 35.4.2 Voltage reference selection

The ADC can be configured to accept one of the two voltage reference pairs as the reference voltage ($V_{REFSH}$ and $V_{REFSL}$) used for conversions. Each pair contains a positive reference that must be between the minimum Ref Voltage High and $V_{DDA}$, and a ground reference that must be at the same potential as $V_{SSA}$. The two pairs are external ($V_{REFH}$ and $V_{REFL}$) and alternate ($V_{ALTH}$ and $V_{ALTL}$). These voltage references are selected using SC2[REFSEL]. The alternate ($V_{ALTH}$ and $V_{ALTL}$) voltage reference pair may select additional external pins or internal sources depending on MCU configuration. See the chip configuration information on the voltage references specific to this MCU.

### 35.4.3 Hardware trigger and channel selects

The ADC module has a selectable asynchronous hardware conversion trigger, ADHWT, that is enabled when SC2[ADTRG] is set and a hardware trigger select event, ADHWTSn, has occurred. This source is not available on all MCUs. See the Chip Configuration chapter for information on the ADHWT source and the ADHWTSn configurations specific to this MCU.

When an ADHWT source is available and hardware trigger is enabled, that is SC2[ADTRG]=1, a conversion is initiated on the rising-edge of ADHWT after a hardware trigger select event, that is, ADHWTSn, has occurred. If a conversion is in progress when a rising-edge of a trigger occurs, the rising-edge is ignored. In continuous convert configuration, only the initial rising-edge to launch continuous conversions is observed, and until conversion is aborted, the ADC continues to do conversions on the same SCn register that initiated the conversion. The hardware trigger function operates in conjunction with any of the conversion modes and configurations.

The hardware trigger select event, that is, ADHWTSn, must be set prior to the receipt of the ADHWT signal. If these conditions are not met, the converter may ignore the trigger or use the incorrect configuration. If a hardware trigger select event is asserted during a conversion, it must stay asserted until the end of current conversion and remain set until the receipt of the ADHWT signal to trigger a new conversion. The channel and status fields selected for the conversion depend on the active trigger select signal:

- ADHWTSA active selects SC1A
- ADHWTSn active selects SC1n

**Note**

Asserting more than one hardware trigger select signal (ADHWTSn) at the same time results in unknown results. To avoid this, select only one hardware trigger select signal (ADHWTSn) prior to the next intended conversion.

When the conversion is completed, the result is placed in the Rn registers associated with the ADHWTSn received. For example:

- ADHWTSA active selects RA register
- ADHWTSn active selects Rn register

The conversion complete flag associated with the ADHWTSn received, that is, SC1n[COCO], is then set and an interrupt is generated if the respective conversion complete interrupt has been enabled, that is, SC1[AIEN]=1.

## 35.4.4 Conversion control

Conversions can be performed as determined by CFG1[MODE] and SC1n[DIFF] as shown in the description of CFG1[MODE].

Conversions can be initiated by a software or hardware trigger. In addition, the ADC module can be configured for:

- Low-power operation
- Long sample time
- Continuous conversion
- Hardware average
- Automatic compare of the conversion result to a software determined compare value

### 35.4.4.1 Initiating conversions

A conversion is initiated:

- Following a write to SC1A, with SC1n[ADCH] not all 1's, if software triggered operation is selected, that is, when SC2[ADTRG]=0.
- Following a hardware trigger, or ADHWT event, if hardware triggered operation is selected, that is, SC2[ADTRG]=1, and a hardware trigger select event, ADHWTSn, has occurred. The channel and status fields selected depend on the active trigger select signal:
  - ADHWTSA active selects SC1A
  - ADHWTSn active selects SC1n
  - if neither is active, the off condition is selected

**Note**

Selecting more than one ADHWTSn prior to a conversion completion will result in unknown results. To avoid this, select only one ADHWTSn prior to a conversion completion.

- Following the transfer of the result to the data registers when continuous conversion is enabled, that is, when ADCO=1.

If continuous conversions are enabled, a new conversion is automatically initiated after the completion of the current conversion, by:. In software triggered operation, that is, when ADTRG=0, continuous conversions begin after SC1A is written and continue until aborted. In hardware triggered operation, that is, when ADTRG=1 and one ADHWTSn event has occurred, continuous conversions begin after a hardware trigger event and continue until aborted.

If hardware averaging is enabled, a new conversion is automatically initiated after the completion of the current conversion until the correct number of conversions are completed. In software triggered operation, conversions begin after SC1A is written. In hardware triggered operation, conversions begin after a hardware trigger. If continuous conversions are also enabled, a new set of conversions to be averaged are initiated following the last of the selected number of conversions.

### 35.4.4.2 Completing conversions

A conversion is completed when the result of the conversion is transferred into the data result registers, Rn. If the compare functions are disabled, this is indicated by setting of SC1n[COCO]. If hardware averaging is enabled, the respective SC1n[COCO] sets only if the last of the selected number of conversions is completed. If the compare function is enabled, the respective SC1n[COCO] sets and conversion result data is transferred only if the compare condition is true. If both hardware averaging and compare functions are enabled, then the respective SC1n[COCO] sets only if the last of the selected number of conversions is completed and the compare condition is true. An interrupt is generated if the respective SC1n[AIEN] is high at the time that the respective SC1n[COCO] is set.

### 35.4.4.3 Aborting conversions

Any conversion in progress is aborted when:

- Writing to SC1A while it is actively controlling a conversion, aborts the current conversion. In Software Trigger mode, when SC2[ADTRG]=0, a write to SC1A initiates a new conversion if SC1A[ADCH] is equal to a value other than all 1s. Writing to any of the SC1B–SC1n registers while that specific SC1B–SC1n register is actively controlling a conversion aborts the current conversion. The SC1(B-n) registers are not used for software trigger operation and therefore writes to the SC1(B-n) registers do not initiate a new conversion.
- A write to any ADC register besides the SC1A-SC1n registers occurs. This indicates that a change in mode of operation has occurred and the current conversion is therefore invalid.
- The MCU is reset or enters Low-Power Stop modes.
- The MCU enters Normal Stop mode with ADACK not enabled.

When a conversion is aborted, the contents of the data registers, Rn, are not altered. The data registers continue to be the values transferred after the completion of the last successful conversion. If the conversion was aborted by a reset or Low-Power Stop modes, RA and Rn return to their reset states.

#### 35.4.4.4 Power control

The ADC module remains in its idle state until a conversion is initiated. If ADACK is selected as the conversion clock source, but the asynchronous clock output is disabled, that is CFG2[ADACKEN]=0, the ADACK clock generator also remains in its idle state (disabled) until a conversion is initiated. If the asynchronous clock output is enabled, that is, CFG2[ADACKEN]=1, it remains active regardless of the state of the ADC or the MCU power mode.

Power consumption when the ADC is active can be reduced by setting CFG1[ADLPC]. This results in a lower maximum value for $f_{ADCK}$.

#### 35.4.4.5 Sample time and total conversion time

For short sample, that is, when CFG1[ADLSMP]=0, there is a 2-cycle adder for first conversion over the base sample time of four ADCK cycles. For high speed conversions, that is, when CFG2[ADHSC]=1, there is an additional 2-cycle adder on any conversion. The table below summarizes sample times for the possible ADC configurations.

| ADC configuration | | | Sample time (ADCK cycles) | |
|---|---|---|---|---|
| CFG1[ADLSMP] | CFG2[ADLSTS] | CFG2[ADHSC] | First or Single | Subsequent |
| 0 | X | 0 | 6 | 4 |
| 1 | 00 | 0 | 24 | |
| 1 | 01 | 0 | 16 | |
| 1 | 10 | 0 | 10 | |
| 1 | 11 | 0 | 6 | |
| 0 | X | 1 | 8 | 6 |
| 1 | 00 | 1 | 26 | |
| 1 | 01 | 1 | 18 | |
| 1 | 10 | 1 | 12 | |
| 1 | 11 | 1 | 8 | |

The total conversion time depends upon:

- The sample time as determined by CFG1[ADLSMP] and CFG2[ADLSTS]
- The MCU bus frequency
- The conversion mode, as determined by CFG1[MODE] and SC1n[DIFF]
- The high speed configuration, that is, CFG2[ADHSC]
- The frequency of the conversion clock, that is, $f_{ADCK}$.

CFG2[ADHSC] is used to configure a higher clock input frequency. This will allow faster overall conversion times. To meet internal ADC timing requirements, CFG2[ADHSC] adds additional ADCK cycles. Conversions with CFG2[ADHSC]=1 take two more ADCK cycles. CFG2[ADHSC] must be used when the ADCLK exceeds the limit for CFG2[ADHSC]=0.

After the module becomes active, sampling of the input begins.

1. CFG1[ADLSMP] and CFG2[ADLSTS] select between sample times based on the conversion mode that is selected.
2. When sampling is completed, the converter is isolated from the input channel and a successive approximation algorithm is applied to determine the digital value of the analog signal.
3. The result of the conversion is transferred to Rn upon completion of the conversion algorithm.

If the bus frequency is less than $f_{ADCK}$, precise sample time for continuous conversions cannot be guaranteed when short sample is enabled, that is, when CFG1[ADLSMP]=0.

The maximum total conversion time is determined by the clock source chosen and the divide ratio selected. The clock source is selectable by CFG1[ADICLK], and the divide ratio is specified by CFG1[ADIV].

The maximum total conversion time for all configurations is summarized in the equation below. See the following tables for the variables referenced in the equation.

$$\text{ConversionTime} = \text{SFCAdder} + \text{AverageNum} \times (\text{BCT} + \text{LSTAdder} + \text{HSCAdder})$$

**Figure 35-92. Conversion time equation**

**Table 35-104. Single or first continuous time adder (SFCAdder)**

| CFG1[ADLSMP] | CFG2[ADACKEN] | CFG1[ADICLK] | Single or first continuous time adder (SFCAdder) |
|---|---|---|---|
| 1 | x | 0x, 10 | 3 ADCK cycles + 5 bus clock cycles |
| 1 | 1 | 11 | 3 ADCK cycles + 5 bus clock cycles[1] |
| 1 | 0 | 11 | 5 µs + 3 ADCK cycles + 5 bus clock cycles |
| 0 | x | 0x, 10 | 5 ADCK cycles + 5 bus clock cycles |
| 0 | 1 | 11 | 5 ADCK cycles + 5 bus clock cycles[1] |
| 0 | 0 | 11 | 5 µs + 5 ADCK cycles + 5 bus clock cycles |

1. To achieve this time, CFG2[ADACKEN] must be 1 for at least 5 µs prior to the conversion is initiated.

**Table 35-105. Average number factor (AverageNum)**

| SC3[AVGE] | SC3[AVGS] | Average number factor (AverageNum) |
|---|---|---|
| 0 | xx | 1 |
| 1 | 00 | 4 |
| 1 | 01 | 8 |
| 1 | 10 | 16 |
| 1 | 11 | 32 |

**Table 35-106. Base conversion time (BCT)**

| Mode | Base conversion time (BCT) |
|---|---|
| 8b single-ended | 17 ADCK cycles |
| 9b differential | 27 ADCK cycles |
| 10b single-ended | 20 ADCK cycles |
| 11b differential | 30 ADCK cycles |
| 12b single-ended | 20 ADCK cycles |
| 13b differential | 30 ADCK cycles |
| 16b single-ended | 25 ADCK cycles |
| 16b differential | 34 ADCK cycles |

**Table 35-107. Long sample time adder (LSTAdder)**

| CFG1[ADLSMP] | CFG2[ADLSTS] | Long sample time adder (LSTAdder) |
|---|---|---|
| 0 | xx | 0 ADCK cycles |
| 1 | 00 | 20 ADCK cycles |
| 1 | 01 | 12 ADCK cycles |
| 1 | 10 | 6 ADCK cycles |
| 1 | 11 | 2 ADCK cycles |

**Table 35-108. High-speed conversion time adder (HSCAdder)**

| CFG2[ADHSC] | High-speed conversion time adder (HSCAdder) |
|---|---|
| 0 | 0 ADCK cycles |
| 1 | 2 ADCK cycles |

**Note**

The ADCK frequency must be between $f_{ADCK}$ minimum and $f_{ADCK}$ maximum to meet ADC specifications.

### 35.4.4.6 Conversion time examples

The following examples use the Figure 35-92, and the information provided in Table 35-104 through Table 35-108.

#### 35.4.4.6.1 Typical conversion time configuration

A typical configuration for ADC conversion is:

- 10-bit mode, with the bus clock selected as the input clock source
- The input clock divide-by-1 ratio selected
- Bus frequency of 8 MHz
- Long sample time disabled
- High-speed conversion disabled

The conversion time for a single conversion is calculated by using the Figure 35-92, and the information provided in Table 35-104 through Table 35-108. The table below lists the variables of Figure 35-92.

**Table 35-109. Typical conversion time**

| Variable | Time |
|---|---|
| SFCAdder | 5 ADCK cycles + 5 bus clock cycles |

*Table continues on the next page...*

**Table 35-109. Typical conversion time (continued)**

| Variable | Time |
|---|---|
| AverageNum | 1 |
| BCT | 20 ADCK cycles |
| LSTAdder | 0 |
| HSCAdder | 0 |

The resulting conversion time is generated using the parameters listed in the preceding table. Therefore, for a bus clock and an ADCK frequency equal to 8 MHz, the resulting conversion time is 3.75 µs.

#### 35.4.4.6.2 Long conversion time configuration

A configuration for long ADC conversion is:

- 16-bit differential mode with the bus clock selected as the input clock source
- The input clock divide-by-8 ratio selected
- Bus frequency of 8 MHz
- Long sample time enabled
- Configured for longest adder
- High-speed conversion disabled
- Average enabled for 32 conversions

The conversion time for this conversion is calculated by using the Figure 35-92, and the information provided in Table 35-104 through Table 35-108. The following table lists the variables of the Figure 35-92.

**Table 35-110. Typical conversion time**

| Variable | Time |
|---|---|
| SFCAdder | 3 ADCK cycles + 5 bus clock cycles |
| AverageNum | 32 |
| BCT | 34 ADCK cycles |
| LSTAdder | 20 ADCK cycles |
| HSCAdder | 0 |

The resulting conversion time is generated using the parameters listed in the preceding table. Therefore, for bus clock equal to 8 MHz and ADCK equal to 1 MHz, the resulting conversion time is 57.625 µs, that is, AverageNum. This results in a total conversion time of 1.844 ms.

#### 35.4.4.6.3 Short conversion time configuration

A configuration for short ADC conversion is:

- 8-bit Single-Ended mode with the bus clock selected as the input clock source
- The input clock divide-by-1 ratio selected
- Bus frequency of 20 MHz
- Long sample time disabled
- High-speed conversion enabled

The conversion time for this conversion is calculated by using the Figure 35-92, and the information provided in Table 35-104 through Table 35-108. The table below lists the variables of Figure 35-92.

**Table 35-111. Typical conversion time**

| Variable | Time |
|---|---|
| SFCAdder | 5 ADCK cycles + 5 bus clock cycles |
| AverageNum | 1 |
| BCT | 17 ADCK cycles |
| LSTAdder | 0 ADCK cycles |
| HSCAdder | 2 |

The resulting conversion time is generated using the parameters listed in in the preceding table. Therefore, for bus clock and ADCK frequency equal to 20 MHz, the resulting conversion time is 1.45 µs.

### 35.4.4.7 Hardware average function

The hardware average function can be enabled by setting SC3[AVGE]=1 to perform a hardware average of multiple conversions. The number of conversions is determined by the AVGS[1:0] bits, which can select 4, 8, 16, or 32 conversions to be averaged. While the hardware average function is in progress, SC2[ADACT] will be set.

After the selected input is sampled and converted, the result is placed in an accumulator from which an average is calculated once the selected number of conversions have been completed. When hardware averaging is selected, the completion of a single conversion will not set SC1n[COCO].

If the compare function is either disabled or evaluates true, after the selected number of conversions are completed, the average conversion result is transferred into the data result registers, Rn, and SC1n[COCO] is set. An ADC interrupt is generated upon the setting of SC1n[COCO] if the respective ADC interrupt is enabled, that is, SC1n[AIEN]=1.

**Note**

The hardware average function can perform conversions on a channel while the MCU is in Wait or Normal Stop modes. The ADC interrupt wakes the MCU when the hardware average is completed if SC1n[AIEN] was set.

### 35.4.5 Automatic compare function

The compare function can be configured to check whether the result is less than or greater-than-or-equal-to a single compare value, or, if the result falls within or outside a range determined by two compare values. The compare mode is determined by SC2[ACFGT], SC2[ACREN], and the values in the compare value registers, CV1 and CV2. After the input is sampled and converted, the compare values in CV1 and CV2 are used as described in the following table. There are six Compare modes as shown in the following table.

**Table 35-112. Compare modes**

| SC2[ACFGT] | SC2[ACREN] | ADCCV1 relative to ADCCV2 | Function | Compare mode description |
|---|---|---|---|---|
| 0 | 0 | — | Less than threshold | Compare true if the result is less than the CV1 registers. |
| 1 | 0 | — | Greater than or equal to threshold | Compare true if the result is greater than or equal to CV1 registers. |
| 0 | 1 | Less than or equal | Outside range, not inclusive | Compare true if the result is less than CV1 **Or** the result is greater than CV2. |
| 0 | 1 | Greater than | Inside range, not inclusive | Compare true if the result is less than CV1 **And** the result is greater than CV2. |
| 1 | 1 | Less than or equal | Inside range, inclusive | Compare true if the result is greater than or equal to CV1 **And** the result is less than or equal to CV2. |
| 1 | 1 | Greater than | Outside range, inclusive | Compare true if the result is greater than or equal to CV1 **Or** the result is less than or equal to CV2. |

With SC2[ACREN] =1, and if the value of CV1 is less than or equal to the value of CV2, then setting SC2[ACFGT] will select a trigger-if-inside-compare-range inclusive-of-endpoints function. Clearing SC2[ACFGT] will select a trigger-if-outside-compare-range, not-inclusive-of-endpoints function.

If CV1 is greater than CV2, setting SC2[ACFGT] will select a trigger-if-outside-compare-range, inclusive-of-endpoints function. Clearing SC2[ACFGT] will select a trigger-if-inside-compare-range, not-inclusive-of-endpoints function.

If the condition selected evaluates true, SC1n[COCO] is set.

Upon completion of a conversion while the compare function is enabled, if the compare condition is not true, SC1n[COCO] is not set and the conversion result data will not be transferred to the result register, Rn. If the hardware averaging function is enabled, the compare function compares the averaged result to the compare values. The same compare function definitions apply. An ADC interrupt is generated when SC1n[COCO] is set and the respective ADC interrupt is enabled, that is, SC1n[AIEN]=1.

**Note**

The compare function can monitor the voltage on a channel while the MCU is in Wait or Normal Stop modes. The ADC interrupt wakes the MCU when the compare condition is met.

### 35.4.6 Calibration function

The ADC contains a self-calibration function that is required to achieve the specified accuracy. Calibration must be run, or valid calibration values written, after any reset and before a conversion is initiated. The calibration function sets the offset calibration value, the minus-side calibration values, and the plus-side calibration values. The offset calibration value is automatically stored in the ADC offset correction register (OFS), and the plus-side and minus-side calibration values are automatically stored in the ADC plus-side and minus-side calibration registers, CLPx and CLMx. The user must configure the ADC correctly prior to calibration, and must generate the plus-side and minus-side gain calibration results and store them in the ADC plus-side gain register (PG) after the calibration function completes.

Prior to calibration, the user must configure the ADC's clock source and frequency, low power configuration, voltage reference selection, sample time, and high speed configuration according to the application's clock source availability and needs. If the application uses the ADC in a wide variety of configurations, the configuration for which the highest accuracy is required should be selected, or multiple calibrations can be done for the different configurations. For best calibration results:

- Set hardware averaging to maximum, that is, SC3[AVGE]=1 and SC3[AVGS]=11 for an average of 32
- Set ADC clock frequency $f_{ADCK}$ less than or equal to 4 MHz
- $V_{REFH}=V_{DDA}$
- Calibrate at nominal voltage and temperature

The input channel, conversion mode continuous function, compare function, resolution mode, and differential/single-ended mode are all ignored during the calibration function.

To initiate calibration, the user sets SC3[CAL] and the calibration will automatically begin if the SC2[ADTRG] is 0. If SC2[ADTRG] is 1, SC3[CAL] will not get set and SC3[CALF] will be set. While calibration is active, no ADC register can be written and no stop mode may be entered, or the calibration routine will be aborted causing SC3[CAL] to clear and SC3[CALF] to set. At the end of a calibration sequence, SC1n[COCO] will be set. SC1n[AIEN] can be used to allow an interrupt to occur at the end of a calibration sequence. At the end of the calibration routine, if SC3[CALF] is not set, the automatic calibration routine is completed successfully.

To complete calibration, the user must generate the gain calibration values using the following procedure:

1. Initialize or clear a 16-bit variable in RAM.
2. Add the plus-side calibration results CLP0, CLP1, CLP2, CLP3, CLP4, and CLPS to the variable.
3. Divide the variable by two.
4. Set the MSB of the variable.
5. The previous two steps can be achieved by setting the carry bit, rotating to the right through the carry bit on the high byte and again on the low byte.
6. Store the value in the plus-side gain calibration register PG.
7. Repeat the procedure for the minus-side gain calibration value.

When calibration is complete, the user may reconfigure and use the ADC as desired. A second calibration may also be performed, if desired, by clearing and again setting SC3[CAL].

Overall, the calibration routine may take as many as 14k ADCK cycles and 100 bus cycles, depending on the results and the clock source chosen. For an 8 MHz clock source, this length amounts to about 1.7 ms. To reduce this latency, the calibration values, which are offset, plus-side and minus-side gain, and plus-side and minus-side calibration values, may be stored in flash memory after an initial calibration and recovered prior to the first ADC conversion. This method can reduce the calibration latency to 20 register store operations on all subsequent power, reset, or Low-Power Stop mode recoveries.

### 35.4.7 User-defined offset function

OFS contains the user-selected or calibration-generated offset error correction value. This register is a 2’s complement, left-justified. The value in OFS is subtracted from the conversion and the result is transferred into the result registers, Rn. If the result is greater than the maximum or less than the minimum result value, it is forced to the appropriate limit for the current mode of operation.

The formatting of the OFS is different from the data result register, Rn, to preserve the resolution of the calibration value regardless of the conversion mode selected. Lower order bits are ignored in lower resolution modes. For example, in 8-bit single-ended mode, OFS[14:7] are subtracted from D[7:0]; OFS[15] indicates the sign (negative numbers are effectively added to the result) and OFS[6:0] are ignored. The same bits are used in 9-bit differential mode because OFS[15] indicates the sign bit, which maps to D[8]. For 16-bit differential mode, OFS[15:0] are directly subtracted from the conversion result data D[15:0]. In 16-bit single-ended mode, there is no field in the OFS corresponding to the least significant result D[0], so odd values, such as -1 or +1, cannot be subtracted from the result.

OFS is automatically set according to calibration requirements once the self-calibration sequence is done, that is, SC3[CAL] is cleared. The user may write to OFS to override the calibration result if desired. If the OFS is written by the user to a value that is different from the calibration value, the ADC error specifications may not be met. Storing the value generated by the calibration function in memory before overwriting with a user-specified value is recommended.

**Note**

There is an effective limit to the values of offset that can be set by the user. If the magnitude of the offset is too high, the results of the conversions will cap off at the limits.

The offset calibration function may be employed by the user to remove application offsets or DC bias values. OFS may be written with a number in 2's complement format and this offset will be subtracted from the result, or hardware averaged value. To add an offset, store the negative offset in 2's complement format and the effect will be an addition. An offset correction that results in an out-of-range value will be forced to the minimum or maximum value. The minimum value for single-ended conversions is 0x0000; for a differential conversion it is 0x8000.

To preserve accuracy, the calibrated offset value initially stored in OFS must be added to the user-defined offset. For applications that may change the offset repeatedly during operation, store the initial offset calibration value in flash so it can be recovered and added to any user offset adjustment value and the sum stored in OFS.

### 35.4.8 Temperature sensor

The ADC module includes a temperature sensor whose output is connected to one of the ADC analog channel inputs. The following equation provides an approximate transfer function of the temperature sensor.

$$Temp = 25 - \left(\left( V_{TEMP} - V_{TEMP25} \right) \div m \right)$$

**Figure 35-93. Approximate transfer function of the temperature sensor**

where:

- $V_{TEMP}$ is the voltage of the temperature sensor channel at the ambient temperature.
- $V_{TEMP25}$ is the voltage of the temperature sensor channel at 25 °C.
- m is referred as temperature sensor slope in the device data sheet. It is the hot or cold voltage versus temperature slope in V/°C.

For temperature calculations, use the $V_{TEMP25}$ and temperature sensor slope values from the ADC Electricals table.

In application code, the user reads the temperature sensor channel, calculates $V_{TEMP}$, and compares to $V_{TEMP25}$. If $V_{TEMP}$ is greater than $V_{TEMP25}$ the cold slope value is applied in the preceding equation. If $V_{TEMP}$ is less than $V_{TEMP25}$, the hot slope value is applied in the preceding equation. ADC Electricals table may only specify one temperature sensor slope value. In that case, the user could use the same slope for the calculation across the operational temperature range.

For more information on using the temperature sensor, see the application note titled *Temperature Sensor for the HCS08 Microcontroller Family* (document AN3031).

### 35.4.9 MCU wait mode operation

Wait mode is a lower-power consumption Standby mode from which recovery is fast because the clock sources remain active. If a conversion is in progress when the MCU enters Wait mode, it continues until completion. Conversions can be initiated while the MCU is in Wait mode by means of the hardware trigger or if continuous conversions are enabled.

The bus clock, bus clock divided by two, and ADACK are available as conversion clock sources while in Wait mode. The use of ALTCLK as the conversion clock source in Wait is dependent on the definition of ALTCLK for this MCU. See the Chip Configuration information on ALTCLK specific to this MCU.

If the compare and hardware averaging functions are disabled, a conversion complete event sets SC1n[COCO] and generates an ADC interrupt to wake the MCU from Wait mode if the respective ADC interrupt is enabled, that is, when SC1n[AIEN]=1. If the hardware averaging function is enabled, SC1n[COCO] will set, and generate an interrupt if enabled, when the selected number of conversions are completed. If the compare function is enabled, SC1n[COCO] will set, and generate an interrupt if enabled, only if the compare conditions are met. If a single conversion is selected and the compare trigger is not met, the ADC will return to its idle state and cannot wake the MCU from Wait mode unless a new conversion is initiated by the hardware trigger.

## 35.4.10 MCU Normal Stop mode operation

Stop mode is a low-power consumption Standby mode during which most or all clock sources on the MCU are disabled.

### 35.4.10.1 Normal Stop mode with ADACK disabled

If the asynchronous clock, ADACK, is not selected as the conversion clock, executing a stop instruction aborts the current conversion and places the ADC in its Idle state. The contents of the ADC registers, including Rn, are unaffected by Normal Stop mode. After exiting from Normal Stop mode, a software or hardware trigger is required to resume conversions.

### 35.4.10.2 Normal Stop mode with ADACK enabled

If ADACK is selected as the conversion clock, the ADC continues operation during Normal Stop mode. See the chip configuration chapter for configuration information for this MCU.

If a conversion is in progress when the MCU enters Normal Stop mode, it continues until completion. Conversions can be initiated while the MCU is in Normal Stop mode by means of the hardware trigger or if continuous conversions are enabled.

If the compare and hardware averaging functions are disabled, a conversion complete event sets SC1n[COCO] and generates an ADC interrupt to wake the MCU from Normal Stop mode if the respective ADC interrupt is enabled, that is, when SC1n[AIEN]=1. The result register, Rn, will contain the data from the first completed conversion that occurred during Normal Stop mode. If the hardware averaging function is enabled, SC1n[COCO] will set, and generate an interrupt if enabled, when the selected number of conversions are completed. If the compare function is enabled, SC1n[COCO] will set, and generate an interrupt if enabled, only if the compare conditions are met. If a single conversion is selected and the compare is not true, the ADC will return to its Idle state and cannot wake the MCU from Normal Stop mode unless a new conversion is initiated by another hardware trigger.

### 35.4.11 MCU Low-Power Stop mode operation

The ADC module is automatically disabled when the MCU enters Low-Power Stop mode. All module registers contain their reset values following exit from Low-Power Stop mode. Therefore, the module must be re-enabled and re-configured following exit from Low-Power Stop mode.

**NOTE**

For the chip specific modes of operation, see the power management information for the device.

## 35.5 Initialization information

This section gives an example that provides some basic direction on how to initialize and configure the ADC module. The user can configure the module for 16-bit, 12-bit, 10-bit, or 8-bit single-ended resolution or 16-bit, 13-bit, 11-bit, or 9-bit differential resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. For information used in this example, refer to Table 35-107, Table 35-108, and Table 35-109.

**Note**

Hexadecimal values are designated by a preceding 0x, binary values designated by a preceding %, and decimal values have no preceding character.

### 35.5.1 ADC module initialization example

#### 35.5.1.1 Initialization sequence

Before the ADC module can be used to complete conversions, an initialization procedure must be performed. A typical sequence is:

1. Calibrate the ADC by following the calibration instructions in Calibration function.
2. Update CFG to select the input clock source and the divide ratio used to generate ADCK. This register is also used for selecting sample time and low-power configuration.
3. Update SC2 to select the conversion trigger, hardware or software, and compare function options, if enabled.
4. Update SC3 to select whether conversions will be continuous or completed only once (ADCO) and whether to perform hardware averaging.
5. Update SC1:SC1n registers to select whether conversions will be single-ended or differential and to enable or disable conversion complete interrupts. Also, select the input channel which can be used to perform conversions.

#### 35.5.1.2 Pseudo-code example

In this example, the ADC module is set up with interrupts enabled to perform a single 10-bit conversion at low-power with a long sample time on input channel 1, where ADCK is derived from the bus clock divided by 1.

**CFG1 = 0x98 (%10011000)**

```
        Bit 7   ADLPC   1       Configures for low power, lowers maximum clock speed.
        Bit 6:5 ADIV    00      Sets the ADCK to the input clock ÷ 1.
        Bit 4   ADLSMP  1       Configures for long sample time.
        Bit 3:2   MODE   10     Selects the single-ended 10-bit conversion, differential 11-
bit conversion.
        Bit 1:0   ADICLK   00    Selects the bus clock.
```

**SC2 = 0x00 (%00000000)**

```
        Bit 7   ADACT   0       Flag indicates if a conversion is in progress.
        Bit 6   ADTRG   0       Software trigger selected.
        Bit 5   ACFE    0       Compare function disabled.
        Bit 4   ACFGT   0       Not used in this example.
        Bit 3   ACREN   0       Compare range disabled.
 Bit 2 DMAEN 0 DMA request disabled.
        Bit 1:0 REFSEL  00      Selects default voltage reference pin pair (External pins VREFH
and VREFL).
```

**SC1A = 0x41 (%01000001)**

```
        Bit 7   COCO   0       Read-only flag which is set when a conversion completes.
        Bit 6   AIEN   1       Conversion complete interrupt enabled.
 Bit 5 DIFF 0 Single-ended conversion selected.
        Bit 4:0 ADCH   00001   Input channel 1 selected as ADC input channel.
```

### RA = 0xxx

```
        Holds results of conversion.
```

### CV = 0xxx

```
        Holds compare value when compare function enabled.
```

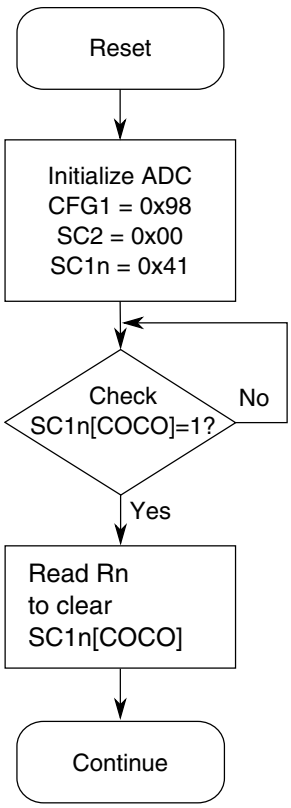



**Figure 35-94. Initialization flowchart example**

## 35.6 Application information

The ADC has been designed to be integrated into a microcontroller for use in embedded control applications requiring an ADC.

### 35.6.1 External pins and routing

#### 35.6.1.1 Analog supply pins

Depending on the device, the analog power and ground supplies, $V_{DDA}$ and $V_{SSA}$, of the ADC module are available as:

- $V_{DDA}$ and $V_{SSA}$ available as separate pins—When available on a separate pin, both $V_{DDA}$ and $V_{SSA}$ must be connected to the same voltage potential as their corresponding MCU digital supply, $V_{DD}$ and $V_{SS}$, and must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.
- $V_{SSA}$ is shared on the same pin as the MCU digital $V_{SS}$.
- $V_{SSA}$ and $V_{DDA}$ are shared with the MCU digital supply pins—In these cases, there are separate pads for the analog supplies bonded to the same pin as the corresponding digital supply so that some degree of isolation between the supplies is maintained.

If separate power supplies are used for analog and digital power, the ground connection between these supplies must be at the $V_{SSA}$ pin. This must be the only ground connection between these supplies, if possible. $V_{SSA}$ makes a good single point ground location.

#### 35.6.1.2 Analog voltage reference pins

In addition to the analog supplies, the ADC module has connections for two reference voltage inputs used by the converter:

- $V_{REFSH}$ is the high reference voltage for the converter.
- $V_{REFSL}$ is the low reference voltage for the converter.

The ADC can be configured to accept one of two voltage reference pairs for $V_{REFSH}$ and $V_{REFSL}$. Each pair contains a positive reference and a ground reference. The two pairs are external, $V_{REFH}$ and $V_{REFL}$ and alternate, $V_{ALTH}$ and $V_{ALTL}$. These voltage references are selected using SC2[REFSEL]. The alternate voltage reference pair, $V_{ALTH}$ and $V_{ALTL}$, may select additional external pins or internal sources based on MCU configuration. See the chip configuration information on the voltage references specific to this MCU.

In some packages, the external or alternate pairs are connected in the package to $V_{DDA}$ and $V_{SSA}$, respectively. One of these positive references may be shared on the same pin as $V_{DDA}$ on some devices. One of these ground references may be shared on the same pin as $V_{SSA}$ on some devices.

If externally available, the positive reference may be connected to the same potential as $V_{DDA}$ or may be driven by an external source to a level between the minimum Ref Voltage High and the $V_{DDA}$ potential. The positive reference must never exceed $V_{DDA}$. If externally available, the ground reference must be connected to the same voltage potential as $V_{SSA}$. The voltage reference pairs must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

AC current in the form of current spikes required to supply charge to the capacitor array at each successive approximation step is drawn through the $V_{REFH}$ and $V_{REFL}$ loop. The best external component to meet this current demand is a 0.1 µF capacitor with good high-frequency characteristics. This capacitor is connected between $V_{REFH}$ and $V_{REFL}$ and must be placed as near as possible to the package pins. Resistance in the path is not recommended because the current causes a voltage drop that could result in conversion errors. Inductance in this path must be minimum, that is, parasitic only.

#### 35.6.1.3 Analog input pins

The external analog inputs are typically shared with digital I/O pins on MCU devices.

Empirical data shows that capacitors on the analog inputs improve performance in the presence of noise or when the source impedance is high. Use of 0.01 µF capacitors with good high-frequency characteristics is sufficient. These capacitors are not necessary in all cases, but when used, they must be placed as near as possible to the package pins and be referenced to $V_{SSA}$.

For proper conversion, the input voltage must fall between $V_{REFH}$ and $V_{REFL}$. If the input is equal to or exceeds $V_{REFH}$, the converter circuit converts the signal to 0xFFF, which is full scale 12-bit representation, 0x3FF, which is full scale 10-bit representation, or 0xFF, which is full scale 8-bit representation. If the input is equal to or less than $V_{REFL}$, the converter circuit converts it to 0x000. Input voltages between $V_{REFH}$ and $V_{REFL}$ are straight-line linear conversions. There is a brief current associated with $V_{REFL}$ when the sampling capacitor is charging.

For minimal loss of accuracy due to current injection, pins adjacent to the analog input pins must not be transitioning during conversions.

### 35.6.2 Sources of error

#### 35.6.2.1 Sampling error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy.

$$RAS + RADIN = SC / (FMAX * NUMTAU * CADIN)$$

**Figure 35-95. Sampling equation**

Where:

RAS = External analog source resistance

SC = Number of ADCK cycles used during sample window

CADIN = Internal ADC input capacitance

NUMTAU = -ln(LSBERR / $2^N$)

LSBERR = value of acceptable sampling error in LSBs

N = 8 in 8-bit mode, 10 in 10-bit mode, 12 in 12-bit mode or 16 in 16-bit mode

Higher source resistances or higher-accuracy sampling is possible by setting CFG1[ADLSMP] and changing CFG2[ADLSTS] to increase the sample window, or decreasing ADCK frequency to increase sample time.

### 35.6.2.2 Pin leakage error

Leakage on the I/O pins can cause conversion error if the external analog source resistance, $R_{AS}$, is high. If this error cannot be tolerated by the application, keep $R_{AS}$ lower than $V_{REFH}$ / (4 × $I_{LEAK}$ × $2^N$) for less than 1/4 LSB leakage error, where N = 8 in 8-bit mode, 10 in 10-bit mode, 12 in 12-bit mode, or 16 in 16-bit mode.

### 35.6.2.3 Noise-induced errors

System noise that occurs during the sample or conversion process can affect the accuracy of the conversion. The ADC accuracy numbers are guaranteed as specified only if the following conditions are met:

- There is a 0.1 µF low-ESR capacitor from $V_{REFH}$ to $V_{REFL}$.
- There is a 0.1 µF low-ESR capacitor from $V_{DDA}$ to $V_{SSA}$.
- If inductive isolation is used from the primary supply, an additional 1 µF capacitor is placed from $V_{DDA}$ to $V_{SSA}$.
- $V_{SSA}$, and $V_{REFL}$, if connected, is connected to $V_{SS}$ at a quiet point in the ground plane.
- Operate the MCU in Wait or Normal Stop mode before initiating (hardware-triggered conversions) or immediately after initiating (hardware- or software-triggered conversions) the ADC conversion.

- For software triggered conversions, immediately follow the write to SC1 with a Wait instruction or Stop instruction.
- For Normal Stop mode operation, select ADACK as the clock source. Operation in Normal Stop reduces $V_{DD}$ noise but increases effective conversion time due to stop recovery.

- There is no I/O switching, input or output, on the MCU during the conversion.

There are some situations where external system activity causes radiated or conducted noise emissions or excessive $V_{DD}$ noise is coupled into the ADC. In these situations, or when the MCU cannot be placed in Wait or Normal Stop mode, or I/O activity cannot be halted, the following actions may reduce the effect of noise on the accuracy:

- Place a 0.01 µF capacitor ($C_{AS}$) on the selected input channel to $V_{REFL}$ or $V_{SSA}$. This improves noise issues, but affects the sample rate based on the external analog source resistance.
- Average the result by converting the analog input many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1 LSB, one-time error.
- Reduce the effect of synchronous noise by operating off the asynchronous clock, that is, ADACK, and averaging. Noise that is synchronous to ADCK cannot be averaged out.

### 35.6.2.4 Code width and quantization error

The ADC quantizes the ideal straight-line transfer function into 65536 steps in the 16-bit mode). Each step ideally has the same height, that is, 1 code, and width. The width is defined as the delta between the transition points to one code and the next. The ideal code width for an N-bit converter, where N can be 16, 12, 10, or 8, defined as 1 LSB, is:

$$1LSB=\left( V_{REFH}\right)/2^{N}$$

**Figure 35-96. Ideal code width for an N-bit converter**

There is an inherent quantization error due to the digitization of the result. For 8-bit, 10-bit, or 12-bit conversions, the code transitions when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be ± 1/2 LSB in 8-bit, 10-bit, or 12-bit modes. As a consequence, however, the code width of the first (0x000) conversion is only 1/2 LSB and the code width of the last (0xFF or 0x3FF) is 1.5 LSB.

For 16-bit conversions, the code transitions only after the full code width is present, so the quantization error is -1 LSB to 0 LSB and the code width of each step is 1 LSB.

### 35.6.2.5 Linearity errors

The ADC may also exhibit non-linearity of several forms. Every effort has been made to reduce these errors, but the system designers must be aware of these errors because they affect overall accuracy:

- Zero-scale error ($E_{ZS}$), sometimes called offset: This error is defined as the difference between the actual code width of the first conversion and the ideal code width. This is 1/2 LSB in 8-bit, 10-bit, or 12-bit modes and 1 LSB in 16-bit mode. If the first conversion is 0x001, the difference between the actual 0x001 code width and its ideal (1 LSB) is used.
- Full-scale error ($E_{FS}$): This error is defined as the difference between the actual code width of the last conversion and the ideal code width. This is 1.5 LSB in 8-bit, 10-bit, or 12-bit modes and 1 LSB in 16-bit mode. If the last conversion is 0x3FE, the difference between the actual 0x3FE code width and its ideal (1 LSB) is used.
- Differential non-linearity (DNL): This error is defined as the worst-case difference between the actual code width and the ideal code width for all conversions.
- Integral non-linearity (INL): This error is defined as the highest-value or absolute value that the running sum of DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.
- Total unadjusted error (TUE): This error is defined as the difference between the actual transfer function and the ideal straight-line transfer function and includes all forms of error.

### 35.6.2.6 Code jitter, non-monotonicity, and missing codes

Analog-to-digital converters are susceptible to three special forms of error:

- Code jitter: Code jitter is when, at certain points, a given input voltage converts to one of the two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the converter yields the lower code, and vice-versa. However, even small amounts of system noise can cause the converter to be indeterminate, between two codes, for a range of input voltages around the transition voltage.

This error may be reduced by repeatedly sampling the input and averaging the result. Additionally, the techniques discussed in Noise-induced errors reduces this error.

- Non-monotonicity: Non-monotonicity is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage.
- Missing codes: Missing codes are those values never converted for any input value.

In 8-bit or 10-bit mode, the ADC is guaranteed to be monotonic and have no missing codes.

# Chapter 36
# Comparator (CMP)

## 36.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The comparator (CMP) module provides a circuit for comparing two analog input voltages. The comparator circuit is designed to operate across the full range of the supply voltage, known as rail-to-rail operation.

The Analog MUX (ANMUX) provides a circuit for selecting an analog input signal from eight channels. One signal is provided by the 6-bit digital-to-analog converter (DAC). The mux circuit is designed to operate across the full range of the supply voltage.

The 6-bit DAC is 64-tap resistor ladder network which provides a selectable voltage reference for applications where voltage reference is needed. The 64-tap resistor ladder network divides the supply reference $V_{in}$ into 64 voltage levels. A 6-bit digital signal input selects the output voltage level, which varies from $V_{in}$ to $V_{in}/64$. $V_{in}$ can be selected from two voltage sources, $V_{in1}$ and $V_{in2}$. The 6-bit DAC from a comparator is available as an on-chip internal signal only and is not available externally to a pin.

## 36.2 CMP features

The CMP has the following features:

- Operational over the entire supply range
- Inputs may range from rail to rail
- Programmable hysteresis control

- Selectable interrupt on rising-edge, falling-edge, or both rising or falling edges of the comparator output
- Selectable inversion on comparator output
- Capability to produce a wide range of outputs such as:
  - Sampled
  - Windowed, which is ideal for certain PWM zero-crossing-detection applications
  - Digitally filtered:
    - Filter can be bypassed
    - Can be clocked via external SAMPLE signal or scaled bus clock
- External hysteresis can be used at the same time that the output filter is used for internal functions
- Two software selectable performance levels:
  - Shorter propagation delay at the expense of higher power
  - Low power, with longer propagation delay
- DMA transfer support
  - A comparison event can be selected to trigger a DMA transfer
- Functional in all modes of operation
- The window and filter functions are not available in the following modes:
  - Stop
  - VLPS
  - LLS
  - VLLSx

## 36.3 6-bit DAC key features

- 6-bit resolution
- Selectable supply reference source
- Power Down mode to conserve power when not in use
- Option to route the output to internal comparator input

## 36.4 ANMUX key features

- Two 8-to-1 channel mux
- Operational over the entire supply range

## 36.5 CMP, DAC and ANMUX diagram

The following figure shows the block diagram for the High-Speed Comparator, DAC, and ANMUX modules.

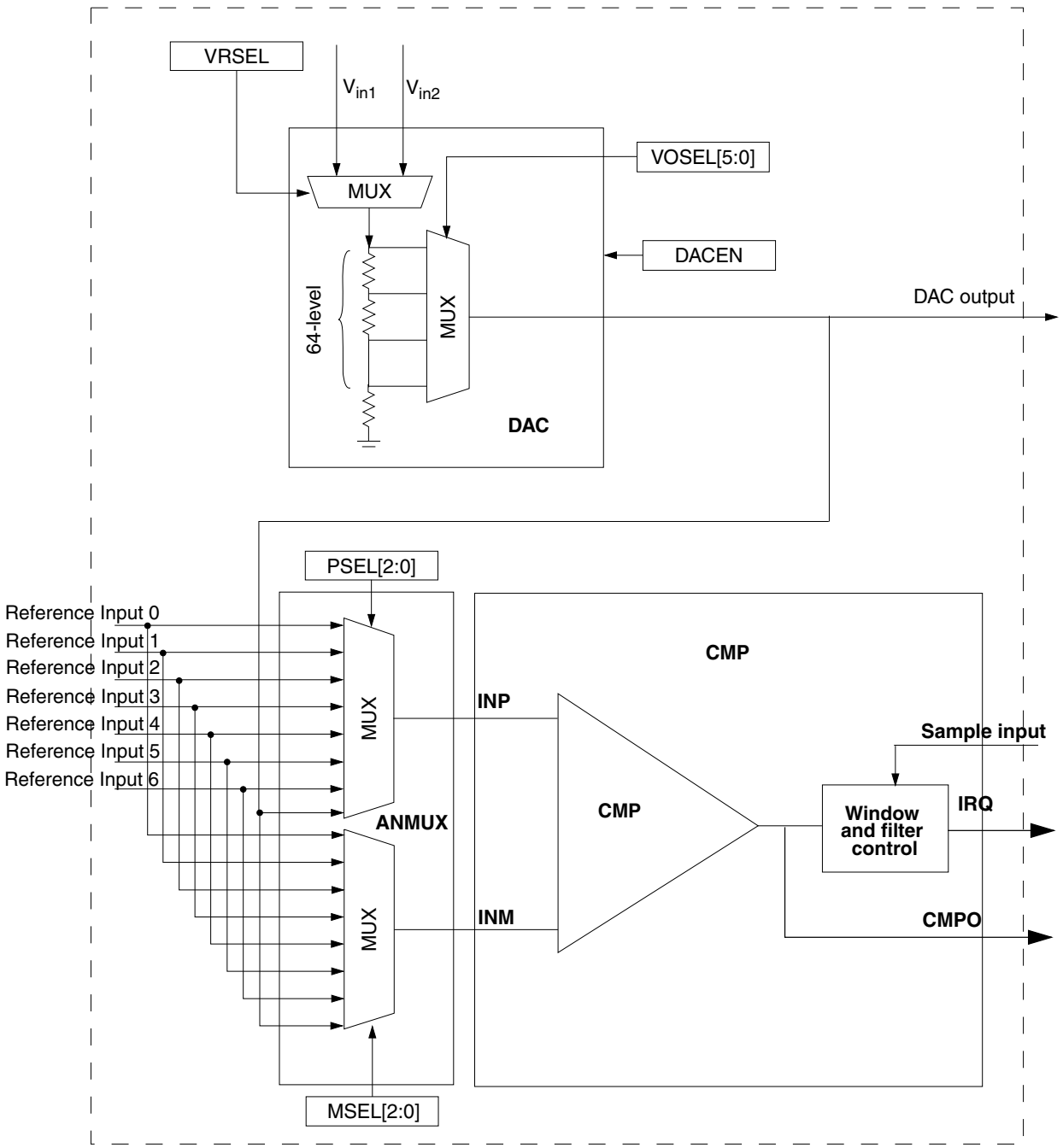



**Figure 36-1. CMP, DAC and ANMUX block diagram**

## 36.6 CMP block diagram

The following figure shows the block diagram for the CMP module.

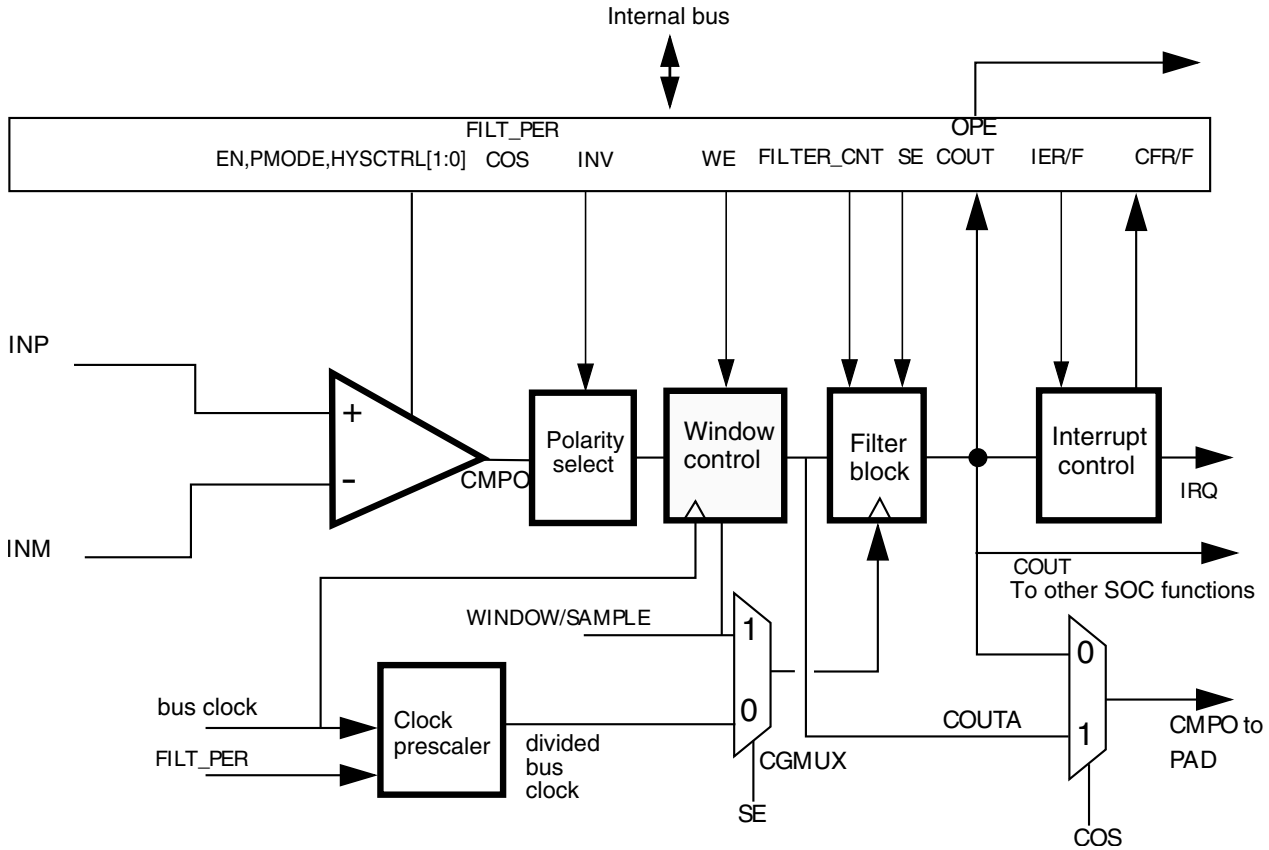



**Figure 36-2. Comparator module block diagram**

In the CMP block diagram:

- The Window Control block is bypassed when CR1[WE] = 0
- If CR1[WE] = 1, the comparator output will be sampled on every bus clock when WINDOW=1 to generate COUTA. Sampling does NOT occur when WINDOW = 0.
- The Filter block is bypassed when not in use.
- The Filter block acts as a simple sampler if the filter is bypassed and CR0[FILTER_CNT] is set to 0x01.
- The Filter block filters based on multiple samples when the filter is bypassed and CR0[FILTER_CNT] is set greater than 0x01.
  - If CR1[SE] = 1, the external SAMPLE input is used as sampling clock
  - IF CR1[SE] = 0, the divided bus clock is used as sampling clock

- If enabled, the Filter block will incur up to one bus clock additional latency penalty on COUT due to the fact that COUT, which is crossing clock domain boundaries, must be resynchronized to the bus clock.
- CR1[WE] and CR1[SE] are mutually exclusive.

## 36.7 Memory map/register definitions

**CMP memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4007_3000 | CMP Control Register 0 (CMP0_CR0) | 8 | R/W | 00h | 36.7.1/848 |
| 4007_3001 | CMP Control Register 1 (CMP0_CR1) | 8 | R/W | 00h | 36.7.2/849 |
| 4007_3002 | CMP Filter Period Register (CMP0_FPR) | 8 | R/W | 00h | 36.7.3/851 |
| 4007_3003 | CMP Status and Control Register (CMP0_SCR) | 8 | R/W | 00h | 36.7.4/851 |
| 4007_3004 | DAC Control Register (CMP0_DACCR) | 8 | R/W | 00h | 36.7.5/852 |
| 4007_3005 | MUX Control Register (CMP0_MUXCR) | 8 | R/W | 00h | 36.7.6/853 |
| 4007_3008 | CMP Control Register 0 (CMP1_CR0) | 8 | R/W | 00h | 36.7.1/848 |
| 4007_3009 | CMP Control Register 1 (CMP1_CR1) | 8 | R/W | 00h | 36.7.2/849 |
| 4007_300A | CMP Filter Period Register (CMP1_FPR) | 8 | R/W | 00h | 36.7.3/851 |
| 4007_300B | CMP Status and Control Register (CMP1_SCR) | 8 | R/W | 00h | 36.7.4/851 |
| 4007_300C | DAC Control Register (CMP1_DACCR) | 8 | R/W | 00h | 36.7.5/852 |
| 4007_300D | MUX Control Register (CMP1_MUXCR) | 8 | R/W | 00h | 36.7.6/853 |
| 4007_3010 | CMP Control Register 0 (CMP2_CR0) | 8 | R/W | 00h | 36.7.1/848 |
| 4007_3011 | CMP Control Register 1 (CMP2_CR1) | 8 | R/W | 00h | 36.7.2/849 |
| 4007_3012 | CMP Filter Period Register (CMP2_FPR) | 8 | R/W | 00h | 36.7.3/851 |
| 4007_3013 | CMP Status and Control Register (CMP2_SCR) | 8 | R/W | 00h | 36.7.4/851 |
| 4007_3014 | DAC Control Register (CMP2_DACCR) | 8 | R/W | 00h | 36.7.5/852 |
| 4007_3015 | MUX Control Register (CMP2_MUXCR) | 8 | R/W | 00h | 36.7.6/853 |

### 36.7.1 CMP Control Register 0 (CMPx_CR0)

Address: Base address + 0h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | FILTER_CNT | | | 0 | 0 | HYSTCTR | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMPx_CR0 field descriptions**

| Field | Description |
|---|---|
| 7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6–4<br>FILTER_CNT | Filter Sample Count<br><br>Represents the number of consecutive samples that must agree prior to the comparator ouput filter accepting a new output state. For information regarding filter programming and latency, see the Functional description.<br><br>000 Filter is disabled. If SE = 1, then COUT is a logic 0. This is not a legal state, and is not recommended. If SE = 0, COUT = COUTA.<br>001 One sample must agree. The comparator output is simply sampled.<br>010 2 consecutive samples must agree.<br>011 3 consecutive samples must agree.<br>100 4 consecutive samples must agree.<br>101 5 consecutive samples must agree.<br>110 6 consecutive samples must agree.<br>111 7 consecutive samples must agree. |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 1–0<br>HYSTCTR | Comparator hard block hysteresis control<br><br>Defines the programmable hysteresis level. The hysteresis values associated with each level are device-specific. See the Data Sheet of the device for the exact values.<br><br>00 Level 0<br>01 Level 1<br>10 Level 2<br>11 Level 3 |

## 36.7.2 CMP Control Register 1 (CMPx_CR1)

Address: Base address + 1h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | SE | WE | 0 | PMODE | INV | COS | OPE | EN |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMPx_CR1 field descriptions**

| Field | Description |
|---|---|
| 7<br>SE | Sample Enable<br><br>At any given time, either SE or WE can be set. If a write to this register attempts to set both, then SE is set and WE is cleared. However, avoid writing 1s to both field locations because this "11" case is reserved and may change in future implementations. |

*Table continues on the next page...*

## CMPx_CR1 field descriptions (continued)

| Field | Description |
|---|---|
| | 0 Sampling mode is not selected.<br>1 Sampling mode is selected. |
| 6<br>WE | Windowing Enable<br><br>At any given time, either SE or WE can be set. If a write to this register attempts to set both, then SE is set and WE is cleared. However, avoid writing 1s to both field locations because this "11" case is reserved and may change in future implementations.<br><br>0 Windowing mode is not selected.<br>1 Windowing mode is selected. |
| 5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>PMODE | Power Mode Select<br><br>See the electrical specifications table in the device Data Sheet for details.<br><br>0 Low-Speed (LS) Comparison mode selected. In this mode, CMP has slower output propagation delay and lower current consumption.<br>1 High-Speed (HS) Comparison mode selected. In this mode, CMP has faster output propagation delay and higher current consumption. |
| 3<br>INV | Comparator INVERT<br><br>Allows selection of the polarity of the analog comparator function. It is also driven to the COUT output, on both the device pin and as SCR[COUT], when OPE=0.<br><br>0 Does not invert the comparator output.<br>1 Inverts the comparator output. |
| 2<br>COS | Comparator Output Select<br><br>0 Set the filtered comparator output (CMPO) to equal COUT.<br>1 Set the unfiltered comparator output (CMPO) to equal COUTA. |
| 1<br>OPE | Comparator Output Pin Enable<br><br>0 CMPO is not available on the associated CMPO output pin. If the comparator does not own the pin, this field has no effect.<br>1 CMPO is available on the associated CMPO output pin.<br><br>The comparator output (CMPO) is driven out on the associated CMPO output pin if the comparator owns the pin. If the comparator does not own the field, this bit has no effect. |
| 0<br>EN | Comparator Module Enable<br><br>Enables the Analog Comparator module. When the module is not enabled, it remains in the off state, and consumes no power. When the user selects the same input from analog mux to the positive and negative port, the comparator is disabled automatically.<br><br>0 Analog Comparator is disabled.<br>1 Analog Comparator is enabled. |

## 36.7.3 CMP Filter Period Register (CMPx_FPR)

Address: Base address + 2h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | FILT_PER | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### CMPx_FPR field descriptions

| Field | Description |
|---|---|
| 7–0<br>FILT_PER | Filter Sample Period<br><br>Specifies the sampling period, in bus clock cycles, of the comparator output filter, when CR1[SE]=0. Setting FILT_PER to 0x0 disables the filter. Filter programming and latency details appear in the Functional description.<br><br>This field has no effect when CR1[SE]=1. In that case, the external SAMPLE signal is used to determine the sampling period. |

## 36.7.4 CMP Status and Control Register (CMPx_SCR)

Address: Base address + 3h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | DMAEN | 0 | IER | IEF | CFR | CFF | COUT |
| Write | | | | | | w1c | w1c | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### CMPx_SCR field descriptions

| Field | Description |
|---|---|
| 7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6<br>DMAEN | DMA Enable Control<br><br>Enables the DMA transfer triggered from the CMP module. When this field is set, a DMA request is asserted when CFR or CFF is set.<br><br>0 DMA is disabled.<br>1 DMA is enabled. |
| 5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>IER | Comparator Interrupt Enable Rising<br><br>Enables the CFR interrupt from the CMP. When this field is set, an interrupt will be asserted when CFR is set. |

*Table continues on the next page...*

**CMPx_SCR field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Interrupt is disabled.<br>1 Interrupt is enabled. |
| 3<br>IEF | Comparator Interrupt Enable Falling<br><br>Enables the CFF interrupt from the CMP. When this field is set, an interrupt will be asserted when CFF is set.<br><br>0 Interrupt is disabled.<br>1 Interrupt is enabled. |
| 2<br>CFR | Analog Comparator Flag Rising<br><br>Detects a rising-edge on COUT, when set, during normal operation. CFR is cleared by writing 1 to it. During Stop modes, CFR is level sensitive .<br><br>0 Rising-edge on COUT has not been detected.<br>1 Rising-edge on COUT has occurred. |
| 1<br>CFF | Analog Comparator Flag Falling<br><br>Detects a falling-edge on COUT, when set, during normal operation. CFF is cleared by writing 1 to it. During Stop modes, CFF is level sensitive .<br><br>0 Falling-edge on COUT has not been detected.<br>1 Falling-edge on COUT has occurred. |
| 0<br>COUT | Analog Comparator Output<br><br>Returns the current value of the Analog Comparator output, when read. The field is reset to 0 and will read as CR1[INV] when the Analog Comparator module is disabled, that is, when CR1[EN] = 0. Writes to this field are ignored. |

## 36.7.5 DAC Control Register (CMPx_DACCR)

Address: Base address + 4h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | DACEN | VRSEL | VOSEL | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMPx_DACCR field descriptions**

| Field | Description |
|---|---|
| 7<br>DACEN | DAC Enable<br><br>Enables the DAC. When the DAC is disabled, it is powered down to conserve power.<br><br>0 DAC is disabled.<br>1 DAC is enabled. |
| 6<br>VRSEL | Supply Voltage Reference Source Select |

*Table continues on the next page...*

**CMPx_DACCR field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 V is selected as resistor ladder network supply reference V. $_{in1in}$<br>1 V is selected as resistor ladder network supply reference V. $_{in2in}$ |
| 5–0<br>VOSEL | DAC Output Voltage Select<br><br>Selects an output voltage from one of 64 distinct levels.<br><br>`DACO = (V` $_{in}$ `/64) * (VOSEL[5:0] + 1)` , so the DACO range is from V $_{in}$ /64 to V $_{in}$ . |

## 36.7.6 MUX Control Register (CMPx_MUXCR)

Address: Base address + 5h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | PSTM | 0 | PSEL | | | MSEL | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMPx_MUXCR field descriptions**

| Field | Description |
|---|---|
| 7<br>PSTM | Pass Through Mode Enable<br><br>This bit is used to enable to MUX pass through mode. Pass through mode is always available but for some devices this feature must be always disabled due to the lack of package pins.<br><br>0 Pass Through Mode is disabled.<br>1 Pass Through Mode is enabled. |
| 6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5–3<br>PSEL | Plus Input Mux Control<br><br>Determines which input is selected for the plus input of the comparator. For INx inputs, see CMP, DAC, and ANMUX block diagrams.<br><br>**NOTE:** When an inappropriate operation selects the same input for both muxes, the comparator automatically shuts down to prevent itself from becoming a noise generator.<br><br>000 IN0<br>001 IN1<br>010 IN2<br>011 IN3<br>100 IN4<br>101 IN5<br>110 IN6<br>111 IN7 |
| 2–0<br>MSEL | Minus Input Mux Control<br><br>Determines which input is selected for the minus input of the comparator. For INx inputs, see CMP, DAC, and ANMUX block diagrams. |

*Table continues on the next page...*

CMPx_MUXCR field descriptions (continued)

| Field | Description |
|---|---|
| | **NOTE:** When an inappropriate operation selects the same input for both muxes, the comparator automatically shuts down to prevent itself from becoming a noise generator.<br><br>000 IN0<br>001 IN1<br>010 IN2<br>011 IN3<br>100 IN4<br>101 IN5<br>110 IN6<br>111 IN7 |

## 36.8 Functional description

The CMP module can be used to compare two analog input voltages applied to INP and INM. CMPO is high when the non-inverting input is greater than the inverting input, and is low when the non-inverting input is less than the inverting input. This signal can be selectively inverted by setting CR1[INV] = 1.

SCR[IER] and SCR[IEF] are used to select the condition which will cause the CMP module to assert an interrupt to the processor. SCR[CFF] is set on a falling-edge and SCR[CFR] is set on rising-edge of the comparator output. The optionally filtered CMPO can be read directly through SCR[COUT].

### 36.8.1 CMP functional modes

There are three main sub-blocks to the CMP module:

- The comparator itself
- The window function
- The filter function

The filter, CR0[FILTER_CNT], can be clocked from an internal or external clock source. The filter is programmable with respect to the number of samples that must agree before a change in the output is registered. In the simplest case, only one sample must agree. In this case, the filter acts as a simple sampler.

The external sample input is enabled using CR1[SE]. When set, the output of the comparator is sampled only on rising edges of the sample input.

The "windowing mode" is enabled by setting CR1[WE]. When set, the comparator output is sampled only when WINDOW=1. 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 comparator filter and sampling features can be combined as shown in the following table. Individual modes are discussed below.

**Table 36-29. Comparator sample/filter controls**

| Mode # | CR1[EN] | CR1[WE] | CR1[SE] | CR0[FILTER_CNT] | FPR[FILT_PER] | Operation |
|---|---|---|---|---|---|---|
| 1 | 0 | X | X | X | X | **Disabled**<br>See the Disabled mode (# 1). |
| 2A | 1 | 0 | 0 | 0x00 | X | **Continuous Mode**<br>See the Continuous mode (#s 2A & 2B). |
| 2B | 1 | 0 | 0 | X | 0x00 | |
| 3A | 1 | 0 | 1 | 0x01 | X | **Sampled, Non-Filtered mode**<br>See the Sampled, Non-Filtered mode (#s 3A & 3B). |
| 3B | 1 | 0 | 0 | 0x01 | > 0x00 | |
| 4A | 1 | 0 | 1 | > 0x01 | X | **Sampled, Filtered mode**<br>See the Sampled, Filtered mode (#s 4A & 4B). |
| 4B | 1 | 0 | 0 | > 0x01 | > 0x00 | |
| 5A | 1 | 1 | 0 | 0x00 | X | **Windowed mode**<br>Comparator output is sampled on every rising bus clock edge when SAMPLE=1 to generate COUTA.<br>See the Windowed mode (#s 5A & 5B). |
| 5B | 1 | 1 | 0 | X | 0x00 | |
| 6 | 1 | 1 | 0 | 0x01 | 0x01–0xFF | **Windowed/Resampled mode**<br>Comparator output is sampled on every rising bus clock edge when SAMPLE=1 to generate COUTA, which is then resampled on an interval determined by FILT_PER to generate COUT.<br>See the Windowed/Resampled mode (# 6). |
| 7 | 1 | 1 | 0 | > 0x01 | 0x01–0xFF | **Windowed/Filtered mode**<br>Comparator output is sampled on every rising bus clock edge when SAMPLE=1 to generate COUTA, which is then resampled and filtered to generate COUT.<br>See the Windowed/Filtered mode (#7). |
| All other combinations of CR1[EN], CR1[WE], CR1[SE], CR0[FILTER_CNT], and FPR[FILT_PER] are illegal. | | | | | | |

For cases where a comparator is used to drive a fault input, for example, for a motor-control module such as FTM, it must be configured to operate in Continuous mode so that an external fault can immediately pass through the comparator to the target fault circuitry.

**Note**

Filtering and sampling settings must be changed only after setting CR1[SE]=0 and CR0[FILTER_CNT]=0x00. This resets the filter to a known state.

### 36.8.1.1 Disabled mode (# 1)

In Disabled mode, the analog comparator is non-functional and consumes no power. CMPO is 0 in this mode.

### 36.8.1.2 Continuous mode (#s 2A & 2B)

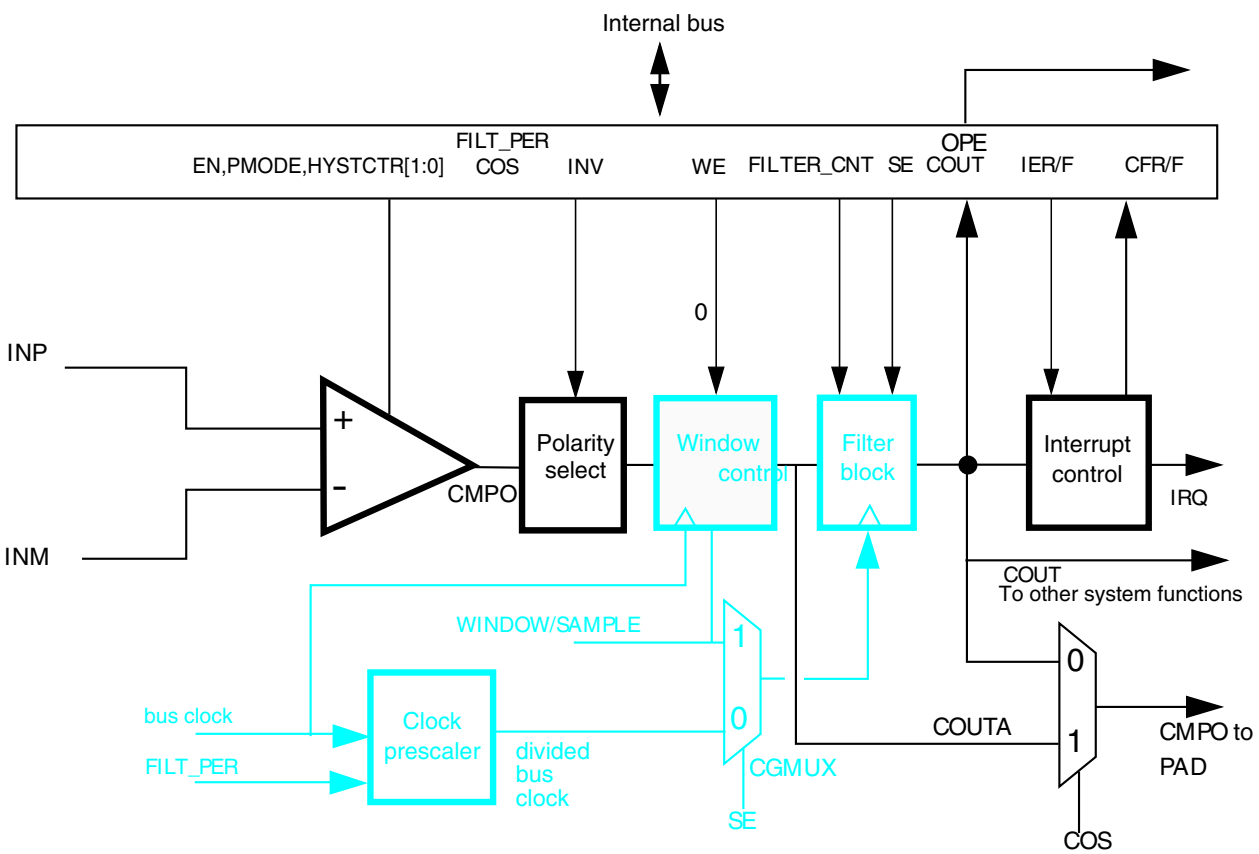



**Figure 36-27. Comparator operation in Continuous mode**

**NOTE**

See the chip configuration section for the source of sample/ window input.

The analog comparator block is powered and active. CMPO may be optionally inverted, but is not subject to external sampling or filtering. Both window control and filter blocks are completely bypassed. SCR[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 control configurations which result in disabling the filter block, see the Filter Block Bypass Logic diagram.

### 36.8.1.3 Sampled, Non-Filtered mode (#s 3A & 3B)

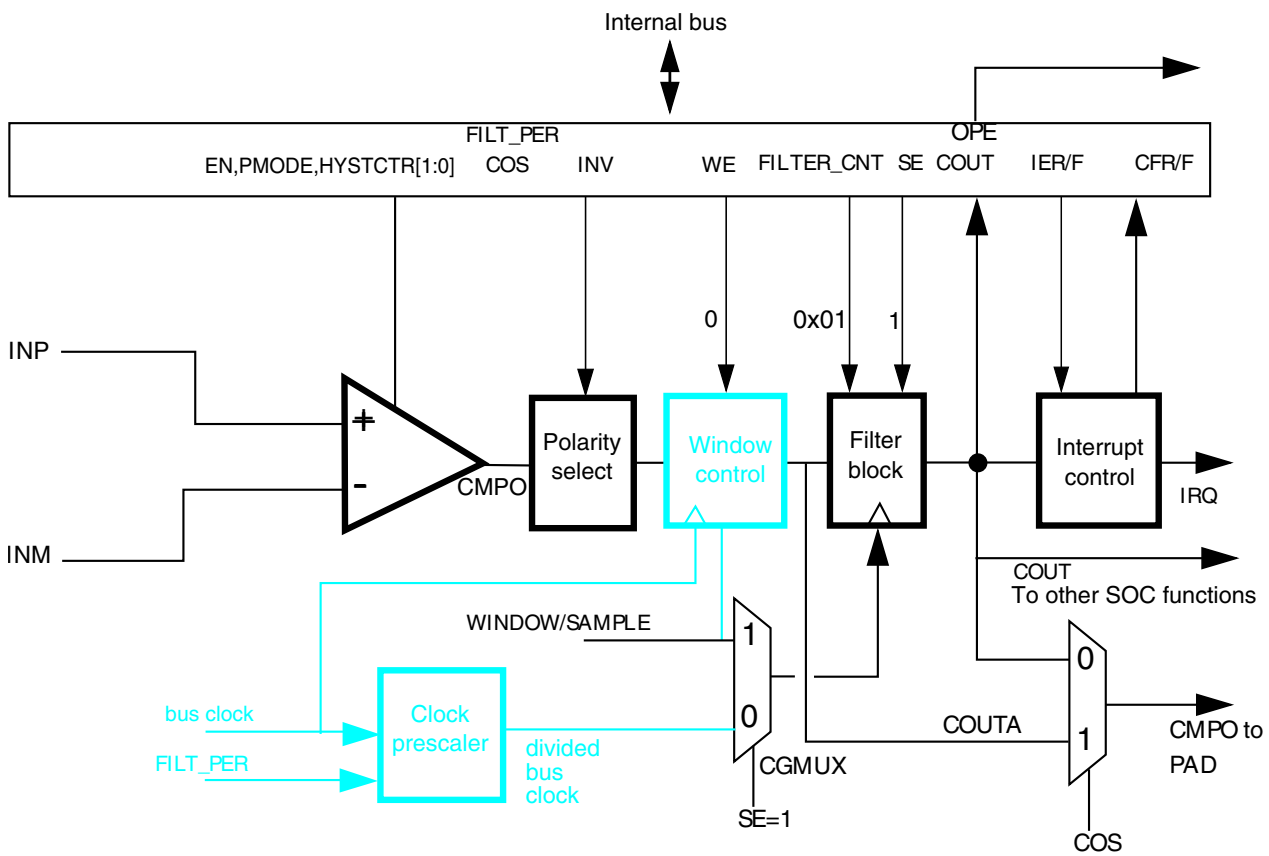



**Figure 36-28. Sampled, Non-Filtered (# 3A): sampling point externally driven**

In Sampled, Non-Filtered mode, the analog comparator block is powered and active. 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 filter block clock input.

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 comparator filter has no other function than sample/hold of the comparator output in this mode (# 3B).

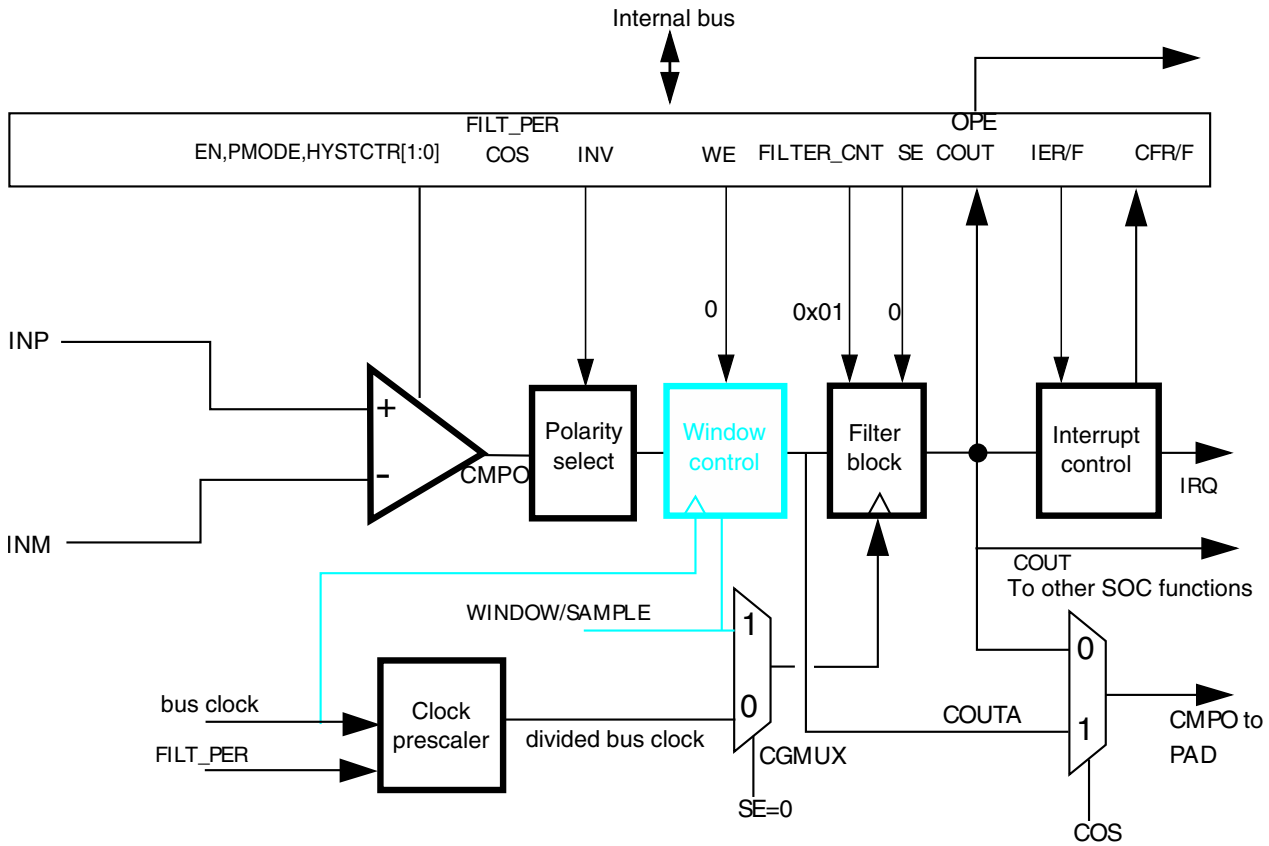



**Figure 36-29. Sampled, Non-Filtered (# 3B): sampling interval internally derived**

## 36.8.1.4 Sampled, Filtered mode (#s 4A & 4B)

In Sampled, Filtered mode, the analog comparator block is powered and active. 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 filter block clock input.

The only difference in operation between Sampled, Non-Filtered (# 3A) and Sampled, Filtered (# 4A) is that, now, CR0[FILTER_CNT]>1, which activates filter operation.

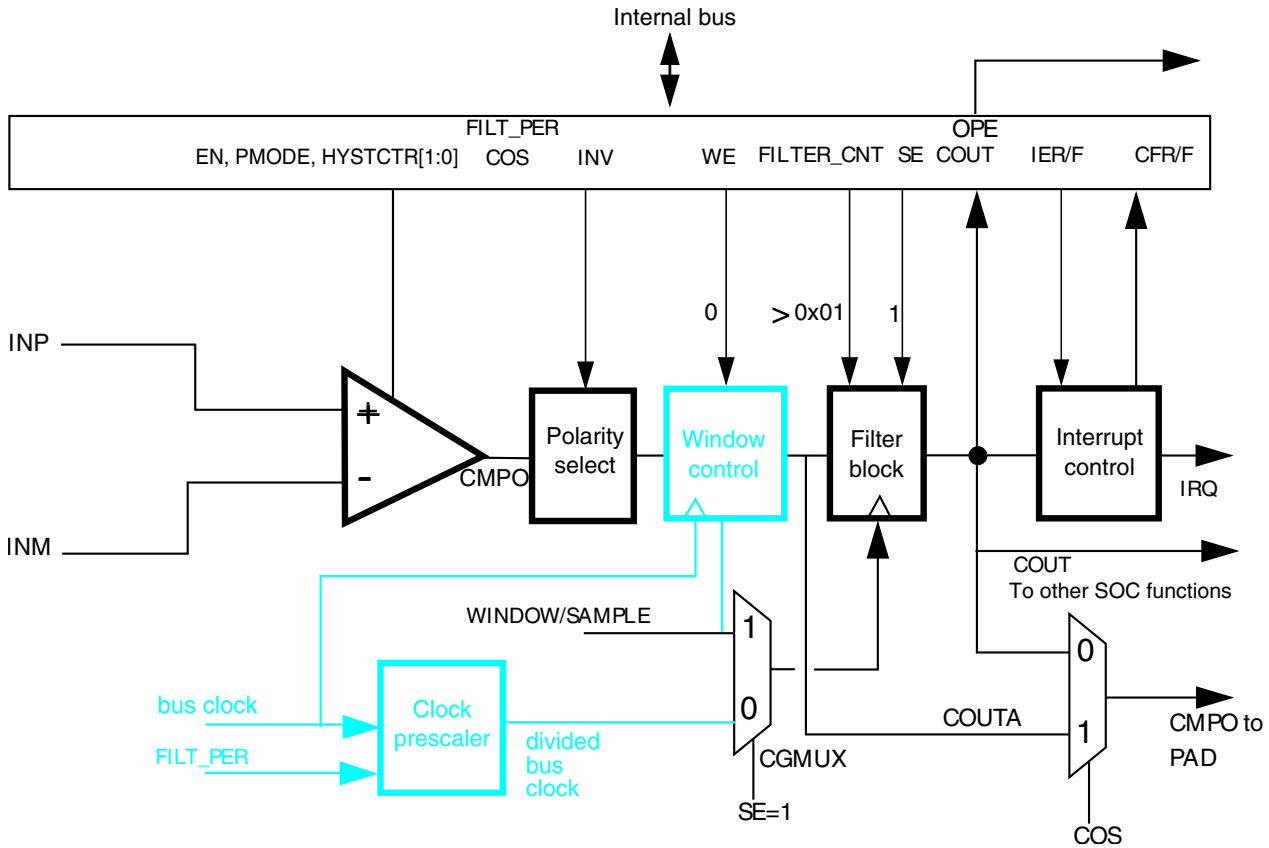



**Figure 36-30. Sampled, Filtered (# 4A): sampling point externally driven**

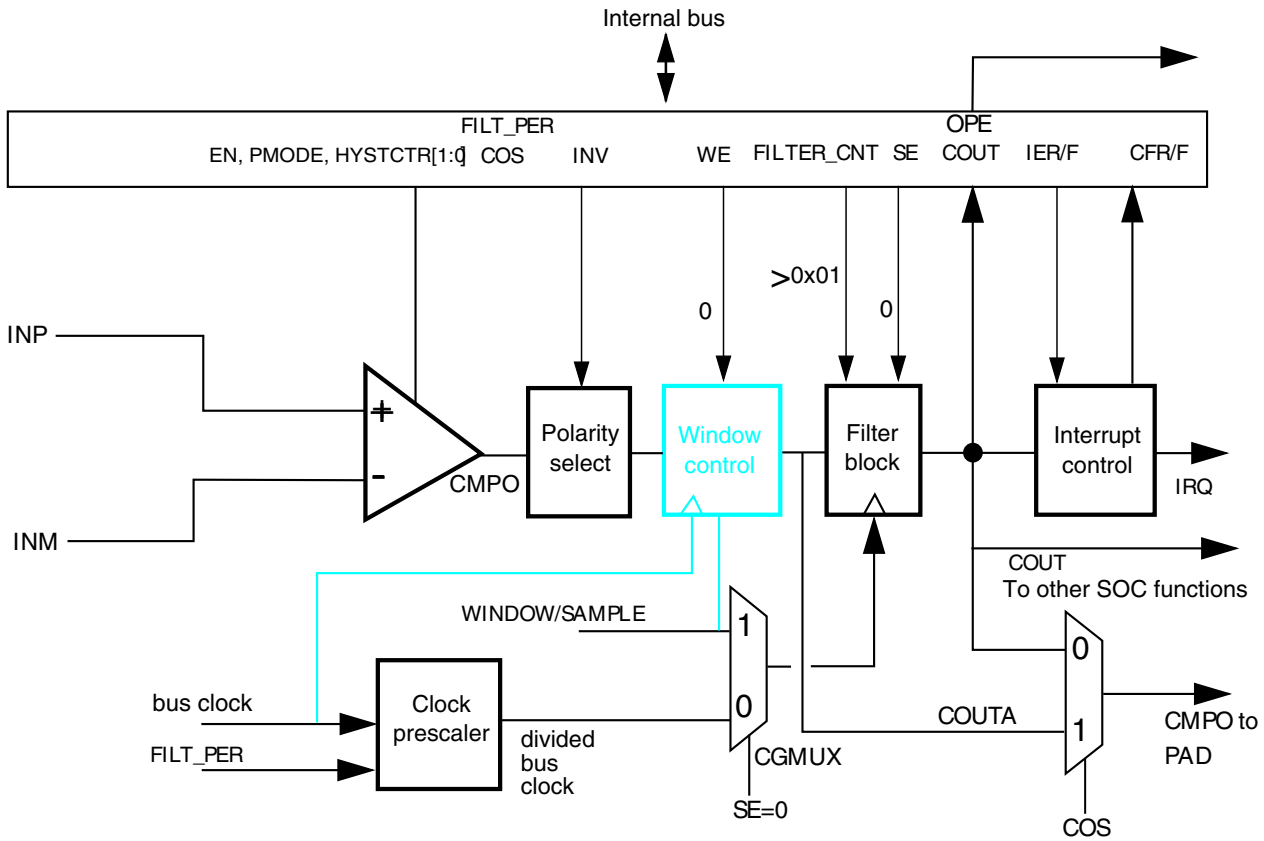



**Figure 36-31. Sampled, Filtered (# 4B): sampling point internally derived**

The only difference in operation between Sampled, Non-Filtered (# 3B) and Sampled, Filtered (# 4B) is that now, CR0[FILTER_CNT]>1, which activates filter operation.

### 36.8.1.5 Windowed mode (#s 5A & 5B)

The following figure illustrates comparator operation in the Windowed 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.

**NOTE**

The analog comparator output is passed to COUTA only when the WINDOW signal is high.

In actual operation, COUTA may lag the analog inputs by up to one bus clock cycle plus the combinational path delay through the comparator and polarity select logic.

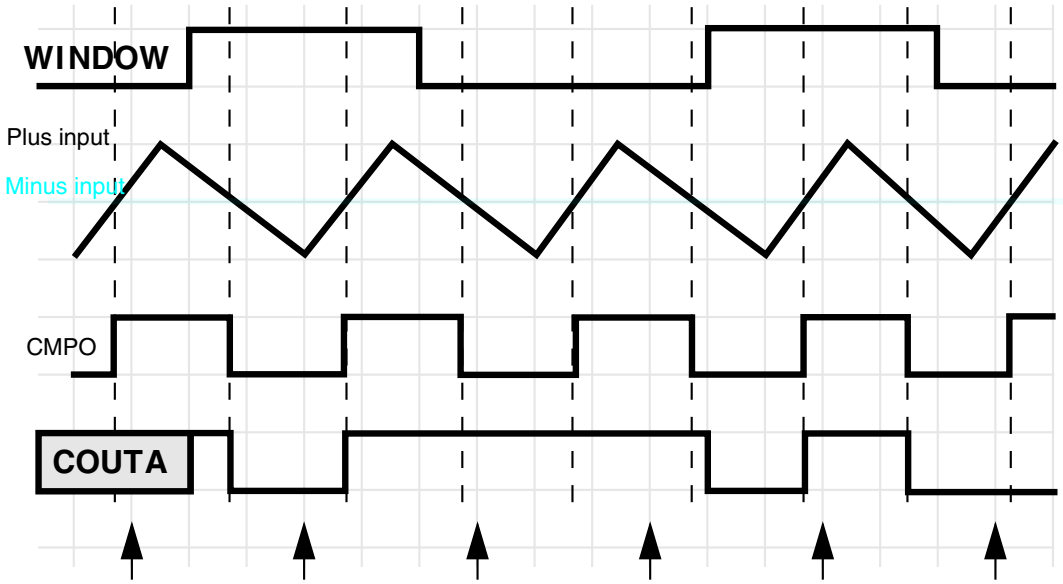



**Figure 36-32. Windowed mode operation**

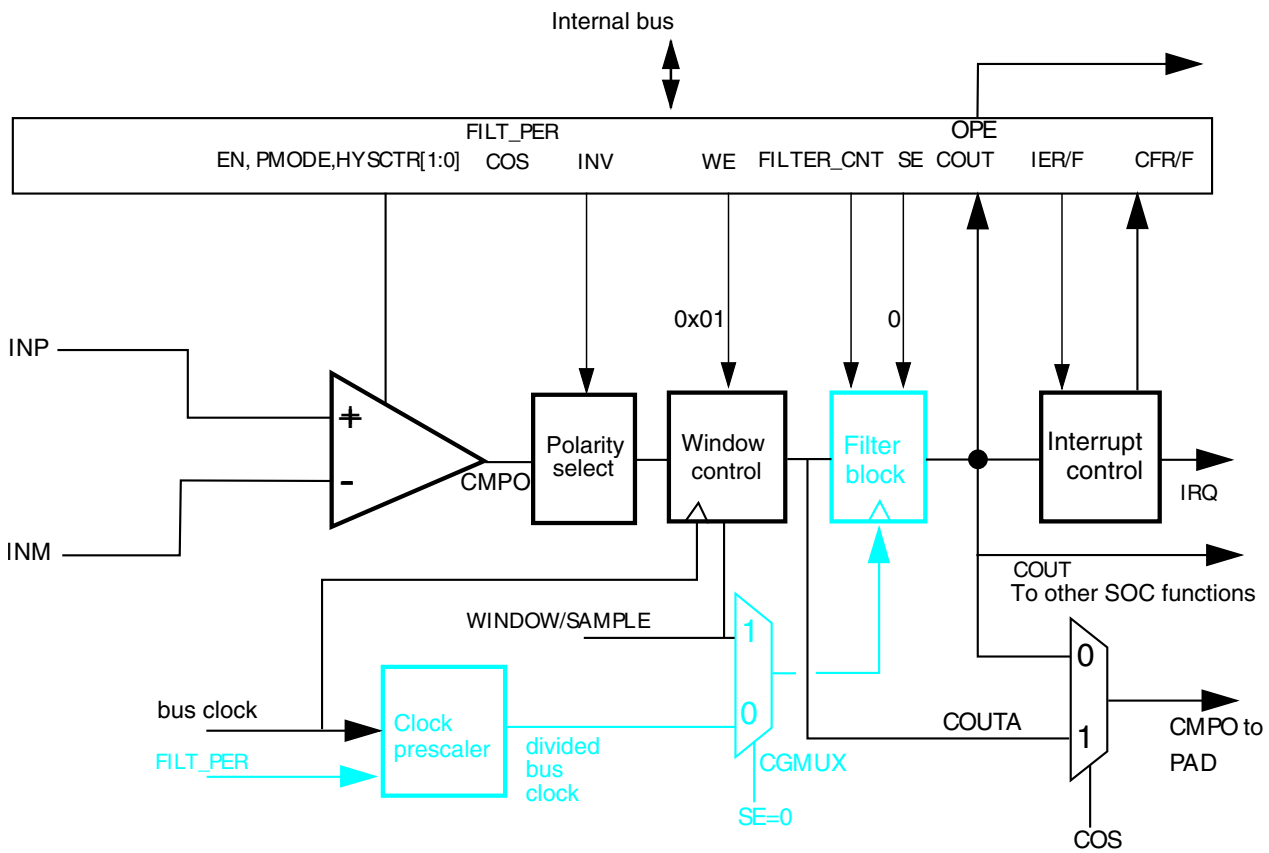



**Figure 36-33. Windowed mode**

For control configurations which result in disabling the filter block, see Filter Block Bypass Logic diagram.

When any windowed mode is active, COUTA is clocked by the bus clock whenever WINDOW = 1. The last latched value is held when WINDOW = 0.

### 36.8.1.6 Windowed/Resampled mode (# 6)

The following figure uses the same input stimulus shown in Figure 36-32, 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.

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.

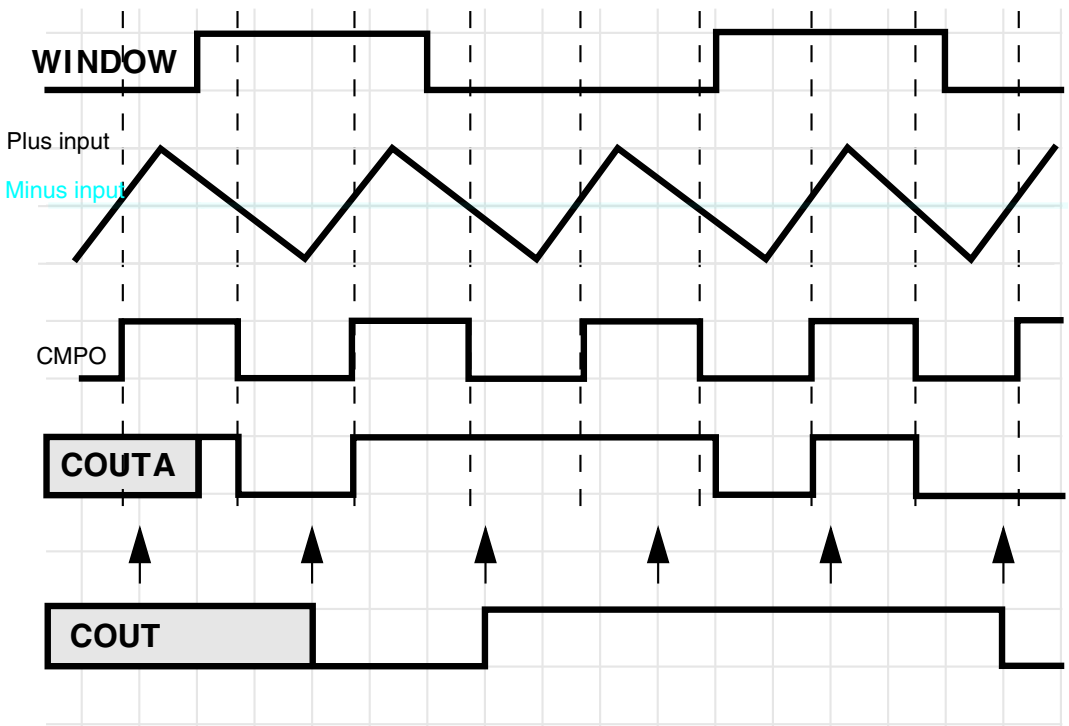



**Figure 36-34. Windowed/resampled mode operation**

This mode of operation results in an unfiltered string of comparator samples where the interval between the samples is determined by FPR[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 CR0[FILTER_CNT] must be 1.

### 36.8.1.7 Windowed/Filtered mode (#7)

This is the most complex mode of operation for the comparator block, as it uses both windowing and filtering features. It also has the highest latency of any of the modes. This can be approximated: up to 1 bus clock synchronization in the window function + ((CR0[FILTER_CNT] * FPR[FILT_PER]) + 1) * bus clock for the filter function.

When any windowed mode is active, COUTA is clocked by the bus clock whenever WINDOW = 1. The last latched value is held when WINDOW = 0.

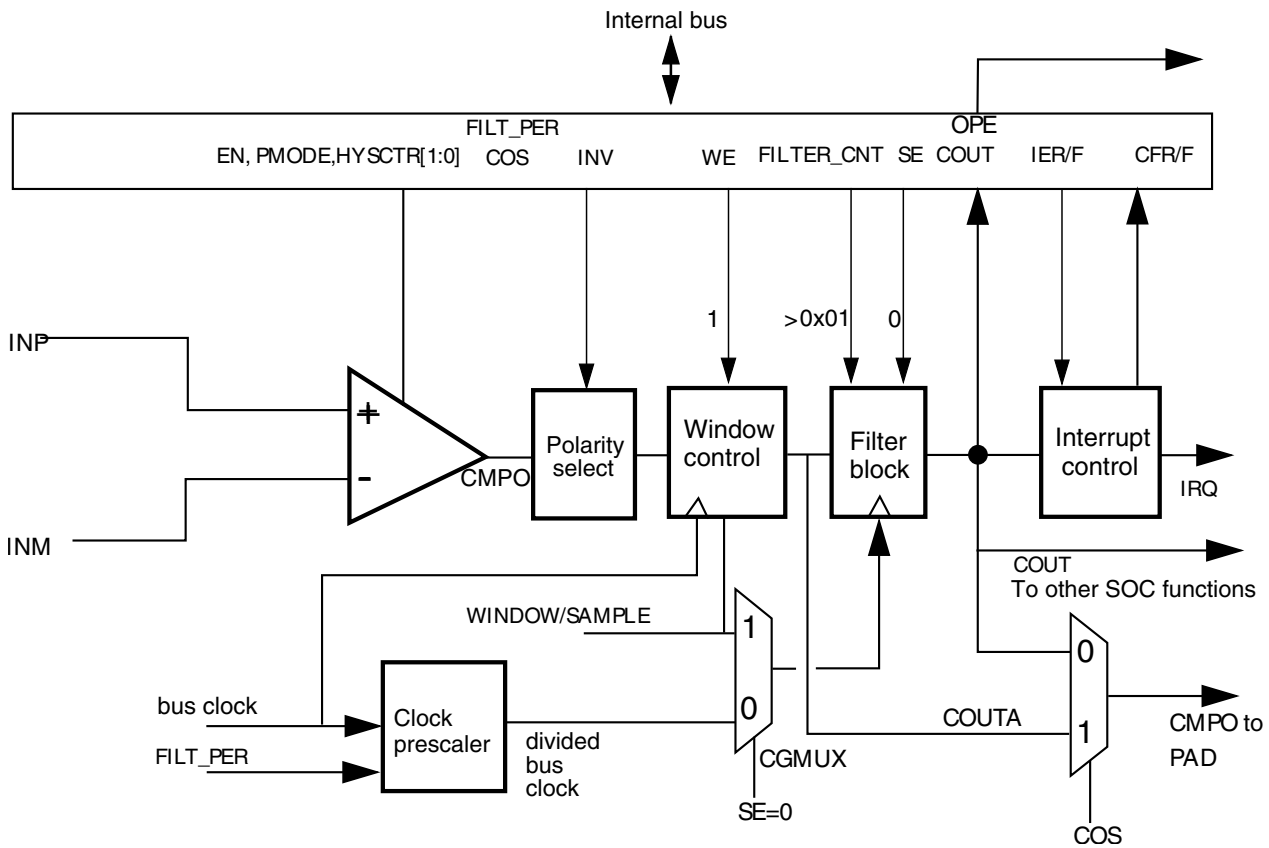



**Figure 36-35. Windowed/Filtered mode**

## 36.8.2 Power modes

### 36.8.2.1 Wait mode operation

During Wait and VLPW modes, the CMP, if enabled, continues to operate normally and a CMP interrupt can wake the MCU.

#### 36.8.2.2 Stop mode operation

Depending on clock restrictions related to the MCU core or core peripherals, the MCU is brought out of stop when a compare event occurs and the corresponding interrupt is enabled. Similarly, if CR1[OPE] is enabled, the comparator output operates as in the normal operating mode and comparator output is placed onto the external pin. In Stop modes, the comparator can be operational in both:

- High-Speed (HS) Comparison mode when CR1[PMODE] = 1
- Low-Speed (LS) Comparison mode when CR1[PMODE] = 0

It is recommended to use the LS mode to minimize power consumption.

If stop is exited with a reset, all comparator registers are put into their reset state.

#### 36.8.2.3 Low-Leakage mode operation

When the chip is in Low-Leakage modes:

- The CMP module is partially functional and is limited to Low-Speed mode, regardless of CR1[PMODE] setting
- Windowed, Sampled, and Filtered modes are not supported
- The CMP output pin is latched and does not reflect the compare output state.

The positive- and negative-input voltage can be supplied from external pins or the DAC output. The MCU can be brought out of the Low-Leakage mode if a compare event occurs and the CMP interrupt is enabled. After wakeup from low-leakage modes, the CMP module is in the reset state except for SCR[CFF] and SCR[CFR].

### 36.8.3 Startup and operation

A typical startup sequence is as follows.

The time required to stabilize COUT will be 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 Sheets for power-on delays of the comparators. The windowing function has a maximum of one bus clock period delay. The filter delay is specified in the Low-pass filter.

During operation, the propagation delay of the selected data paths must always be considered. It may take many bus clock cycles for COUT and SCR[CFR]/SCR[CFF] to reflect an input change or a configuration change to one of the components involved in the data path.

When programmed for filtering modes, COUT will initially be equal to 0, until sufficient clock cycles have elapsed to fill all stages of the filter. This occurs even if COUTA is at a logic 1.

### 36.8.4 Low-pass filter

The low-pass filter operates on the unfiltered and unsynchronized and 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 FPR[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.

#### 36.8.4.1 Enabling filter modes

Filter modes can be enabled by:

- Setting CR0[FILTER_CNT] > 0x01 and
- Setting FPR[FILT_PER] to a nonzero value or setting CR1[SE]=1

If using the divided bus clock to drive the filter, it will take samples of COUTA every FPR[FILT_PER] bus clock cycles.

The filter output will be at logic 0 when first initalized, and will subsequently change when all the consecutive CR0[FILTER_CNT] samples agree that the output value has changed. In other words, SCR[COUT] will be 0 for some initial period, even when COUTA is at logic 1.

Setting both CR1[SE] and FPR[FILT_PER] to 0 disables the filter and eliminates switching current associated with the filtering process.

**Note**

Always switch to this setting prior to making any changes in filter parameters. This resets the filter to a known state. Switching CR0[FILTER_CNT] on the fly without this intermediate step can result in unexpected behavior.

If CR1[SE]=1, the filter takes samples of COUTA on each positive transition of the sample input. The output state of the filter changes when all the consecutive CR0[FILTER_CNT] samples agree that the output value has changed.

### 36.8.4.2 Latency issues

The value of FPR[FILT_PER] or SAMPLE period must be set such that the sampling period is just longer than the period of the expected noise. This way a noise spike will corrupt only one sample. The value of CR0[FILTER_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 CR0[FILTER_CNT].

The values of FPR[FILT_PER] or SAMPLE period and CR0[FILTER_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 CR0[FILTER_CNT].

The following table 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.

**Table 36-30. Comparator sample/filter maximum latencies**

| Mode # | CR1[EN] | CR1[WE] | CR1[SE] | CR0[FILTER_CNT] | FPR[FILT_PER] | Operation | Maximum latency[1] |
|---|---|---|---|---|---|---|---|
| 1 | 0 | X | X | X | X | Disabled | N/A |
| 2A | 1 | 0 | 0 | 0x00 | X | Continuous Mode | $T_{PD}$ |
| 2B | 1 | 0 | 0 | X | 0x00 | | |
| 3A | 1 | 0 | 1 | 0x01 | X | Sampled, Non-Filtered mode | $T_{PD} + T_{SAMPLE} + T_{per}$ |
| 3B | 1 | 0 | 0 | 0x01 | > 0x00 | | $T_{PD}$ + (FPR[FILT_PER] * $T_{per}$) + $T_{per}$ |

*Table continues on the next page...*

**Table 36-30. Comparator sample/filter maximum latencies (continued)**

| Mode # | CR1[EN] | CR1[WE] | CR1[SE] | CR0[FILTER_CNT] | FPR[FILT_PER] | Operation | Maximum latency[1] |
|---|---|---|---|---|---|---|---|
| 4A | 1 | 0 | 1 | > 0x01 | X | Sampled, Filtered mode | $T_{PD}$ + (CR0[FILTER_CNT] * $T_{SAMPLE}$) + $T_{per}$ |
| 4B | 1 | 0 | 0 | > 0x01 | > 0x00 | | $T_{PD}$ + (CR0[FILTER_CNT] * FPR[FILT_PER] x $T_{per}$) + $T_{per}$ |
| 5A | 1 | 1 | 0 | 0x00 | X | Windowed mode | $T_{PD}$ + $T_{per}$ |
| 5B | 1 | 1 | 0 | X | 0x00 | | $T_{PD}$ + $T_{per}$ |
| 6 | 1 | 1 | 0 | 0x01 | 0x01 - 0xFF | Windowed / Resampled mode | $T_{PD}$ + (FPR[FILT_PER] * $T_{per}$) + 2$T_{per}$ |
| 7 | 1 | 1 | 0 | > 0x01 | 0x01 - 0xFF | Windowed / Filtered mode | $T_{PD}$ + (CR0[FILTER_CNT] * FPR[FILT_PER] x $T_{per}$) + 2$T_{per}$ |

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.

## 36.9 CMP interrupts

The CMP module is capable of generating an interrupt on either the rising- or falling-edge of the comparator output, or both. The following table gives the conditions in which the interrupt request is asserted and deasserted.

| When | Then |
|---|---|
| SCR[IER] and SCR[CFR] are set | The interrupt request is asserted |
| SCR[IEF] and SCR[CFF] are set | The interrupt request is asserted |
| SCR[IER] and SCR[CFR] are cleared for a rising-edge interrupt | The interrupt request is deasserted |
| SCR[IEF] and SCR[CFF] are cleared for a falling-edge interrupt | The interrupt request is deasserted |

## 36.10 DMA support

Normally, the CMP generates a CPU interrupt if there is a change on the COUT. When DMA support is enabled by setting SCR[DMAEN] and the interrupt is enabled by setting SCR[IER], SCR[IEF], 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 that deasserts the DMA transfer request and clears the flag to allow a subsequent change on comparator output to occur and force another DMA request.

The comparator can remain functional in STOP modes. When DMA support is enabled by setting SCR[DMAEN] and the interrupt is enabled by setting SCR[IER], SCR[IEF], or both, the corresponding change on COUT forces a DMA transfer request to wake up the system from STOP modes. After the data transfer has finished, system will go back to STOP modes. Refer to DMA chapters in the device reference manual for the asynchronous DMA function for details.

## 36.11 CMP Asyncrhonous DMA support

The comparator can remain functional in STOP modes. When DMA support is enabled by setting SCR[DMAEN] and the interrupt is enabled by setting SCR[IER], SCR[IEF], or both, the corresponding change on COUT forces a DMA transfer request to wake up the system from STOP modes. After the data transfer has finished, system will go back to STOP modes. Refer to DMA chapters in the device reference manual for the asynchronous DMA function for details.

## 36.12 Digital-to-analog converter

The following figure shows the block diagram of the DAC module. It contains a 64-tap resistor ladder network and a 64-to-1 multiplexer, which selects an output voltage from one of 64 distinct levels that outputs from DACO. It is controlled through the DAC Control Register (DACCR). Its supply reference source can be selected from two sources $V_{in1}$ and $V_{in2}$. The module can be powered down or disabled when not in use. When in Disabled mode, DACO is connected to the analog ground.

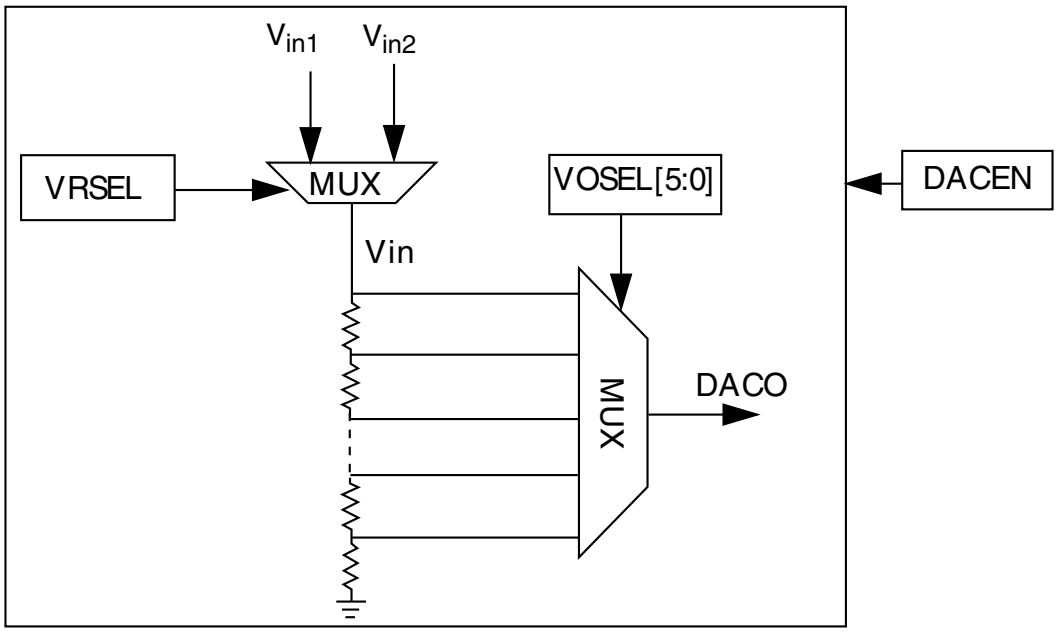



**Figure 36-36. 6-bit DAC block diagram**

## 36.13 DAC functional description

This section provides DAC functional description.

### 36.13.1 Voltage reference source select

- $V_{in1}$ connects to the primary voltage source as supply reference of 64 tap resistor ladder
- $V_{in2}$ connects to an alternate voltage source

## 36.14 DAC resets

This module has a single reset input, corresponding to the chip-wide peripheral reset.

## 36.15 DAC clocks

This module has a single clock input, the bus clock.

## 36.16 DAC interrupts

This module has no interrupts.

# Chapter 37
# 12-bit Digital-to-Analog Converter (DAC)

## 37.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The 12-bit digital-to-analog converter (DAC) is a low-power, general-purpose DAC. The output of the DAC can be placed on an external pin or set as one of the inputs to the analog comparator, op-amps, or ADC.

## 37.2 Features

The features of the DAC module include:

- On-chip programmable reference generator output. The voltage output range is from 1⁄4096 $V_{in}$ to $V_{in}$, and the step is 1⁄4096 $V_{in}$, where $V_{in}$ is the input voltage.
- $V_{in}$ can be selected from two reference sources
- Static operation in Normal Stop mode
- 16-word data buffer supported with configurable watermark and multiple operation modes
- DMA support

## 37.3 Block diagram

The block diagram of the DAC module is as follows:

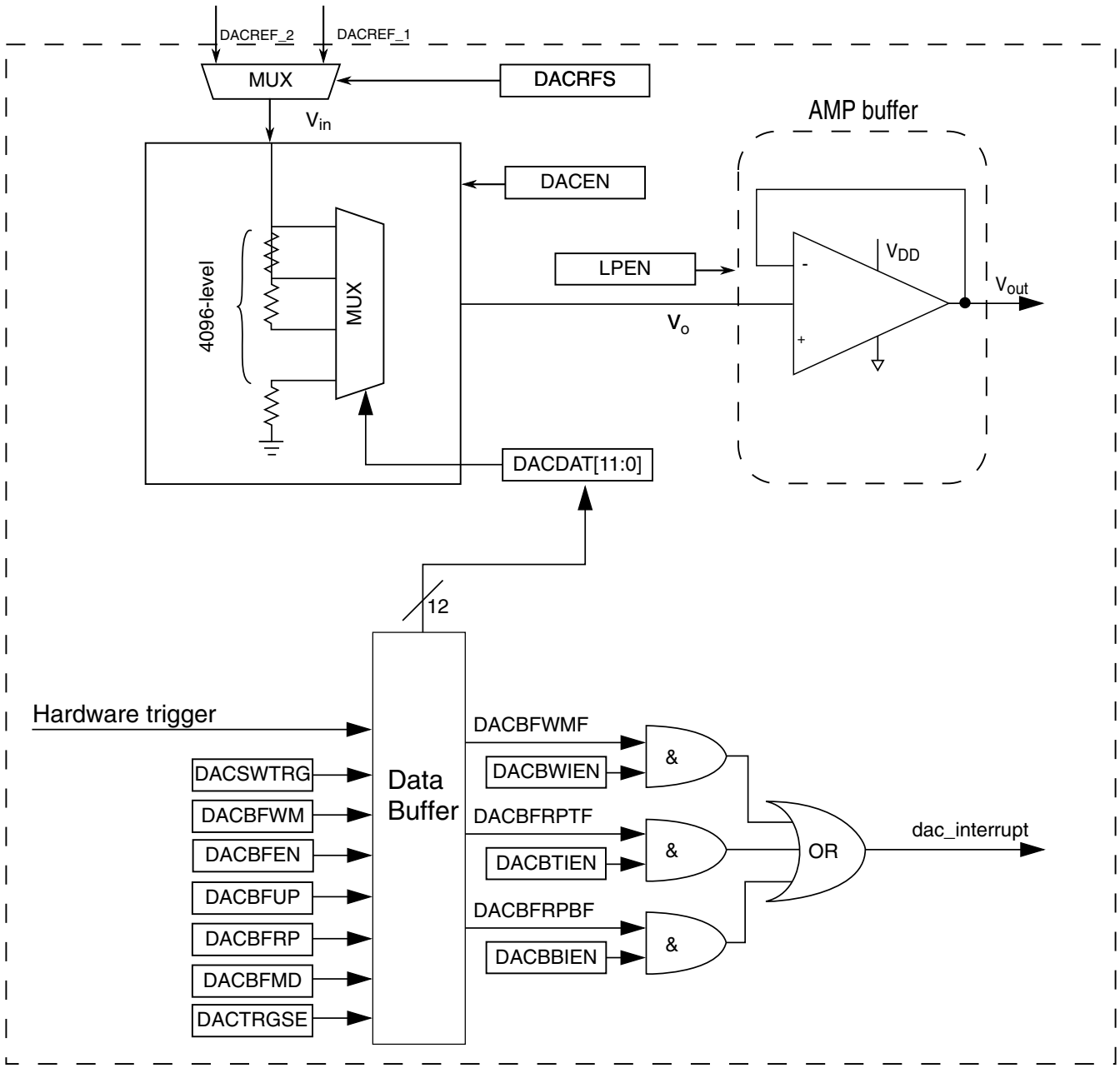



**Figure 37-1. DAC block diagram**

## 37.4 Memory map/register definition

The DAC has registers to control analog comparator and programmable voltage divider to perform the digital-to-analog functions.

## DAC memory map

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 400C_C000 | DAC Data Low Register (DAC0_DAT0L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_C001 | DAC Data High Register (DAC0_DAT0H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_C002 | DAC Data Low Register (DAC0_DAT1L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_C003 | DAC Data High Register (DAC0_DAT1H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_C004 | DAC Data Low Register (DAC0_DAT2L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_C005 | DAC Data High Register (DAC0_DAT2H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_C006 | DAC Data Low Register (DAC0_DAT3L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_C007 | DAC Data High Register (DAC0_DAT3H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_C008 | DAC Data Low Register (DAC0_DAT4L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_C009 | DAC Data High Register (DAC0_DAT4H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_C00A | DAC Data Low Register (DAC0_DAT5L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_C00B | DAC Data High Register (DAC0_DAT5H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_C00C | DAC Data Low Register (DAC0_DAT6L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_C00D | DAC Data High Register (DAC0_DAT6H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_C00E | DAC Data Low Register (DAC0_DAT7L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_C00F | DAC Data High Register (DAC0_DAT7H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_C010 | DAC Data Low Register (DAC0_DAT8L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_C011 | DAC Data High Register (DAC0_DAT8H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_C012 | DAC Data Low Register (DAC0_DAT9L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_C013 | DAC Data High Register (DAC0_DAT9H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_C014 | DAC Data Low Register (DAC0_DAT10L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_C015 | DAC Data High Register (DAC0_DAT10H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_C016 | DAC Data Low Register (DAC0_DAT11L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_C017 | DAC Data High Register (DAC0_DAT11H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_C018 | DAC Data Low Register (DAC0_DAT12L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_C019 | DAC Data High Register (DAC0_DAT12H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_C01A | DAC Data Low Register (DAC0_DAT13L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_C01B | DAC Data High Register (DAC0_DAT13H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_C01C | DAC Data Low Register (DAC0_DAT14L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_C01D | DAC Data High Register (DAC0_DAT14H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_C01E | DAC Data Low Register (DAC0_DAT15L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_C01F | DAC Data High Register (DAC0_DAT15H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_C020 | DAC Status Register (DAC0_SR) | 8 | R/W | 02h | 37.4.3/875 |
| 400C_C021 | DAC Control Register (DAC0_C0) | 8 | R/W | 00h | 37.4.4/876 |
| 400C_C022 | DAC Control Register 1 (DAC0_C1) | 8 | R/W | 00h | 37.4.5/877 |
| 400C_C023 | DAC Control Register 2 (DAC0_C2) | 8 | R/W | 0Fh | 37.4.6/878 |
| 400C_D000 | DAC Data Low Register (DAC1_DAT0L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_D001 | DAC Data High Register (DAC1_DAT0H) | 8 | R/W | 00h | 37.4.2/875 |

*Table continues on the next page...*

## DAC memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 400C_D002 | DAC Data Low Register (DAC1_DAT1L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_D003 | DAC Data High Register (DAC1_DAT1H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_D004 | DAC Data Low Register (DAC1_DAT2L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_D005 | DAC Data High Register (DAC1_DAT2H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_D006 | DAC Data Low Register (DAC1_DAT3L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_D007 | DAC Data High Register (DAC1_DAT3H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_D008 | DAC Data Low Register (DAC1_DAT4L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_D009 | DAC Data High Register (DAC1_DAT4H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_D00A | DAC Data Low Register (DAC1_DAT5L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_D00B | DAC Data High Register (DAC1_DAT5H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_D00C | DAC Data Low Register (DAC1_DAT6L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_D00D | DAC Data High Register (DAC1_DAT6H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_D00E | DAC Data Low Register (DAC1_DAT7L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_D00F | DAC Data High Register (DAC1_DAT7H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_D010 | DAC Data Low Register (DAC1_DAT8L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_D011 | DAC Data High Register (DAC1_DAT8H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_D012 | DAC Data Low Register (DAC1_DAT9L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_D013 | DAC Data High Register (DAC1_DAT9H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_D014 | DAC Data Low Register (DAC1_DAT10L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_D015 | DAC Data High Register (DAC1_DAT10H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_D016 | DAC Data Low Register (DAC1_DAT11L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_D017 | DAC Data High Register (DAC1_DAT11H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_D018 | DAC Data Low Register (DAC1_DAT12L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_D019 | DAC Data High Register (DAC1_DAT12H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_D01A | DAC Data Low Register (DAC1_DAT13L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_D01B | DAC Data High Register (DAC1_DAT13H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_D01C | DAC Data Low Register (DAC1_DAT14L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_D01D | DAC Data High Register (DAC1_DAT14H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_D01E | DAC Data Low Register (DAC1_DAT15L) | 8 | R/W | 00h | 37.4.1/875 |
| 400C_D01F | DAC Data High Register (DAC1_DAT15H) | 8 | R/W | 00h | 37.4.2/875 |
| 400C_D020 | DAC Status Register (DAC1_SR) | 8 | R/W | 02h | 37.4.3/875 |
| 400C_D021 | DAC Control Register (DAC1_C0) | 8 | R/W | 00h | 37.4.4/876 |
| 400C_D022 | DAC Control Register 1 (DAC1_C1) | 8 | R/W | 00h | 37.4.5/877 |
| 400C_D023 | DAC Control Register 2 (DAC1_C2) | 8 | R/W | 0Fh | 37.4.6/878 |

## 37.4.1 DAC Data Low Register (DACx_DATnL)

Address: Base address + 0h offset + (2d × i), where i=0d to 15d

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | DATA0 | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### DACx_DATnL field descriptions

| Field | Description |
|---|---|
| 7–0<br>DATA0 | When the DAC buffer is not enabled, DATA[11:0] controls the output voltage based on the following formula: $V_{out} = V_{in} * (1 + DACDAT0[11:0])/4096$<br>When the DAC buffer is enabled, DATA is mapped to the 16-word buffer. |

## 37.4.2 DAC Data High Register (DACx_DATnH)

Address: Base address + 1h offset + (2d × i), where i=0d to 15d

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | | | | DATA1 | | | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### DACx_DATnH field descriptions

| Field | Description |
|---|---|
| 7–4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3–0<br>DATA1 | When the DAC Buffer is not enabled, DATA[11:0] controls the output voltage based on the following formula. $V_{out} = V_{in} * (1 + DACDAT0[11:0])/4096$<br>When the DAC buffer is enabled, DATA[11:0] is mapped to the 16-word buffer. |

## 37.4.3 DAC Status Register (DACx_SR)

If DMA is enabled, the flags can be cleared automatically by DMA when the DMA request is done. Writing 0 to a field clears it whereas writing 1 has no effect. After reset, DACBFRPTF is set and can be cleared by software, if needed. The flags are set only when the data buffer status is changed.

**NOTE**

Do not use 32/16-bit accesses to this register.

Address: Base address + 20h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | | | | | DACBFWMF | DACBFRPTF | DACBFRPBF |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

### DACx_SR field descriptions

| Field | Description |
|---|---|
| 7–3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>DACBFWMF | DAC Buffer Watermark Flag<br><br>0 The DAC buffer read pointer has not reached the watermark level.<br>1 The DAC buffer read pointer has reached the watermark level. |
| 1<br>DACBFRPTF | DAC Buffer Read Pointer Top Position Flag<br><br>0 The DAC buffer read pointer is not zero.<br>1 The DAC buffer read pointer is zero. |
| 0<br>DACBFRPBF | DAC Buffer Read Pointer Bottom Position Flag<br><br>0 The DAC buffer read pointer is not equal to C2[DACBFUP].<br>1 The DAC buffer read pointer is equal to C2[DACBFUP]. |

## 37.4.4 DAC Control Register (DACx_C0)

### NOTE

Do not use 32- or 16-bit accesses to this register.

Address: Base address + 21h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | DACEN | DACRFS | DACTRGSEL | 0 | LPEN | DACBWIEN | DACBTIEN | DACBBIEN |
| Write | | | | DACSWTRG | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### DACx_C0 field descriptions

| Field | Description |
|---|---|
| 7<br>DACEN | DAC Enable<br><br>Starts the Programmable Reference Generator operation.<br><br>0 The DAC system is disabled.<br>1 The DAC system is enabled. |
| 6<br>DACRFS | DAC Reference Select |

*Table continues on the next page...*

**DACx_C0 field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 The DAC selects DACREF_1 as the reference voltage.<br>1 The DAC selects DACREF_2 as the reference voltage. |
| 5<br>DACTRGSEL | DAC Trigger Select<br><br>0 The DAC hardware trigger is selected.<br>1 The DAC software trigger is selected. |
| 4<br>DACSWTRG | DAC Software Trigger<br><br>Active high. This is a write-only field, which always reads 0. If DAC software trigger is selected and buffer is enabled, writing 1 to this field will advance the buffer read pointer once.<br><br>0 The DAC soft trigger is not valid.<br>1 The DAC soft trigger is valid. |
| 3<br>LPEN | DAC Low Power Control<br><br>**NOTE:** See the 12-bit DAC electrical characteristics of the device data sheet for details on the impact of the modes below.<br><br>0 High-Power mode<br>1 Low-Power mode |
| 2<br>DACBWIEN | DAC Buffer Watermark Interrupt Enable<br><br>0 The DAC buffer watermark interrupt is disabled.<br>1 The DAC buffer watermark interrupt is enabled. |
| 1<br>DACBTIEN | DAC Buffer Read Pointer Top Flag Interrupt Enable<br><br>0 The DAC buffer read pointer top flag interrupt is disabled.<br>1 The DAC buffer read pointer top flag interrupt is enabled. |
| 0<br>DACBBIEN | DAC Buffer Read Pointer Bottom Flag Interrupt Enable<br><br>0 The DAC buffer read pointer bottom flag interrupt is disabled.<br>1 The DAC buffer read pointer bottom flag interrupt is enabled. |

## 37.4.5 DAC Control Register 1 (DACx_C1)

**NOTE**

Do not use 32- or 16-bit accesses to this register.

Address: Base address + 22h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | DMAEN | 0 | | DACBFWM | | DACBFMD | | DACBFEN |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### DACx_C1 field descriptions

| Field | Description |
|---|---|
| 7<br>DMAEN | DMA Enable Select<br><br>0 DMA is disabled.<br>1 DMA is enabled. When DMA is enabled, the DMA request will be generated by original interrupts. The interrupts will not be presented on this module at the same time. |
| 6–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4–3<br>DACBFWM | DAC Buffer Watermark Select<br><br>Controls when SR[DACBFWMF] will be set. When the DAC buffer read pointer reaches the word defined by this field, which is 1–4 words away from the upper limit (DACBUP), SR[DACBFWMF] will be set. This allows user configuration of the watermark interrupt.<br><br>00 1 word<br>01 2 words<br>10 3 words<br>11 4 words |
| 2–1<br>DACBFMD | DAC Buffer Work Mode Select<br><br>00 Normal mode<br>01 Swing mode<br>10 One-Time Scan mode<br>11 Reserved |
| 0<br>DACBFEN | DAC Buffer Enable<br><br>0 Buffer read pointer is disabled. The converted data is always the first word of the buffer.<br>1 Buffer read pointer is enabled. The converted data is the word that the read pointer points to. It means converted data can be from any word of the buffer. |

## 37.4.6 DAC Control Register 2 (DACx_C2)

**NOTE**

Do not use 32- or 16-bit accesses to this register.

Address: Base address + 23h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | DACBFRP | | | | DACBFUP | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

### DACx_C2 field descriptions

| Field | Description |
|---|---|
| 7–4<br>DACBFRP | DAC Buffer Read Pointer |

*Table continues on the next page...*

DACx_C2 field descriptions (continued)

| Field | Description |
|---|---|
| | Keeps the current value of the buffer read pointer. |
| 3–0<br>DACBFUP | DAC Buffer Upper Limit<br><br>Selects the upper limit of the DAC buffer. The buffer read pointer cannot exceed it. |

## 37.5 Functional description

The 12-bit DAC module can select one of the two reference inputs—DACREF_1 and DACREF_2 as the DAC reference voltage, $V_{in}$ by C0[DACRFS]. See the module introduction for information on the source for DACREF_1 and DACREF_2.

When the DAC is enabled, it converts the data in DACDAT0[11:0] or the data from the DAC data buffer to a stepped analog output voltage. The output voltage range is from $V_{in}$ to $V_{in}$/4096, and the step is $V_{in}$/4096.

### 37.5.1 DAC data buffer operation

When the DAC is enabled and the buffer is not enabled, the DAC module always converts the data in DAT0 to analog output voltage.

When both the DAC and the buffer are enabled, the DAC converts the data in the data buffer to analog output voltage. The data buffer read pointer advances to the next word whenever any hardware or software trigger event occurs. Refer to Introduction for the hardware trigger connection.

The data buffer can be configured to operate in Normal mode, Swing mode, or One-Time Scan mode. When the buffer operation is switched from one mode to another, the read pointer does not change. The read pointer can be set to any value between 0 and C2[DACBFUP] by writing C2[DACBFRP].

#### 37.5.1.1 DAC data buffer interrupts

There are several interrupts and associated flags that can be configured for the DAC buffer. SR[DACBFRPBF] is set when the DAC buffer read pointer reaches the DAC buffer upper limit, that is, C2[DACBFRP] = C2[DACBFUP]. SR[DACBFRPTF] is set when the DAC read pointer is equal to the start position, 0. Finally, SR[DACBFWMF] is

set when the DAC buffer read pointer has reached the position defined by C1[DACBFWM]. C1[DACBFWM] can be used to generate an interrupt when the DAC buffer read pointer is between 1 to 4 words from C2[DACBFUP].

#### 37.5.1.2 Modes of DAC data buffer operation

The following table describes the different modes of data buffer operation for the DAC module.

**Table 37-118. Modes of DAC data buffer operation**

| Modes | Description |
|---|---|
| Buffer Normal mode | This is the default mode. The buffer works as a circular buffer. The read pointer increases by one, every time the trigger occurs. When the read pointer reaches the upper limit, it goes to 0 directly in the next trigger event. |
| Buffer Swing mode | This mode is similar to the normal mode. However, when the read pointer reaches the upper limit, it does not go to 0. It will descend by 1 in the next trigger events until 0 is reached. |
| Buffer One-time Scan mode | The read pointer increases by 1 every time the trigger occurs. When it reaches the upper limit, it stops there. If read pointer is reset to the address other than the upper limit, it will increase to the upper address and stop there again.<br>**NOTE:** If the software set the read pointer to the upper limit, the read pointer will not advance in this mode. |

### 37.5.2 DMA operation

When DMA is enabled, DMA requests are generated instead of interrupt requests. The DMA Done signal clears the DMA request.

The status register flags are still set and are cleared automatically when the DMA completes.

### 37.5.3 Resets

During reset, the DAC is configured in the default mode and is disabled.

### 37.5.4 Low-Power mode operation

The following table shows the wait mode and the stop mode operation of the DAC module.

**Table 37-119. Modes of operation**

| Modes of operation | Description |
|---|---|
| Wait mode | The DAC will operate normally, if enabled. |
| Stop mode | If enabled, the DAC module continues to operate in Normal Stop mode and the output voltage will hold the value before stop.<br><br>In low-power stop modes, the DAC is fully shut down. |

## NOTE

The assignment of module modes to core modes is chip-specific. For module-to-core mode assignments, see the chapter that describes how modules are configured.

# Chapter 38
# Voltage Reference (VREFV1)

## 38.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The Voltage Reference(VREF) is intended to supply an accurate voltage output that can be trimmed in 0.5 mV steps. The VREF can be used in applications to provide a reference voltage to external devices or used internally as a reference to analog peripherals such as the ADC, DAC, or CMP. The voltage reference has three operating modes that provide different levels of supply rejection and power consumption..

The following figure is a block diagram of the Voltage Reference.

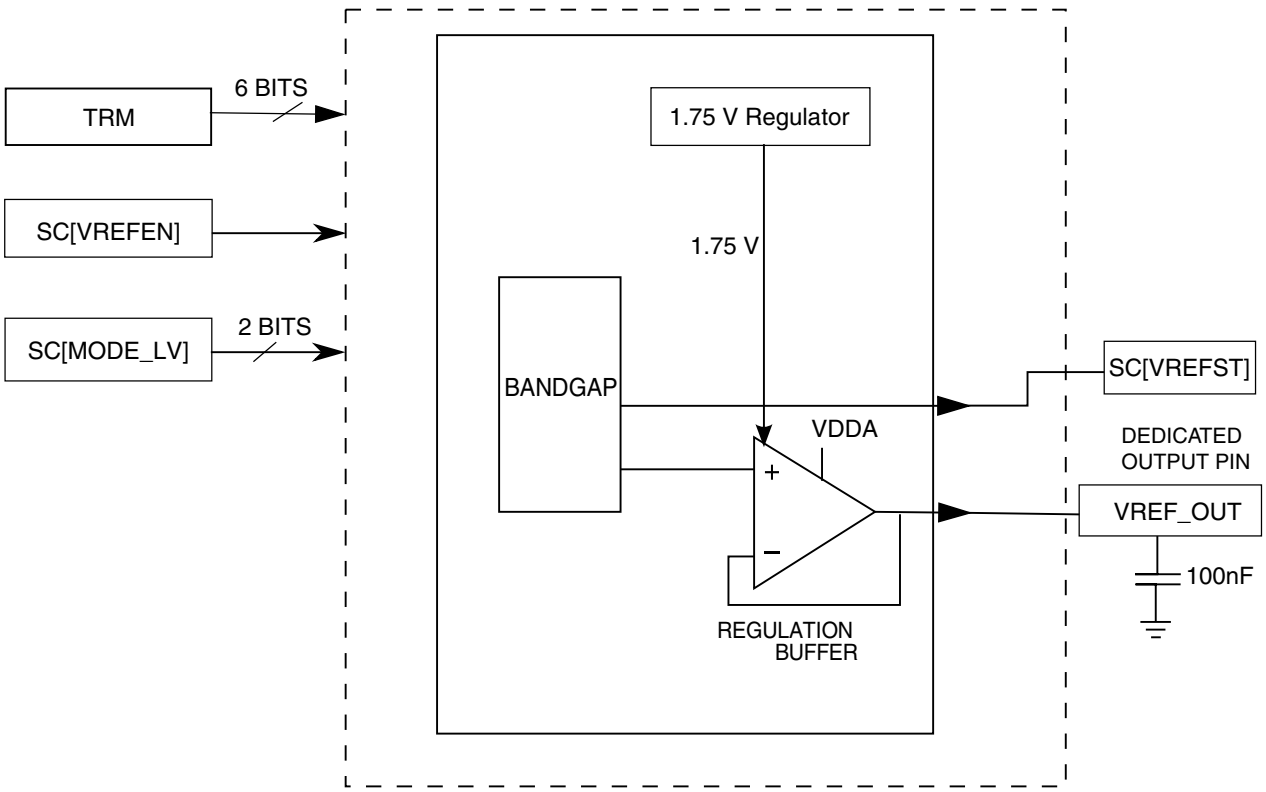



**Figure 38-1. Voltage reference block diagram**

## 38.1.1 Overview

The Voltage Reference provides a buffered reference voltage for use as an external reference. In addition, the buffered reference is available internally for use with on chip peripherals such as ADCs and DACs. Refer to the chip configuration chapter for a description of these options. The reference voltage is output on a dedicated output pin when the VREF is enabled. The Voltage Reference output can be trimmed with a resolution of 0.5mV by means of the TRM register TRIM[5:0] bitfield.

## 38.1.2 Features

The Voltage Reference has the following features:

- Programmable trim register with 0.5 mV steps, automatically loaded with factory trimmed value upon reset
- 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)
- 1.2 V output at room temperature
- Dedicated output pin, VREF_OUT

### 38.1.3 Modes of Operation

The Voltage Reference continues normal operation in Run, Wait, and Stop modes. The Voltage Reference can also run in Very Low Power Run (VLPR), Very Low Power Wait (VLPW) and Very Low Power Stop (VLPS). If it is desired to use the VREF regulator in the very low power modes, the system reference voltage must be enabled in these modes. Refer to the chip configuration chapter for information on enabling this mode of operation. Having the VREF regulator enabled does increase current consumption. In very low power modes it may be desirable to disable the VREF regulator to minimize current consumption. Note however that the accuracy of the output voltage will be 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 how modules are configured.

### 38.1.4 VREF Signal Descriptions

The following table shows the Voltage Reference signals properties.

**Table 38-1. 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.

# 38.2 Memory Map and Register Definition

**VREF memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4007_4000 | VREF Trim Register (VREF_TRM) | 8 | R/W | See section | 38.2.1/886 |
| 4007_4001 | VREF Status and Control Register (VREF_SC) | 8 | R/W | 00h | 38.2.2/887 |

## 38.2.1 VREF Trim Register (VREF_TRM)

This register contains bits that contain the trim data for the Voltage Reference.

Address: 4007_4000h base + 0h offset = 4007_4000h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | Reserved | CHOPEN | TRIM | | | | | |
| Reset | x* | 0 | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**VREF_TRM field descriptions**

| Field | Description |
|---|---|
| 7<br>Reserved | This field is reserved.<br>Upon reset this value is loaded with a factory trim value. |
| 6<br>CHOPEN | Chop oscillator enable. When set, internal chopping operation is enabled and the internal analog offset will be minimized.<br><br>This bit is set during factory trimming of the VREF voltage. This bit should be written to 1 to achieve the performance stated in the data sheet.<br><br>0 Chop oscillator is disabled.<br>1 Chop oscillator is enabled. |
| 5–0<br>TRIM | Trim bits<br><br>These bits change the resulting VREF by approximately ± 0.5 mV for each step.<br><br>**NOTE:** Min = minimum and max = maximum voltage reference output. For minimum and maximum voltage reference output values, refer to the Data Sheet for this chip.<br><br>000000 Min<br>.... ....<br>111111 Max |

## 38.2.2 VREF Status and Control Register (VREF_SC)

This register contains the control bits used to enable the internal voltage reference and to select the buffer mode to be used.

Address: 4007_4000h base + 1h offset = 4007_4001h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | VREFEN | REGEN | ICOMPEN | 0 | 0 | VREFST | MODE_LV | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**VREF_SC field descriptions**

| Field | Description |
|---|---|
| 7<br>VREFEN | Internal Voltage Reference enable<br><br>This bit is used to enable the bandgap reference within the Voltage Reference module.<br><br>**NOTE:** After the VREF is enabled, turning off the clock to the VREF module via the corresponding clock gate register will not disable the VREF. VREF must be disabled via this VREFEN bit.<br><br>0 The module is disabled.<br>1 The module is enabled. |
| 6<br>REGEN | Regulator enable<br><br>This bit is used to enable 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 chapter for a description on how this can be achieved.<br><br>This bit is set during factory trimming of the VREF voltage. This bit should be written to 1 to achieve the performance stated in the data sheet.<br><br>0 Internal 1.75 V regulator is disabled.<br>1 Internal 1.75 V regulator is enabled. |
| 5<br>ICOMPEN | Second order curvature compensation enable<br><br>This bit is set during factory trimming of the VREF voltage. This bit should be written to 1 to achieve the performance stated in the data sheet.<br><br>0 Disabled<br>1 Enabled |
| 4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>VREFST | Internal Voltage Reference stable |

*Table continues on the next page...*

**VREF_SC field descriptions (continued)**

| Field | Description |
|---|---|
| | This bit indicates that the bandgap reference within the Voltage Reference module has completed its startup and stabilization.<br><br>0 The module is disabled or not stable.<br>1 The module is stable. |
| 1–0<br>MODE_LV | Buffer Mode selection<br><br>These bits select the buffer modes for the Voltage Reference module.<br><br>00 Bandgap on only, for stabilization and startup<br>01 High power buffer mode enabled<br>10 Low-power buffer mode enabled<br>11 Reserved |

## 38.3 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 100 nF capacitor must always be connected between VREF_OUT and VSSA if the VREF is being used.

The following table shows all possible function configurations of the Voltage Reference.

**Table 38-5. Voltage Reference function configurations**

| SC[VREFEN] | SC[MODE_LV] | Configuration | Functionality |
|---|---|---|---|
| 0 | X | Voltage Reference disabled | Off |
| 1 | 00 | Voltage Reference enabled, bandgap on only | Startup and standby |
| 1 | 01 | Voltage Reference enabled, high-power buffer on | VREF_OUT available for internal and external use. 100 nF capacitor is required. |
| 1 | 10 | Voltage Reference enabled, low power buffer on | VREF_OUT available for internal and external use. 100 nF capacitor is required. |
| 1 | 11 | Reserved | Reserved |

### 38.3.1 Voltage Reference Disabled, SC[VREFEN] = 0

When SC[VREFEN] = 0, the Voltage Reference is disabled, the VREF bandgap and the output buffers are disabled. The Voltage Reference is in off mode.

## 38.3.2 Voltage Reference Enabled, SC[VREFEN] = 1

When SC[VREFEN] = 1, the Voltage Reference is enabled, and different modes should be set by the SC[MODE_LV] bits.

### 38.3.2.1 SC[MODE_LV]=00

The internal VREF bandgap is enabled to generate an accurate 1.2 V output that can be trimmed with the TRM register's TRIM[5:0] bitfield. The bandgap requires some time for startup and stabilization. SC[VREFST] can be monitored to determine if the stabilization and startup is complete.

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) and the value is specified in the appropriate device data sheet.

### 38.3.2.2 SC[MODE_LV] = 01

The internal VREF bandgap is on. The high power buffer is enabled to generate a buffered 1.2 V voltage to VREF_OUT. 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 (SC[MODE_LV] = 00, SC[VREFEN] = 1) 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 SC[VREFST] = 1.

In this mode, a 100 nF capacitor is required to connect between the VREF_OUT pin and VSSA.

### 38.3.2.3 SC[MODE_LV] = 10

The internal VREF bandgap is on. The low power buffer is enabled to generate a buffered 1.2 V voltage to VREF_OUT. 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 (SC[MODE_LV] = 00, SC[VREFEN] = 1) 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 SC[VREFST] = 1.

In this mode, a 100 nF capacitor is required to connect between the VREF_OUT pin and VSSA.

#### 38.3.2.4 SC[MODE_LV] = 11

Reserved

## 38.4 Initialization/Application Information

The Voltage Reference requires some time for startup and stabilization. After SC[VREFEN] = 1, SC[VREFST] can be monitored to determine if the stabilization and startup is completed. The settling time of internal bandgap reference is typically 35ms, which means there is 35ms delay after the bandgap is enabled, the SC[VREFST] can only be useful after the bandgap is stable after the settling time.

When the Voltage Reference is already enabled and stabilized, changing SC[MODE_LV] will not clear SC[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 TRM[CHOPEN], SC[REGEN] and SC[ICOMPEN] bits are written to 1 during factory trimming of the VREF voltage. These bits should be written to 1 to achieve the perfromance stated in the device data sheet.

# Chapter 39
# Programmable Delay Block (PDB)

## 39.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The Programmable Delay Block (PDB) provides controllable delays from either an internal or an external trigger, or a programmable interval tick, to the hardware trigger inputs of ADCs and/or generates the interval triggers to DACs, so that the precise timing between ADC conversions and/or DAC updates can be achieved. The PDB can optionally provide pulse outputs (Pulse-Out's) that are used as the sample window in the CMP block.

### 39.1.1 Features

- Up to 15 trigger input sources and one software trigger source
- Up to eight configurable PDB channels for ADC hardware trigger
  - One PDB channel is associated with one ADC
  - One trigger output for ADC hardware trigger and up to eight pre-trigger outputs for ADC trigger select per PDB channel
  - Trigger outputs can be enabled or disabled independently
  - One 16-bit delay register per pre-trigger output
  - Optional bypass of the delay registers of the pre-trigger outputs
  - Operation in One-Shot or Continuous modes

  - Optional back-to-back mode operation, which enables the ADC conversions complete to trigger the next PDB channel
  - One programmable delay interrupt
  - One sequence error interrupt
  - One channel flag and one sequence error flag per pre-trigger
  - DMA support
- Up to eight DAC interval triggers
  - One interval trigger output per DAC
  - One 16-bit delay interval register per DAC trigger output
  - Optional bypass of the delay interval trigger registers
  - Optional external triggers
- Up to eight pulse outputs (pulse-out's)
  - Pulse-out's can be enabled or disabled independently
  - Programmable pulse width

**NOTE**

The number of PDB input and output triggers are chip-specific. See the chip configuration information for details.

### 39.1.2 Implementation

In this section, the following letters refer to the number of output triggers:

- N—Total available number of PDB channels.
- n—PDB channel number, valid from 0 to *N*-1.
- M—Total available pre-trigger per PDB channel.
- m—Pre-trigger number, valid from 0 to *M*-1.
- X—Total number of DAC interval triggers.
- x—DAC interval trigger output number, valid from 0 to *X*-1.

- Y—Total number of Pulse-Out's.
- y—Pulse-Out number, valid value is from 0 to *Y*-1.

**NOTE**

The number of module output triggers to core is chip-specific. For module to core output triggers implementation, see the chip configuration information.

### 39.1.3 Back-to-back acknowledgment connections

PDB back-to-back operation acknowledgment connections are chip-specific. For implementation, see the chip configuration information.

### 39.1.4 DAC External Trigger Input Connections

The implementation of DAC external trigger inputs is chip-specific. See the chip configuration information for details.

### 39.1.5 Block diagram

This diagram illustrates the major components of the PDB.

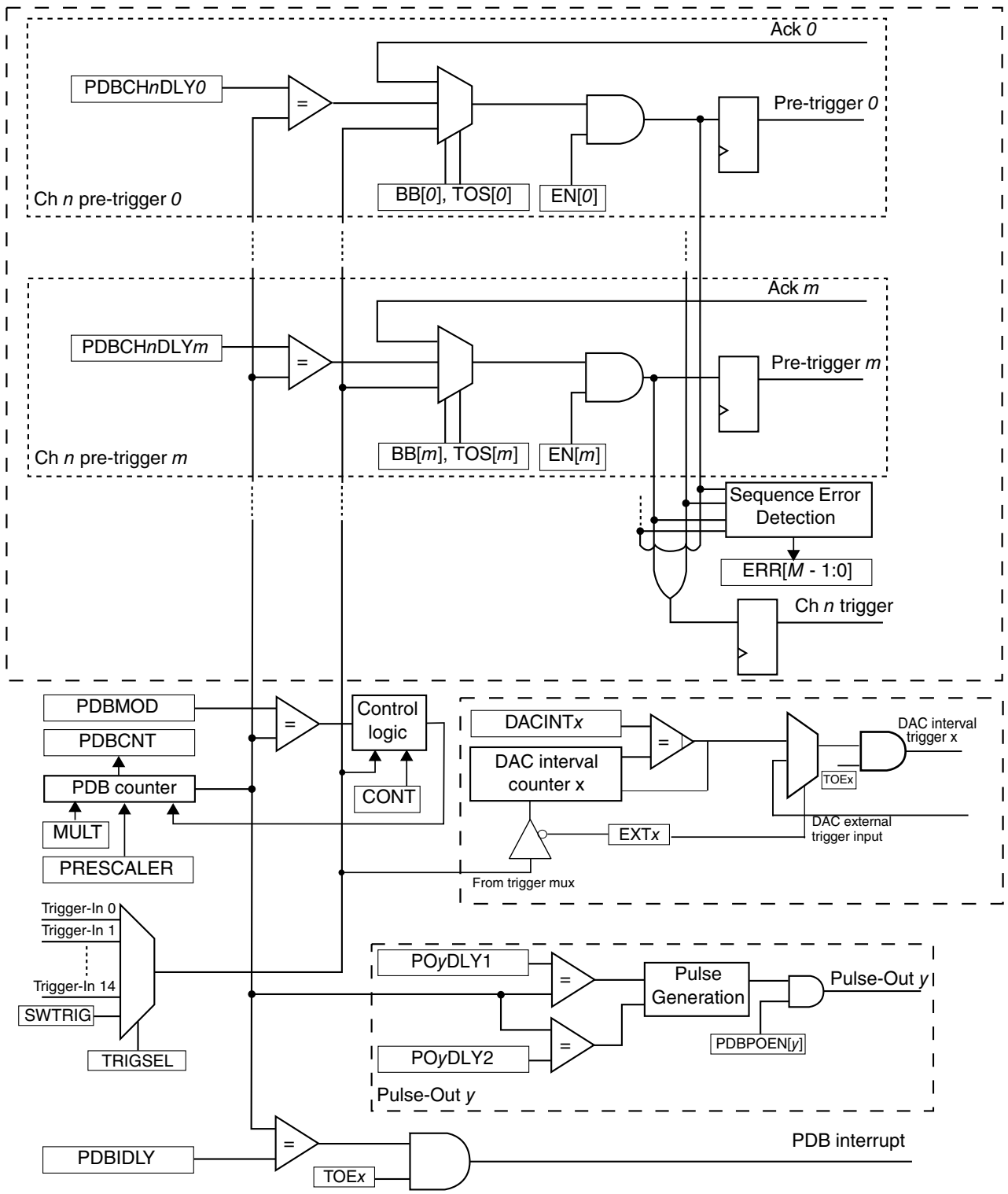



**Figure 39-1. PDB block diagram**

In this diagram, only one PDB channel *n*, one DAC interval trigger *x*, and one Pulse-Out *y* are shown. The PDB-enabled control logic and the sequence error interrupt logic are not shown.

### 39.1.6 Modes of operation

PDB ADC trigger operates in the following modes:

- Disabled—Counter is off, all pre-trigger and trigger outputs are low if PDB is not in back-to-back operation of Bypass mode.
- Debug—Counter is paused when processor is in Debug mode, and the counter for the DAC trigger is also paused in Debug mode.
- Enabled One-Shot—Counter is enabled and restarted at count zero upon receiving a positive edge on the selected trigger input source or software trigger is selected and SC[SWTRIG] is written with 1. In each PDB channel, an enabled pre-trigger asserts once per trigger input event. The trigger output asserts whenever any of the pre-triggers is asserted.
- Enabled Continuous—Counter is enabled and restarted at count zero. The counter is rolled over to zero again when the count reaches the value specified in the modulus register, and the counting is restarted. This enables a continuous stream of pre-triggers/trigger outputs as a result of a single trigger input event.
- Enabled Bypassed—The pre-trigger and trigger outputs assert immediately after a positive edge on the selected trigger input source or software trigger is selected and SC[SWTRIG] is written with 1, that is the delay registers are bypassed. It is possible to bypass any one or more of the delay registers; therefore, this mode can be used in conjunction with One-Shot or Continuous mode.

## 39.2 PDB signal descriptions

This table shows the detailed description of the external signal.

**Table 39-1. PDB signal descriptions**

| Signal | Description | I/O |
|---|---|---|
| EXTRG | External Trigger Input Source<br>If the PDB is enabled and external trigger input source is selected, a positive edge on the EXTRG signal resets and starts the counter. | I |

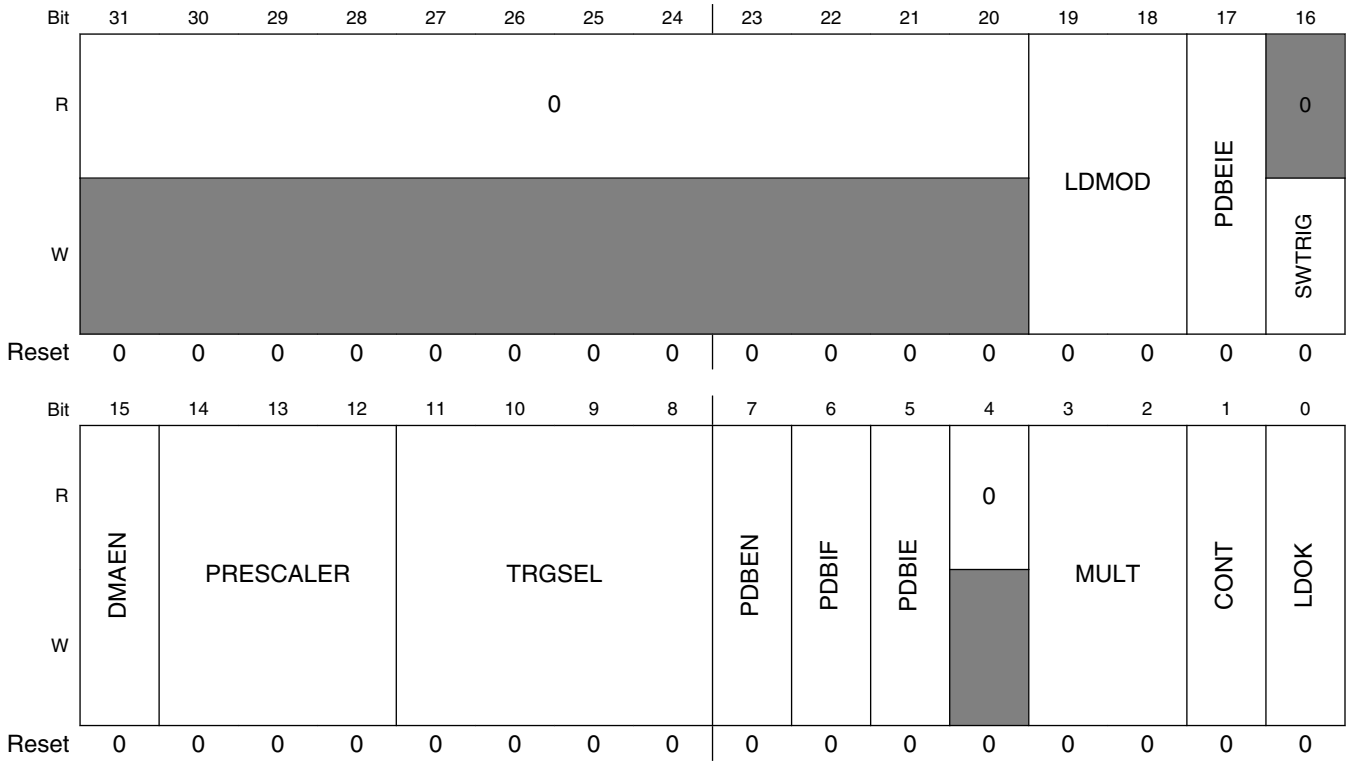
## 39.3 Memory map and register definition

## PDB memory map

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4003_6000 | Status and Control register (PDB0_SC) | 32 | R/W | 0000_0000h | 39.3.1/897 |
| 4003_6004 | Modulus register (PDB0_MOD) | 32 | R/W | 0000_FFFFh | 39.3.2/899 |
| 4003_6008 | Counter register (PDB0_CNT) | 32 | R | 0000_0000h | 39.3.3/900 |
| 4003_600C | Interrupt Delay register (PDB0_IDLY) | 32 | R/W | 0000_FFFFh | 39.3.4/900 |
| 4003_6010 | Channel n Control register 1 (PDB0_CH0C1) | 32 | R/W | 0000_0000h | 39.3.5/901 |
| 4003_6014 | Channel n Status register (PDB0_CH0S) | 32 | R/W | 0000_0000h | 39.3.6/902 |
| 4003_6018 | Channel n Delay 0 register (PDB0_CH0DLY0) | 32 | R/W | 0000_0000h | 39.3.7/902 |
| 4003_601C | Channel n Delay 1 register (PDB0_CH0DLY1) | 32 | R/W | 0000_0000h | 39.3.8/903 |
| 4003_6038 | Channel n Control register 1 (PDB0_CH1C1) | 32 | R/W | 0000_0000h | 39.3.5/901 |
| 4003_603C | Channel n Status register (PDB0_CH1S) | 32 | R/W | 0000_0000h | 39.3.6/902 |
| 4003_6040 | Channel n Delay 0 register (PDB0_CH1DLY0) | 32 | R/W | 0000_0000h | 39.3.7/902 |
| 4003_6044 | Channel n Delay 1 register (PDB0_CH1DLY1) | 32 | R/W | 0000_0000h | 39.3.8/903 |
| 4003_6150 | DAC Interval Trigger n Control register (PDB0_DACINTC0) | 32 | R/W | 0000_0000h | 39.3.9/903 |
| 4003_6154 | DAC Interval n register (PDB0_DACINT0) | 32 | R/W | 0000_0000h | 39.3.10/ 904 |
| 4003_6158 | DAC Interval Trigger n Control register (PDB0_DACINTC1) | 32 | R/W | 0000_0000h | 39.3.9/903 |
| 4003_615C | DAC Interval n register (PDB0_DACINT1) | 32 | R/W | 0000_0000h | 39.3.10/ 904 |
| 4003_6190 | Pulse-Out n Enable register (PDB0_POEN) | 32 | R/W | 0000_0000h | 39.3.11/ 904 |
| 4003_6194 | Pulse-Out n Delay register (PDB0_PO0DLY) | 32 | R/W | 0000_0000h | 39.3.12/ 905 |
| 4003_6198 | Pulse-Out n Delay register (PDB0_PO1DLY) | 32 | R/W | 0000_0000h | 39.3.12/ 905 |
| 4003_619C | Pulse-Out n Delay register (PDB0_PO2DLY) | 32 | R/W | 0000_0000h | 39.3.12/ 905 |

## 39.3.1 Status and Control register (PDBx_SC)

Address: 4003_6000h base + 0h offset = 4003_6000h




### PDBx_SC field descriptions

| Field | Description |
|---|---|
| 31–20<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 19–18<br>LDMOD | Load Mode Select<br><br>Selects the mode to load the MOD, IDLY, CH*n*DLY*m*, INT*x*, and PO*y*DLY registers, after 1 is written to LDOK.<br><br>00 The internal registers are loaded with the values from their buffers immediately after 1 is written to LDOK.<br>01 The internal registers are loaded with the values from their buffers when the PDB counter reaches the MOD register value after 1 is written to LDOK.<br>10 The internal registers are loaded with the values from their buffers when a trigger input event is detected after 1 is written to LDOK.<br>11 The internal registers are loaded with the values from their buffers when either the PDB counter reaches the MOD register value or a trigger input event is detected, after 1 is written to LDOK. |
| 17<br>PDBEIE | PDB Sequence Error Interrupt Enable<br><br>Enables the PDB sequence error interrupt. When this field is set, any of the PDB channel sequence error flags generates a PDB sequence error interrupt. |

*Table continues on the next page...*

## PDBx_SC field descriptions (continued)

| Field | Description |
|---|---|
| | 0 PDB sequence error interrupt disabled.<br>1 PDB sequence error interrupt enabled. |
| 16<br>SWTRIG | Software Trigger<br><br>When PDB is enabled and the software trigger is selected as the trigger input source, writing 1 to this field resets and restarts the counter. Writing 0 to this field has no effect. Reading this field results 0. |
| 15<br>DMAEN | DMA Enable<br><br>When DMA is enabled, the PDBIF flag generates a DMA request instead of an interrupt.<br><br>0 DMA disabled.<br>1 DMA enabled. |
| 14–12<br>PRESCALER | Prescaler Divider Select<br><br>000 Counting uses the peripheral clock divided by multiplication factor selected by MULT.<br>001 Counting uses the peripheral clock divided by twice of the multiplication factor selected by MULT.<br>010 Counting uses the peripheral clock divided by four times of the multiplication factor selected by MULT.<br>011 Counting uses the peripheral clock divided by eight times of the multiplication factor selected by MULT.<br>100 Counting uses the peripheral clock divided by 16 times of the multiplication factor selected by MULT.<br>101 Counting uses the peripheral clock divided by 32 times of the multiplication factor selected by MULT.<br>110 Counting uses the peripheral clock divided by 64 times of the multiplication factor selected by MULT.<br>111 Counting uses the peripheral clock divided by 128 times of the multiplication factor selected by MULT. |
| 11–8<br>TRGSEL | Trigger Input Source Select<br><br>Selects the trigger input source for the PDB. The trigger input source can be internal or external (EXTRG pin), or the software trigger. Please refer to Chip Configuration chapter for the actual PDB input trigger connections.<br><br>0000 Trigger-In 0 is selected.<br>0001 Trigger-In 1 is selected.<br>0010 Trigger-In 2 is selected.<br>0011 Trigger-In 3 is selected.<br>0100 Trigger-In 4 is selected.<br>0101 Trigger-In 5 is selected.<br>0110 Trigger-In 6 is selected.<br>0111 Trigger-In 7 is selected.<br>1000 Trigger-In 8 is selected.<br>1001 Trigger-In 9 is selected.<br>1010 Trigger-In 10 is selected.<br>1011 Trigger-In 11 is selected.<br>1100 Trigger-In 12 is selected.<br>1101 Trigger-In 13 is selected.<br>1110 Trigger-In 14 is selected.<br>1111 Software trigger is selected. |

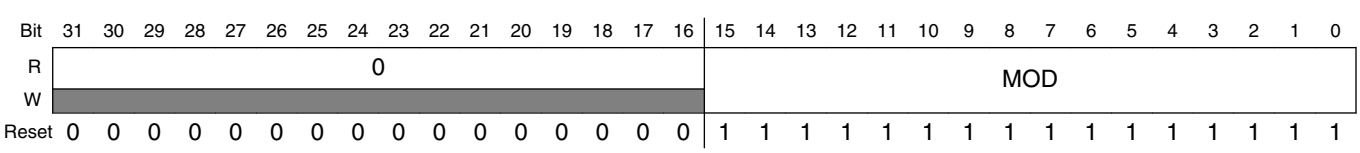
*Table continues on the next page...*

**PDBx_SC field descriptions (continued)**

| Field | Description |
|---|---|
| 7<br>PDBEN | PDB Enable<br><br>0 PDB disabled. Counter is off.<br>1 PDB enabled. |
| 6<br>PDBIF | PDB Interrupt Flag<br><br>This field is set when the counter value is equal to the IDLY register. Writing zero clears this field. |
| 5<br>PDBIE | PDB Interrupt Enable<br><br>Enables the PDB interrupt. When this field is set and DMAEN is cleared, PDBIF generates a PDB interrupt.<br><br>0 PDB interrupt disabled.<br>1 PDB interrupt enabled. |
| 4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3–2<br>MULT | Multiplication Factor Select for Prescaler<br><br>Selects the multiplication factor of the prescaler divider for the counter clock.<br><br>00 Multiplication factor is 1.<br>01 Multiplication factor is 10.<br>10 Multiplication factor is 20.<br>11 Multiplication factor is 40. |
| 1<br>CONT | Continuous Mode Enable<br><br>Enables the PDB operation in Continuous mode.<br><br>0 PDB operation in One-Shot mode<br>1 PDB operation in Continuous mode |
| 0<br>LDOK | Load OK<br><br>Writing 1 to this bit updates the internal registers of MOD, IDLY, CHnDLYm, DACINTx,and POyDLY with the values written to their buffers. The MOD, IDLY, CHnDLYm, DACINTx, and POyDLY will take effect according to the LDMOD.<br><br>After 1 is written to the LDOK field, the values in the buffers of above registers are not effective and the buffers cannot be written until the values in buffers are loaded into their internal registers.<br><br>LDOK can be written only when PDBEN is set or it can be written at the same time with PDBEN being written to 1. It is automatically cleared when the values in buffers are loaded into the internal registers or the PDBEN is cleared. Writing 0 to it has no effect. |

## 39.3.2 Modulus register (PDBx_MOD)

Address: 4003_6000h base + 4h offset = 4003_6004h



### PDBx_MOD field descriptions

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–0<br>MOD | PDB Modulus<br><br>Specifies the period of the counter. When the counter reaches this value, it will be reset back to zero. If the PDB is in Continuous mode, the count begins anew. Reading this field returns the value of the internal register that is effective for the current cycle of PDB. |

## 39.3.3 Counter register (PDBx_CNT)

Address: 4003_6000h base + 8h offset = 4003_6008h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | CNT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### PDBx_CNT field descriptions

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–0<br>CNT | PDB Counter<br><br>Contains the current value of the counter. |

## 39.3.4 Interrupt Delay register (PDBx_IDLY)

Address: 4003_6000h base + Ch offset = 4003_600Ch

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | IDLY | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### PDBx_IDLY field descriptions

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–0<br>IDLY | PDB Interrupt Delay<br><br>Specifies the delay value to schedule the PDB interrupt. It can be used to schedule an independent interrupt at some point in the PDB cycle. If enabled, a PDB interrupt is generated, when the counter is |

*Table continues on the next page...*

**PDBx_IDLY field descriptions (continued)**

| Field | Description |
|---|---|
| | equal to the IDLY. Reading this field returns the value of internal register that is effective for the current cycle of the PDB. |

## 39.3.5 Channel n Control register 1 (PDBx_CH*n*C1)

Each PDB channel has one control register, CHnC1. The bits in this register control the functionality of each PDB channel operation.

Address: 4003_6000h base + 10h offset + (40d × i), where i=0d to 1d

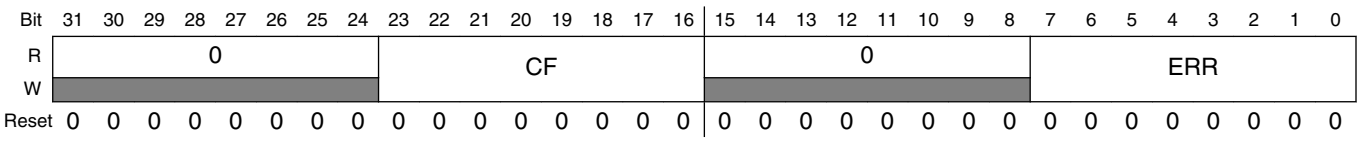
| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | BB | | | | | | | | TOS | | | | | | | | EN | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PDBx_CH*n*C1 field descriptions**

| Field | Description |
|---|---|
| 31–24<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 23–16<br>BB | PDB Channel Pre-Trigger Back-to-Back Operation Enable<br><br>These bits enable the PDB ADC pre-trigger operation as back-to-back mode. Only lower M pre-trigger bits are implemented in this MCU. Back-to-back operation enables the ADC conversions complete to trigger the next PDB channel pre-trigger and trigger output, so that the ADC conversions can be triggered on next set of configuration and results registers. Application code must only enable the back-to-back operation of the PDB pre-triggers at the leading of the back-to-back connection chain.<br><br>0 PDB channel's corresponding pre-trigger back-to-back operation disabled.<br>1 PDB channel's corresponding pre-trigger back-to-back operation enabled. |
| 15–8<br>TOS | PDB Channel Pre-Trigger Output Select<br><br>Selects the PDB ADC pre-trigger outputs. Only lower M pre-trigger fields are implemented in this MCU.<br><br>0 PDB channel's corresponding pre-trigger is in bypassed mode. The pre-trigger asserts one peripheral clock cycle after a rising edge is detected on selected trigger input source or software trigger is selected and SWTRIG is written with 1.<br>1 PDB channel's corresponding pre-trigger asserts when the counter reaches the channel delay register and one peripheral clock cycle after a rising edge is detected on selected trigger input source or software trigger is selected and SETRIG is written with 1. |
| 7–0<br>EN | PDB Channel Pre-Trigger Enable<br><br>These bits enable the PDB ADC pre-trigger outputs. Only lower M pre-trigger bits are implemented in this MCU.<br><br>0 PDB channel's corresponding pre-trigger disabled.<br>1 PDB channel's corresponding pre-trigger enabled. |

## 39.3.6 Channel n Status register (PDBx_CH*n*S)

Address: 4003_6000h base + 14h offset + (40d × i), where i=0d to 1d

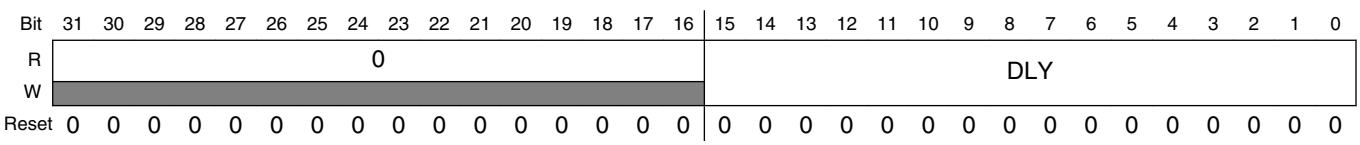



### PDBx_CH*n*S field descriptions

| Field | Description |
|---|---|
| 31–24<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 23–16<br>CF | PDB Channel Flags<br><br>The CF[*m*] bit is set when the PDB counter matches the CH*n*DLY*m*. Write 0 to clear these bits. |
| 15–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7–0<br>ERR | PDB Channel Sequence Error Flags<br><br>Only the lower M bits are implemented in this MCU.<br><br>0 Sequence error not detected on PDB channel's corresponding pre-trigger.<br>1 Sequence error detected on PDB channel's corresponding pre-trigger. ADCn block can be triggered for a conversion by one pre-trigger from PDB channel *n*. When one conversion, which is triggered by one of the pre-triggers from PDB channel *n*, is in progress, new trigger from PDB channel's corresponding pre-trigger m cannot be accepted by ADCn, and ERR[m] is set. Writing 0's to clear the sequence error flags. |

## 39.3.7 Channel n Delay 0 register (PDBx_CH*n*DLY0)

Address: 4003_6000h base + 18h offset + (40d × i), where i=0d to 1d

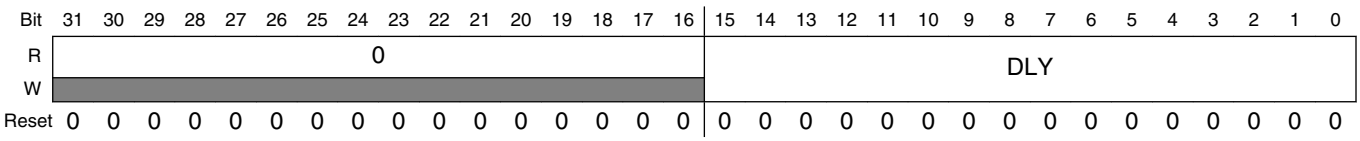



### PDBx_CH*n*DLY0 field descriptions

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–0<br>DLY | PDB Channel Delay<br><br>Specifies the delay value for the channel's corresponding pre-trigger. The pre-trigger asserts when the counter is equal to DLY. Reading this field returns the value of internal register that is effective for the current PDB cycle. |

## 39.3.8 Channel n Delay 1 register (PDB*x*_CH*n*DLY1)

Address: 4003_6000h base + 1Ch offset + (40d × i), where i=0d to 1d

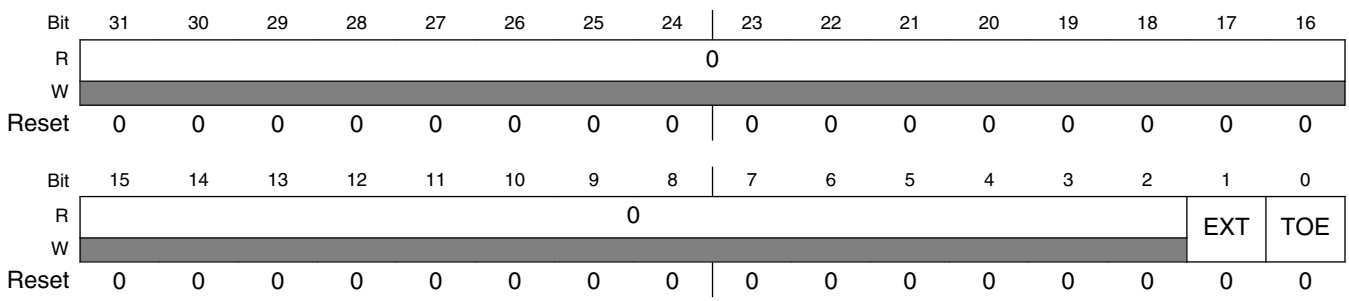



### PDB*x*_CH*n*DLY1 field descriptions

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–0<br>DLY | PDB Channel Delay<br><br>These bits specify the delay value for the channel's corresponding pre-trigger. The pre-trigger asserts when the counter is equal to DLY. Reading these bits returns the value of internal register that is effective for the current PDB cycle. |

## 39.3.9 DAC Interval Trigger n Control register (PDB*x*_DACINTC*n*)

Address: 4003_6000h base + 150h offset + (8d × i), where i=0d to 1d




### PDB*x*_DACINTC*n* field descriptions

| Field | Description |
|---|---|
| 31–2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 1<br>EXT | DAC External Trigger Input Enable<br><br>Enables the external trigger for DAC interval counter.<br><br>0 DAC external trigger input disabled. DAC interval counter is reset and counting starts when a rising edge is detected on selected trigger input source or software trigger is selected and SWTRIG is written with 1.<br>1 DAC external trigger input enabled. DAC interval counter is bypassed and DAC external trigger input triggers the DAC interval trigger. |
| 0<br>TOE | DAC Interval Trigger Enable |

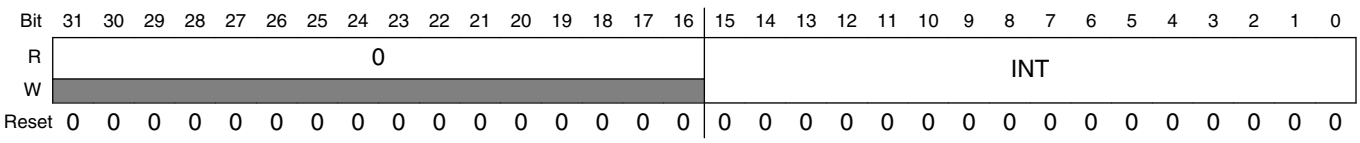
*Table continues on the next page...*

**PDBx_DACINTC*n* field descriptions (continued)**

| Field | Description |
|---|---|
| | This bit enables the DAC interval trigger.<br><br>0 DAC interval trigger disabled.<br>1 DAC interval trigger enabled. |

## 39.3.10 DAC Interval n register (PDBx_DACINT*n*)

Address: 4003_6000h base + 154h offset + (8d × i), where i=0d to 1d

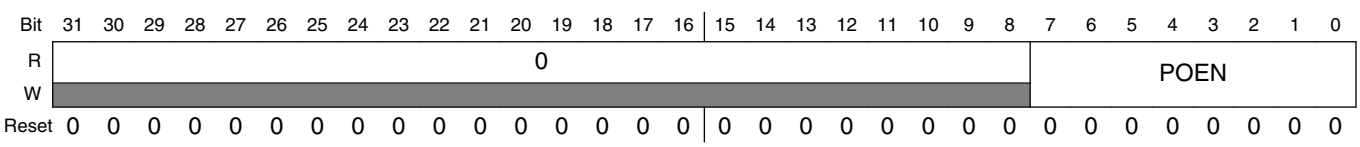



**PDBx_DACINT*n* field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–0<br>INT | DAC Interval<br><br>Specifies the interval value for DAC interval trigger. DAC interval trigger triggers DAC[1:0] update when the DAC interval counter is equal to the DACINT. Reading this field returns the value of internal register that is effective for the current PDB cycle. |

## 39.3.11 Pulse-Out n Enable register (PDBx_POEN)

Address: 4003_6000h base + 190h offset = 4003_6190h




**PDBx_POEN field descriptions**

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7–0<br>POEN | PDB Pulse-Out Enable<br><br>Enables the pulse output. Only lower Y bits are implemented in this MCU.<br><br>0 PDB Pulse-Out disabled<br>1 PDB Pulse-Out enabled |

### 39.3.12 Pulse-Out n Delay register (PDBx_POnDLY)

Address: 4003_6000h base + 194h offset + (4d × i), where i=0d to 2d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DLY1 | | | | | | | | | | | | | | | | DLY2 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PDBx_POnDLY field descriptions**

| Field | Description |
|---|---|
| 31–16<br>DLY1 | PDB Pulse-Out Delay 1<br><br>These bits specify the delay 1 value for the PDB Pulse-Out. Pulse-Out goes high when the PDB counter is equal to the DLY1. Reading these bits returns the value of internal register that is effective for the current PDB cycle. |
| 15–0<br>DLY2 | PDB Pulse-Out Delay 2<br><br>These bits specify the delay 2 value for the PDB Pulse-Out. Pulse-Out goes low when the PDB counter is equal to the DLY2. Reading these bits returns the value of internal register that is effective for the current PDB cycle. |

## 39.4 Functional description

### 39.4.1 PDB pre-trigger and trigger outputs

The PDB contains a counter whose output is compared against several different digital values. If the PDB is enabled, a trigger input event will reset the counter and make it start to count. A trigger input event is defined as a rising edge being detected on selected trigger input source or software trigger being selected and SC[SWTRIG] is written with 1. For each channel, delay *m* determines the time between assertion of the trigger input event to the point at which changes in the pre-trigger *m* output signal is initiated. The time is defined as:

- Trigger input event to pre-trigger *m* = (prescaler X multiplication factor X delay *m*) + 2 peripheral clock cycles
- Add one additional peripheral clock cycle to determine the time at which the channel trigger output change.

Each channel is associated with one ADC block. PDB channel n pre-trigger outputs 0 to *M* and trigger output is connected to ADC hardware trigger select and hardware trigger inputs. The pre-triggers are used to precondition the ADC block prior to the actual trigger. The ADC contains *M* sets of configuration and result registers, allowing it to operate in a ping-pong fashion, alternating conversions between *M* different analog sources. The pre-trigger outputs are used to specify which signal will be sampled next. When pre-trigger *m* is asserted, the ADC conversion is triggered with set *m* of the configuration and result registers.

The waveforms shown in the following diagram illustrate the pre-trigger and trigger outputs of PDB channel *n*. The delays can be independently set via the CH*n*DLY*m* registers. And the pre-triggers can be enabled or disabled in CH*n*C1[EN[*m*]].

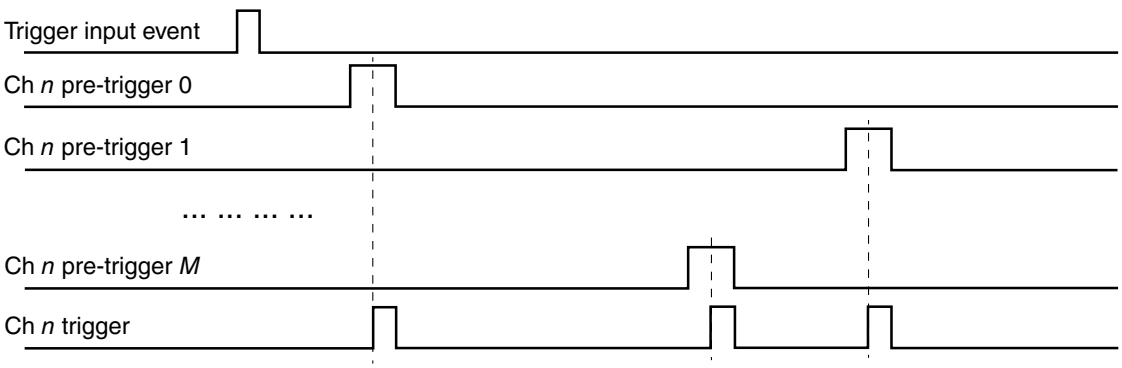



**Figure 39-56. Pre-trigger and trigger outputs**

The delay in CH*n*DLY*m* register can be optionally bypassed, if CH*n*C1[TOS[*m*]] is cleared. In this case, when the trigger input event occurs, the pre-trigger *m* is asserted after two peripheral clock cycles.

The PDB can be configured in back-to-back operation. Back-to-back operation enables the ADC conversions complete to trigger the next PDB channel pre-trigger and trigger outputs, so that the ADC conversions can be triggered on the next set of configuration and results registers. When back-to-back is enabled by setting CH*n*C1[BB[*m*]], the delay *m* is ignored and the pre-trigger *m* is asserted two peripheral cycles after the acknowledgment *m* is received. The acknowledgment connections in this MCU is described in Back-to-back acknowledgment connections.

When an ADC conversion, which is triggered by one of the pre-triggers from PDB channel *n*, is in progress and ADC*n*SC1[COCO] is not set, a new trigger from PDB channel *n* pre-trigger *m* cannot be accepted by ADC*n*. Therefore every time when one PDB channel *n* pre-trigger and trigger output starts an ADC conversion, an internal lock associated with the corresponding pre-trigger is activated. This lock becomes inactive when:

- the corresponding ADC*n*SC1[COCO] is set

- the corresponding PDB pre-trigger is disabled, or
- the PDB is disabled

The channel *n* trigger output is suppressed when any of the locks of the pre-triggers in channel *n* is active. If a new pre-trigger *m* asserts when there is active lock in the PDB channel *n*, a register flag bit, CH*n*S[ERR[*m*]], associated with the pre-trigger *m* is set. If SC[PDBEIE] is set, the sequence error interrupt is generated. Sequence error typically happens because the delay *m* is set too short and the pre-trigger *m* asserts before the previously triggered ADC conversion is completed.

When the PDB counter reaches the value set in IDLY register, the SC[PDBIF] flag is set. A PDB interrupt can be generated if SC[PDBIE] is set and SC[DMAEN] is cleared. If SC[DMAEN] is set, PDB requests a DMA transfer when SC[PDBIF] is set.

The modulus value in the MOD register is used to reset the counter back to zero at the end of the count. If SC[CONT] is set, the counter will then resume a new count. Otherwise, the counter operation will cease until the next trigger input event occurs.

### 39.4.2 PDB trigger input source selection

The PDB has up to 15 trigger input sources, namely Trigger-In 14. They are connected to on-chip or off-chip event sources. The PDB can be triggered by software through SC[SWTRIG]. SC[TRIGSEL] selects the active trigger input source or software trigger.

For the trigger input sources implemented in this MCU, see chip configuration information.

### 39.4.3 DAC interval trigger outputs

PDB can generate the interval triggers for DACs to update their outputs periodically. DAC interval counter *x* is reset and started when a trigger input event occurs if DACINTC*x*[EXT] is cleared. When the interval counter *x* is equal to the value set in DACINT*x* register, the DAC interval trigger *x* output generates a pulse of one peripheral clock cycle width to update the DAC*x*. If DACINTC*x*[EXT] is set, the DAC interval counter is bypassed and the interval trigger output *x* generates a pulse following the detection of a rising edge on the DAC external trigger input. The counter and interval trigger can be disabled by clearing the DACINTC*x*[TOE].

DAC interval counters are also reset when the PDB counter reaches the MOD register value; therefore, when the PDB counter rolls over to zero, the DAC interval counters starts anew.

The DAC interval trigger pulse and the ADC pre-trigger/trigger pulses together allow precise timing of DAC updates and ADC measurements. This is outlined in the typical use case described in the following diagram.

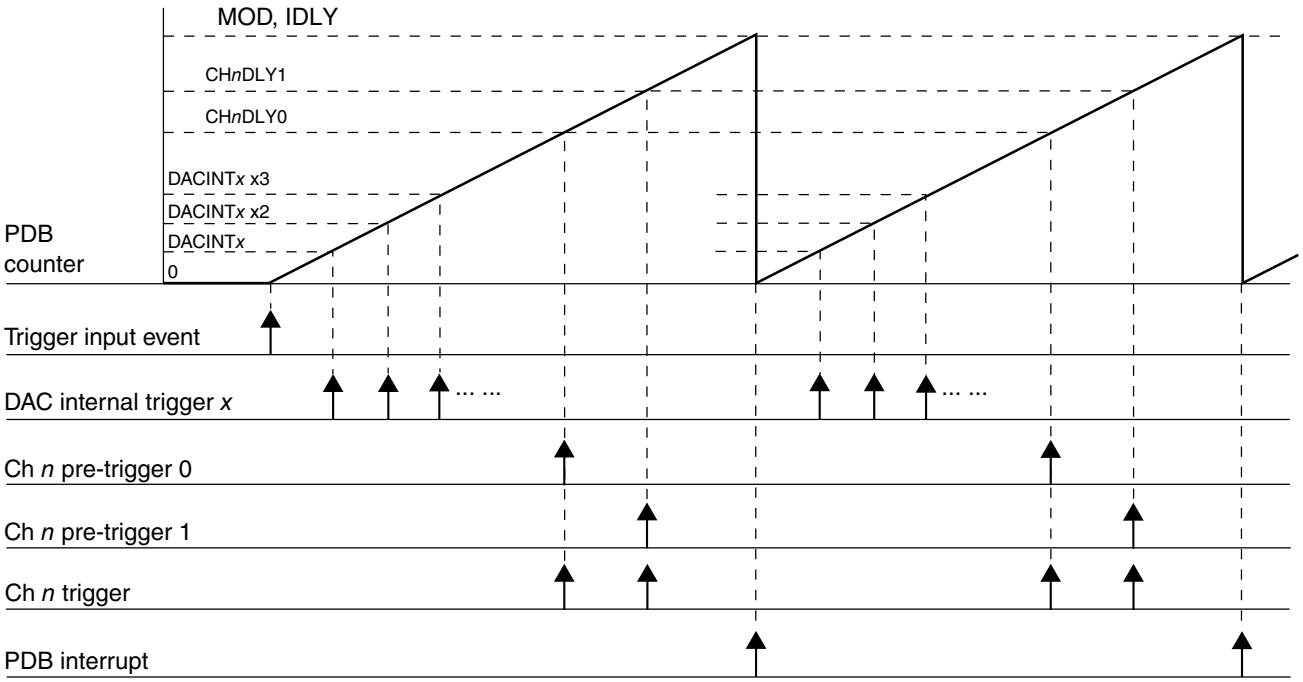



**Figure 39-57. PDB ADC triggers and DAC interval triggers use case**

**NOTE**

Because the DAC interval counters share the prescaler with PDB counter, PDB must be enabled if the DAC interval trigger outputs are used in the applications.

## 39.4.4 Pulse-Out's

PDB can generate pulse outputs of configurable width. When PDB counter reaches the value set in POyDLY[DLY1], the Pulse-Out goes high; when the counter reaches POyDLY[DLY2], it goes low. POyDLY[DLY2] can be set either greater or less than POyDLY[DLY1].

ADC pre-trigger/trigger outputs and Pulse-Out generation have the same time base, because they both share the PDB counter.

The pulse-out connections implemented in this MCU are described in the device's chip configuration details.

### 39.4.5 Updating the delay registers

The following registers control the timing of the PDB operation; and in some of the applications, they may need to become effective at the same time.

- PDB Modulus register (MOD)
- PDB Interrupt Delay register (IDLY)
- PDB Channel *n* Delay *m* register (CH*n*DLY*m*)
- DAC Interval *x* register (DACINT*x*)
- PDB Pulse-Out *y* Delay register (PO*y*DLY)

The internal registers of them are buffered and any values written to them are written first to their buffers. The circumstances that cause their internal registers to be updated with the values from the buffers are summarized as shown in the table below.

**Table 39-58. Circumstances of update to the delay registers**

| SC[LDMOD] | Update to the delay registers |
|---|---|
| 00 | The internal registers are loaded with the values from their buffers immediately after 1 is written to SC[LDOK]. |
| 01 | The PDB counter reaches the MOD register value after 1 is written to SC[LDOK]. |
| 10 | A trigger input event is detected after 1 is written to SC[LDOK]. |
| 11 | Either the PDB counter reaches the MOD register value, or a trigger input event is detected, after 1 is written to SC[LDOK]. |

After 1 is written to SC[LDOK], the buffers cannot be written until the values in buffers are loaded into their internal registers. SC[LDOK] is self-cleared when the internal registers are loaded, so the application code can read it to determine the updates to the internal registers.

The following diagrams show the cases of the internal registers being updated with SC[LDMOD] is 00 and x1.

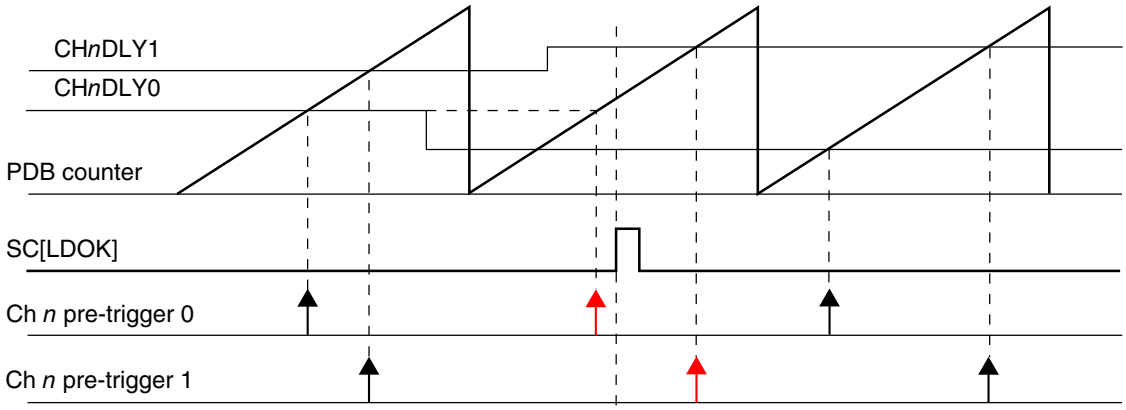



**Figure 39-58. Registers update with SC[LDMOD] = 00**

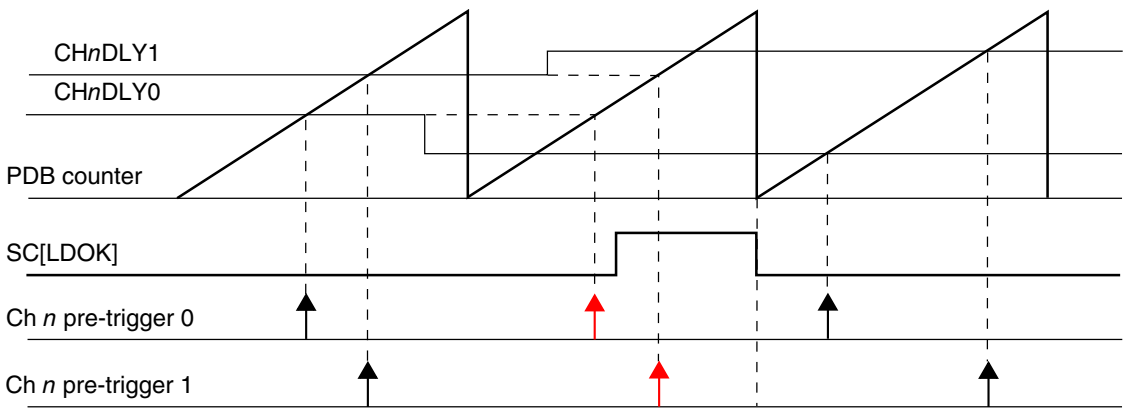



**Figure 39-59. Registers update with SC[LDMOD] = x1**

## 39.4.6 Interrupts

PDB can generate two interrupts: PDB interrupt and PDB sequence error interrupt. The following table summarizes the interrupts.

**Table 39-59. PDB interrupt summary**

| Interrupt | Flags | Enable bit |
|---|---|---|
| PDB Interrupt | SC[PDBIF] | SC[PDBIE] = 1 and SC[DMAEN] = 0 |
| PDB Sequence Error Interrupt | CH*n*S[ERR*m*] | SC[PDBEIE] = 1 |

## 39.4.7 DMA

If SC[DMAEN] is set, PDB can generate a DMA transfer request when SC[PDBIF] is set. When DMA is enabled, the PDB interrupt is not issued.

## 39.5 Application information

### 39.5.1 Impact of using the prescaler and multiplication factor on timing resolution

Use of prescaler and multiplication factor greater than 1 limits the count/delay accuracy in terms of peripheral clock cycles (to the modulus of the prescaler X multiplication factor). If the multiplication factor is set to 1 and the prescaler is set to 2 then the only values of total peripheral clocks that can be detected are even values; if prescaler is set to 4 then the only values of total peripheral clocks that can be decoded as detected are mod(4) and so forth. If the applications need a really long delay value and use a prescaler set to 128, then the resolution would be limited to 128 peripheral clock cycles.

Therefore, use the lowest possible prescaler and multiplication factor for a given application.

# Chapter 40
# FlexTimer Module (FTM)

## 40.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The FlexTimer module (FTM) is a two-to-eight channel timer that supports input capture, output compare, and the generation of PWM signals to control electric motor and power management applications. The FTM time reference is a 16-bit counter that can be used as an unsigned or signed counter.

### 40.1.1 FlexTimer philosophy

The FlexTimer is built upon a simple timer, the HCS08 Timer PWM Module – TPM, used for many years on Freescale's 8-bit microcontrollers. The FlexTimer extends the functionality to meet the demands of motor control, digital lighting solutions, and power conversion, while providing low cost and backwards compatibility with the TPM module.

Several key enhancements are made:

- Signed up counter
- Deadtime insertion hardware
- Fault control inputs
- Enhanced triggering functionality
- Initialization and polarity control

All of the features common with the TPM have fully backwards compatible register assignments. The FlexTimer can also use code on the same core platform without change to perform the same functions.

Motor control and power conversion features have been added through a dedicated set of registers and defaults turn off all new features. The new features, such as hardware deadtime insertion, polarity, fault control, and output forcing and masking, greatly reduce loading on the execution software and are usually each controlled by a group of registers.

FlexTimer input triggers can be from comparators, ADC, or other submodules to initiate timer functions automatically. These triggers can be linked in a variety of ways during integration of the sub modules so please note the options available for used FlexTimer configuration.

Several FlexTimers may be synchronized to provide a larger timer with their counters incrementing in unison, assuming the initialization, the input clocks, the initial and final counting values are the same in each FlexTimer.

All main user access registers are buffered to ease the load on the executing software. A number of trigger options exist to determine which registers are updated with this user defined data.

### 40.1.2 Features

The FTM features include:

- FTM source clock is selectable
  - Source clock can be the system clock, the fixed frequency clock, or an external clock
  - Fixed frequency clock is an additional clock input to allow the selection of an on chip clock source other than the system clock
  - Selecting external clock connects FTM clock to a chip level input pin therefore allowing to synchronize the FTM counter with an off chip clock source
- Prescaler divide-by 1, 2, 4, 8, 16, 32, 64, or 128
- 16-bit counter
  - It can be a free-running counter or a counter with initial and final value
  - The counting can be up or up-down
- Each channel can be configured for input capture, output compare, or edge-aligned PWM mode
- In Input Capture mode:

  - The capture can occur on rising edges, falling edges or both edges
  - An input filter can be selected for some channels
- In Output Compare mode the output signal can be set, cleared, or toggled on match
- All channels can be configured for center-aligned PWM mode
- Each pair of channels can be combined to generate a PWM signal with independent control of both edges of PWM signal
- The FTM channels can operate as pairs with equal outputs, pairs with complementary outputs, or independent channels with independent outputs
- The deadtime insertion is available for each complementary pair
- Generation of match triggers
- Software control of PWM outputs
- Up to 4 fault inputs for global fault control
- The polarity of each channel is configurable
- The generation of an interrupt per channel
- The generation of an interrupt when the counter overflows
- The generation of an interrupt when the fault condition is detected
- Synchronized loading of write buffered FTM registers
- Write protection for critical registers
- Backwards compatible with TPM
- Testing of input captures for a stuck at zero and one conditions
- Dual edge capture for pulse and period width measurement
- Quadrature decoder with input filters, relative position counting, and interrupt on position count or capture of position count on external event

### 40.1.3 Modes of operation

When the MCU is in an active BDM mode, the FTM temporarily suspends all counting until the MCU returns to normal user operating mode. During Stop mode, all FTM input clocks are stopped, so the FTM is effectively disabled until clocks resume. During Wait mode, the FTM continues to operate normally. If the FTM does not need to produce a

real time reference or provide the interrupt sources needed to wake the MCU from Wait mode, the power can then be saved by disabling FTM functions before entering Wait mode.

### 40.1.4 Block diagram

The FTM uses one input/output (I/O) pin per channel, CHn (FTM channel (n)) where n is the channel number (0–7).

The following figure shows the FTM structure. The central component of the FTM is the 16-bit counter with programmable initial and final values and its counting can be up or up-down.

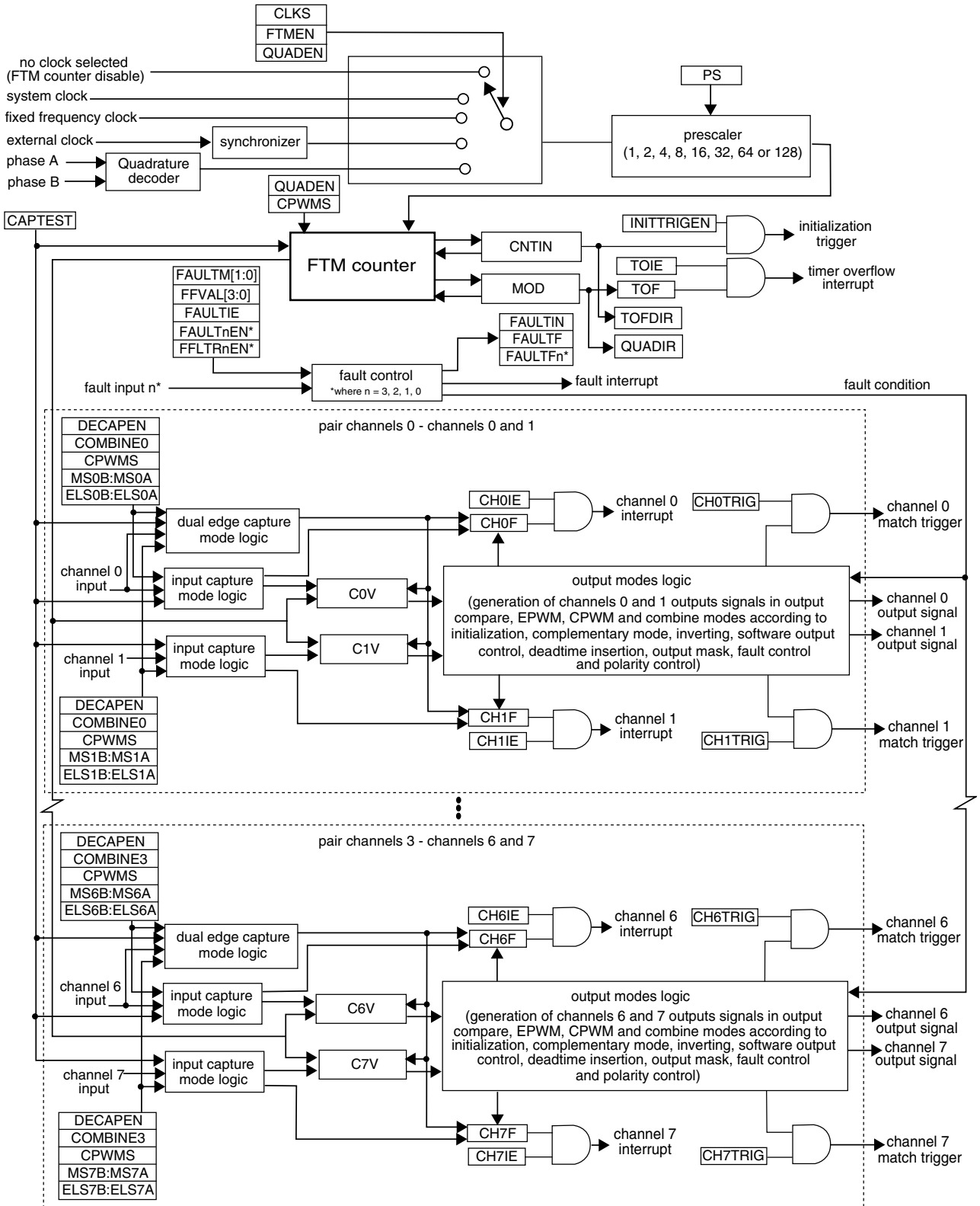



**Figure 40-1. FTM block diagram**

## 40.2 FTM signal descriptions

Table 40-1 shows the user-accessible signals for the FTM.

**Table 40-1. FTM signal descriptions**

| Signal | Description | I/O | Function |
|---|---|---|---|
| EXTCLK | External clock. FTM external clock can be selected to drive the FTM counter. | I | The external clock input signal is used as the FTM counter clock if selected by CLKS[1:0] bits in the SC register. This clock signal must not exceed 1/4 of system clock frequency. The FTM counter prescaler selection and settings are also used when an external clock is selected. |
| CHn | FTM channel (n), where n can be 7-0 | I/O | Each FTM channel can be configured to operate either as input or output. The direction associated with each channel, input or output, is selected according to the mode assigned for that channel. |
| FAULTj | Fault input (j), where j can be 3-0 | I | The fault input signals are used to control the CHn channel output state. If a fault is detected, the FAULTj signal is asserted and the channel output is put in a safe state. The behavior of the fault logic is defined by the FAULTM[1:0] control bits in the MODE register and FAULTEN bit in the COMBINEm register. Note that each FAULTj input may affect all channels selectively since FAULTM[1:0] and FAULTEN control bits are defined for each pair of channels. Because there are several FAULTj inputs, maximum of 4 for the FTM module, each one of these inputs is activated by the FAULTjEN bit in the FLTCTRL register. |
| PHA | Quadrature decoder phase A input. Input pin associated with quadrature decoder phase A. | I | The quadrature decoder phase A input is used as the Quadrature Decoder mode is selected. The phase A input signal is one of the signals that control the FTM counter increment or decrement in the Quadrature Decoder mode. |
| PHB | Quadrature decoder phase B input. Input pin associated with quadrature decoder phase B. | I | The quadrature decoder phase B input is used as the Quadrature Decoder mode is selected. The phase B input signal is one of the signals that control the FTM counter increment or decrement in the Quadrature Decoder mode. |

## 40.3 Memory map and register definition

### 40.3.1 Memory map

This section presents a high-level summary of the FTM registers and how they are mapped.

The registers and bits of an unavailable function in the FTM remain in the memory map and in the reset value, but they have no active function.

**Note**

Do not write in the region from the CNTIN register through the PWMLOAD register when FTMEN = 0.

## 40.3.2 Register descriptions

Accesses to reserved addresses result in transfer errors. Registers for absent channels are considered reserved.

**FTM memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4003_8000 | Status And Control (FTM0_SC) | 32 | R/W | 0000_0000h | 40.3.3/925 |
| 4003_8004 | Counter (FTM0_CNT) | 32 | R/W | 0000_0000h | 40.3.4/926 |
| 4003_8008 | Modulo (FTM0_MOD) | 32 | R/W | 0000_0000h | 40.3.5/927 |
| 4003_800C | Channel (n) Status And Control (FTM0_C0SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 4003_8010 | Channel (n) Value (FTM0_C0V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 4003_8014 | Channel (n) Status And Control (FTM0_C1SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 4003_8018 | Channel (n) Value (FTM0_C1V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 4003_801C | Channel (n) Status And Control (FTM0_C2SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 4003_8020 | Channel (n) Value (FTM0_C2V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 4003_8024 | Channel (n) Status And Control (FTM0_C3SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 4003_8028 | Channel (n) Value (FTM0_C3V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 4003_802C | Channel (n) Status And Control (FTM0_C4SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 4003_8030 | Channel (n) Value (FTM0_C4V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 4003_8034 | Channel (n) Status And Control (FTM0_C5SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 4003_8038 | Channel (n) Value (FTM0_C5V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 4003_803C | Channel (n) Status And Control (FTM0_C6SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 4003_8040 | Channel (n) Value (FTM0_C6V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 4003_8044 | Channel (n) Status And Control (FTM0_C7SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 4003_8048 | Channel (n) Value (FTM0_C7V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 4003_804C | Counter Initial Value (FTM0_CNTIN) | 32 | R/W | 0000_0000h | 40.3.8/931 |
| 4003_8050 | Capture And Compare Status (FTM0_STATUS) | 32 | R/W | 0000_0000h | 40.3.9/931 |
| 4003_8054 | Features Mode Selection (FTM0_MODE) | 32 | R/W | 0000_0004h | 40.3.10/ 933 |
| 4003_8058 | Synchronization (FTM0_SYNC) | 32 | R/W | 0000_0000h | 40.3.11/ 935 |
| 4003_805C | Initial State For Channels Output (FTM0_OUTINIT) | 32 | R/W | 0000_0000h | 40.3.12/ 938 |
| 4003_8060 | Output Mask (FTM0_OUTMASK) | 32 | R/W | 0000_0000h | 40.3.13/ 939 |

*Table continues on the next page...*

## FTM memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4003_8064 | Function For Linked Channels (FTM0_COMBINE) | 32 | R/W | 0000_0000h | 40.3.14/ 941 |
| 4003_8068 | Deadtime Insertion Control (FTM0_DEADTIME) | 32 | R/W | 0000_0000h | 40.3.15/ 946 |
| 4003_806C | FTM External Trigger (FTM0_EXTTRIG) | 32 | R/W | 0000_0000h | 40.3.16/ 947 |
| 4003_8070 | Channels Polarity (FTM0_POL) | 32 | R/W | 0000_0000h | 40.3.17/ 949 |
| 4003_8074 | Fault Mode Status (FTM0_FMS) | 32 | R/W | 0000_0000h | 40.3.18/ 951 |
| 4003_8078 | Input Capture Filter Control (FTM0_FILTER) | 32 | R/W | 0000_0000h | 40.3.19/ 953 |
| 4003_807C | Fault Control (FTM0_FLTCTRL) | 32 | R/W | 0000_0000h | 40.3.20/ 954 |
| 4003_8080 | Quadrature Decoder Control And Status (FTM0_QDCTRL) | 32 | R/W | 0000_0000h | 40.3.21/ 956 |
| 4003_8084 | Configuration (FTM0_CONF) | 32 | R/W | 0000_0000h | 40.3.22/ 958 |
| 4003_8088 | FTM Fault Input Polarity (FTM0_FLTPOL) | 32 | R/W | 0000_0000h | 40.3.23/ 959 |
| 4003_808C | Synchronization Configuration (FTM0_SYNCONF) | 32 | R/W | 0000_0000h | 40.3.24/ 961 |
| 4003_8090 | FTM Inverting Control (FTM0_INVCTRL) | 32 | R/W | 0000_0000h | 40.3.25/ 963 |
| 4003_8094 | FTM Software Output Control (FTM0_SWOCTRL) | 32 | R/W | 0000_0000h | 40.3.26/ 964 |
| 4003_8098 | FTM PWM Load (FTM0_PWMLOAD) | 32 | R/W | 0000_0000h | 40.3.27/ 966 |
| 4003_9000 | Status And Control (FTM1_SC) | 32 | R/W | 0000_0000h | 40.3.3/925 |
| 4003_9004 | Counter (FTM1_CNT) | 32 | R/W | 0000_0000h | 40.3.4/926 |
| 4003_9008 | Modulo (FTM1_MOD) | 32 | R/W | 0000_0000h | 40.3.5/927 |
| 4003_900C | Channel (n) Status And Control (FTM1_C0SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 4003_9010 | Channel (n) Value (FTM1_C0V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 4003_9014 | Channel (n) Status And Control (FTM1_C1SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 4003_9018 | Channel (n) Value (FTM1_C1V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 4003_901C | Channel (n) Status And Control (FTM1_C2SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 4003_9020 | Channel (n) Value (FTM1_C2V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 4003_9024 | Channel (n) Status And Control (FTM1_C3SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 4003_9028 | Channel (n) Value (FTM1_C3V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 4003_902C | Channel (n) Status And Control (FTM1_C4SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 4003_9030 | Channel (n) Value (FTM1_C4V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 4003_9034 | Channel (n) Status And Control (FTM1_C5SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |

*Table continues on the next page...*

## FTM memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4003_9038 | Channel (n) Value (FTM1_C5V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 4003_903C | Channel (n) Status And Control (FTM1_C6SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 4003_9040 | Channel (n) Value (FTM1_C6V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 4003_9044 | Channel (n) Status And Control (FTM1_C7SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 4003_9048 | Channel (n) Value (FTM1_C7V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 4003_904C | Counter Initial Value (FTM1_CNTIN) | 32 | R/W | 0000_0000h | 40.3.8/931 |
| 4003_9050 | Capture And Compare Status (FTM1_STATUS) | 32 | R/W | 0000_0000h | 40.3.9/931 |
| 4003_9054 | Features Mode Selection (FTM1_MODE) | 32 | R/W | 0000_0004h | 40.3.10/ 933 |
| 4003_9058 | Synchronization (FTM1_SYNC) | 32 | R/W | 0000_0000h | 40.3.11/ 935 |
| 4003_905C | Initial State For Channels Output (FTM1_OUTINIT) | 32 | R/W | 0000_0000h | 40.3.12/ 938 |
| 4003_9060 | Output Mask (FTM1_OUTMASK) | 32 | R/W | 0000_0000h | 40.3.13/ 939 |
| 4003_9064 | Function For Linked Channels (FTM1_COMBINE) | 32 | R/W | 0000_0000h | 40.3.14/ 941 |
| 4003_9068 | Deadtime Insertion Control (FTM1_DEADTIME) | 32 | R/W | 0000_0000h | 40.3.15/ 946 |
| 4003_906C | FTM External Trigger (FTM1_EXTTRIG) | 32 | R/W | 0000_0000h | 40.3.16/ 947 |
| 4003_9070 | Channels Polarity (FTM1_POL) | 32 | R/W | 0000_0000h | 40.3.17/ 949 |
| 4003_9074 | Fault Mode Status (FTM1_FMS) | 32 | R/W | 0000_0000h | 40.3.18/ 951 |
| 4003_9078 | Input Capture Filter Control (FTM1_FILTER) | 32 | R/W | 0000_0000h | 40.3.19/ 953 |
| 4003_907C | Fault Control (FTM1_FLTCTRL) | 32 | R/W | 0000_0000h | 40.3.20/ 954 |
| 4003_9080 | Quadrature Decoder Control And Status (FTM1_QDCTRL) | 32 | R/W | 0000_0000h | 40.3.21/ 956 |
| 4003_9084 | Configuration (FTM1_CONF) | 32 | R/W | 0000_0000h | 40.3.22/ 958 |
| 4003_9088 | FTM Fault Input Polarity (FTM1_FLTPOL) | 32 | R/W | 0000_0000h | 40.3.23/ 959 |
| 4003_908C | Synchronization Configuration (FTM1_SYNCONF) | 32 | R/W | 0000_0000h | 40.3.24/ 961 |
| 4003_9090 | FTM Inverting Control (FTM1_INVCTRL) | 32 | R/W | 0000_0000h | 40.3.25/ 963 |
| 4003_9094 | FTM Software Output Control (FTM1_SWOCTRL) | 32 | R/W | 0000_0000h | 40.3.26/ 964 |
| 4003_9098 | FTM PWM Load (FTM1_PWMLOAD) | 32 | R/W | 0000_0000h | 40.3.27/ 966 |

## FTM memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 400B_8000 | Status And Control (FTM2_SC) | 32 | R/W | 0000_0000h | 40.3.3/925 |
| 400B_8004 | Counter (FTM2_CNT) | 32 | R/W | 0000_0000h | 40.3.4/926 |
| 400B_8008 | Modulo (FTM2_MOD) | 32 | R/W | 0000_0000h | 40.3.5/927 |
| 400B_800C | Channel (n) Status And Control (FTM2_C0SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 400B_8010 | Channel (n) Value (FTM2_C0V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 400B_8014 | Channel (n) Status And Control (FTM2_C1SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 400B_8018 | Channel (n) Value (FTM2_C1V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 400B_801C | Channel (n) Status And Control (FTM2_C2SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 400B_8020 | Channel (n) Value (FTM2_C2V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 400B_8024 | Channel (n) Status And Control (FTM2_C3SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 400B_8028 | Channel (n) Value (FTM2_C3V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 400B_802C | Channel (n) Status And Control (FTM2_C4SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 400B_8030 | Channel (n) Value (FTM2_C4V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 400B_8034 | Channel (n) Status And Control (FTM2_C5SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 400B_8038 | Channel (n) Value (FTM2_C5V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 400B_803C | Channel (n) Status And Control (FTM2_C6SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 400B_8040 | Channel (n) Value (FTM2_C6V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 400B_8044 | Channel (n) Status And Control (FTM2_C7SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 400B_8048 | Channel (n) Value (FTM2_C7V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 400B_804C | Counter Initial Value (FTM2_CNTIN) | 32 | R/W | 0000_0000h | 40.3.8/931 |
| 400B_8050 | Capture And Compare Status (FTM2_STATUS) | 32 | R/W | 0000_0000h | 40.3.9/931 |
| 400B_8054 | Features Mode Selection (FTM2_MODE) | 32 | R/W | 0000_0004h | 40.3.10/ 933 |
| 400B_8058 | Synchronization (FTM2_SYNC) | 32 | R/W | 0000_0000h | 40.3.11/ 935 |
| 400B_805C | Initial State For Channels Output (FTM2_OUTINIT) | 32 | R/W | 0000_0000h | 40.3.12/ 938 |
| 400B_8060 | Output Mask (FTM2_OUTMASK) | 32 | R/W | 0000_0000h | 40.3.13/ 939 |
| 400B_8064 | Function For Linked Channels (FTM2_COMBINE) | 32 | R/W | 0000_0000h | 40.3.14/ 941 |
| 400B_8068 | Deadtime Insertion Control (FTM2_DEADTIME) | 32 | R/W | 0000_0000h | 40.3.15/ 946 |
| 400B_806C | FTM External Trigger (FTM2_EXTTRIG) | 32 | R/W | 0000_0000h | 40.3.16/ 947 |
| 400B_8070 | Channels Polarity (FTM2_POL) | 32 | R/W | 0000_0000h | 40.3.17/ 949 |
| 400B_8074 | Fault Mode Status (FTM2_FMS) | 32 | R/W | 0000_0000h | 40.3.18/ 951 |
| 400B_8078 | Input Capture Filter Control (FTM2_FILTER) | 32 | R/W | 0000_0000h | 40.3.19/ 953 |

*Table continues on the next page...*

## FTM memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 400B_807C | Fault Control (FTM2_FLTCTRL) | 32 | R/W | 0000_0000h | 40.3.20/ 954 |
| 400B_8080 | Quadrature Decoder Control And Status (FTM2_QDCTRL) | 32 | R/W | 0000_0000h | 40.3.21/ 956 |
| 400B_8084 | Configuration (FTM2_CONF) | 32 | R/W | 0000_0000h | 40.3.22/ 958 |
| 400B_8088 | FTM Fault Input Polarity (FTM2_FLTPOL) | 32 | R/W | 0000_0000h | 40.3.23/ 959 |
| 400B_808C | Synchronization Configuration (FTM2_SYNCONF) | 32 | R/W | 0000_0000h | 40.3.24/ 961 |
| 400B_8090 | FTM Inverting Control (FTM2_INVCTRL) | 32 | R/W | 0000_0000h | 40.3.25/ 963 |
| 400B_8094 | FTM Software Output Control (FTM2_SWOCTRL) | 32 | R/W | 0000_0000h | 40.3.26/ 964 |
| 400B_8098 | FTM PWM Load (FTM2_PWMLOAD) | 32 | R/W | 0000_0000h | 40.3.27/ 966 |
| 400B_9000 | Status And Control (FTM3_SC) | 32 | R/W | 0000_0000h | 40.3.3/925 |
| 400B_9004 | Counter (FTM3_CNT) | 32 | R/W | 0000_0000h | 40.3.4/926 |
| 400B_9008 | Modulo (FTM3_MOD) | 32 | R/W | 0000_0000h | 40.3.5/927 |
| 400B_900C | Channel (n) Status And Control (FTM3_C0SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 400B_9010 | Channel (n) Value (FTM3_C0V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 400B_9014 | Channel (n) Status And Control (FTM3_C1SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 400B_9018 | Channel (n) Value (FTM3_C1V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 400B_901C | Channel (n) Status And Control (FTM3_C2SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 400B_9020 | Channel (n) Value (FTM3_C2V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 400B_9024 | Channel (n) Status And Control (FTM3_C3SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 400B_9028 | Channel (n) Value (FTM3_C3V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 400B_902C | Channel (n) Status And Control (FTM3_C4SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 400B_9030 | Channel (n) Value (FTM3_C4V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 400B_9034 | Channel (n) Status And Control (FTM3_C5SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 400B_9038 | Channel (n) Value (FTM3_C5V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 400B_903C | Channel (n) Status And Control (FTM3_C6SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 400B_9040 | Channel (n) Value (FTM3_C6V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 400B_9044 | Channel (n) Status And Control (FTM3_C7SC) | 32 | R/W | 0000_0000h | 40.3.6/928 |
| 400B_9048 | Channel (n) Value (FTM3_C7V) | 32 | R/W | 0000_0000h | 40.3.7/930 |
| 400B_904C | Counter Initial Value (FTM3_CNTIN) | 32 | R/W | 0000_0000h | 40.3.8/931 |
| 400B_9050 | Capture And Compare Status (FTM3_STATUS) | 32 | R/W | 0000_0000h | 40.3.9/931 |
| 400B_9054 | Features Mode Selection (FTM3_MODE) | 32 | R/W | 0000_0004h | 40.3.10/ 933 |
| 400B_9058 | Synchronization (FTM3_SYNC) | 32 | R/W | 0000_0000h | 40.3.11/ 935 |

*Table continues on the next page...*

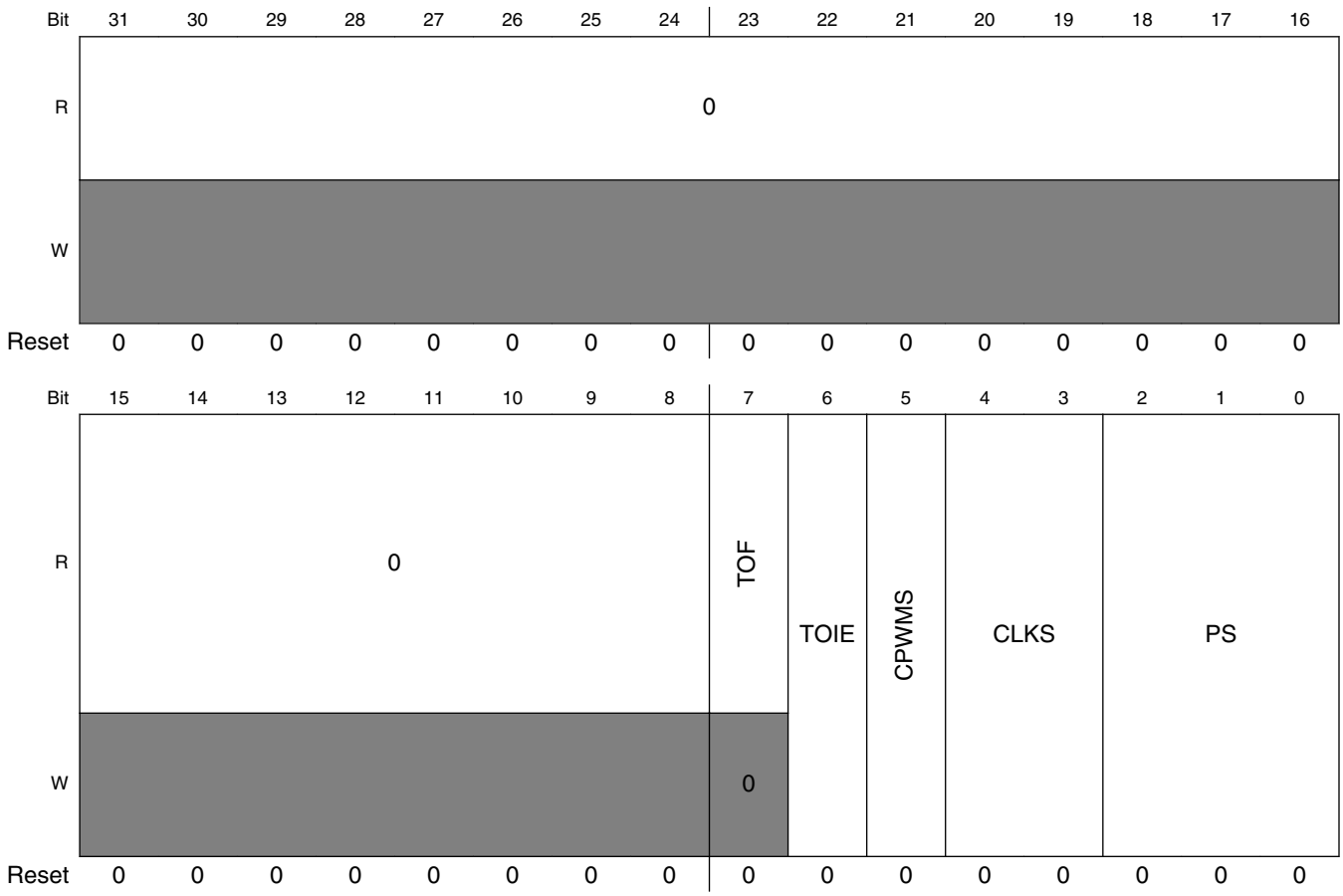
## FTM memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 400B_905C | Initial State For Channels Output (FTM3_OUTINIT) | 32 | R/W | 0000_0000h | 40.3.12/ 938 |
| 400B_9060 | Output Mask (FTM3_OUTMASK) | 32 | R/W | 0000_0000h | 40.3.13/ 939 |
| 400B_9064 | Function For Linked Channels (FTM3_COMBINE) | 32 | R/W | 0000_0000h | 40.3.14/ 941 |
| 400B_9068 | Deadtime Insertion Control (FTM3_DEADTIME) | 32 | R/W | 0000_0000h | 40.3.15/ 946 |
| 400B_906C | FTM External Trigger (FTM3_EXTTRIG) | 32 | R/W | 0000_0000h | 40.3.16/ 947 |
| 400B_9070 | Channels Polarity (FTM3_POL) | 32 | R/W | 0000_0000h | 40.3.17/ 949 |
| 400B_9074 | Fault Mode Status (FTM3_FMS) | 32 | R/W | 0000_0000h | 40.3.18/ 951 |
| 400B_9078 | Input Capture Filter Control (FTM3_FILTER) | 32 | R/W | 0000_0000h | 40.3.19/ 953 |
| 400B_907C | Fault Control (FTM3_FLTCTRL) | 32 | R/W | 0000_0000h | 40.3.20/ 954 |
| 400B_9080 | Quadrature Decoder Control And Status (FTM3_QDCTRL) | 32 | R/W | 0000_0000h | 40.3.21/ 956 |
| 400B_9084 | Configuration (FTM3_CONF) | 32 | R/W | 0000_0000h | 40.3.22/ 958 |
| 400B_9088 | FTM Fault Input Polarity (FTM3_FLTPOL) | 32 | R/W | 0000_0000h | 40.3.23/ 959 |
| 400B_908C | Synchronization Configuration (FTM3_SYNCONF) | 32 | R/W | 0000_0000h | 40.3.24/ 961 |
| 400B_9090 | FTM Inverting Control (FTM3_INVCTRL) | 32 | R/W | 0000_0000h | 40.3.25/ 963 |
| 400B_9094 | FTM Software Output Control (FTM3_SWOCTRL) | 32 | R/W | 0000_0000h | 40.3.26/ 964 |
| 400B_9098 | FTM PWM Load (FTM3_PWMLOAD) | 32 | R/W | 0000_0000h | 40.3.27/ 966 |

## 40.3.3 Status And Control (FTMx_SC)

SC contains the overflow status flag and control bits used to configure the interrupt enable, FTM configuration, clock source, and prescaler factor. These controls relate to all channels within this module.

Address: Base address + 0h offset




**FTMx_SC field descriptions**

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>TOF | Timer Overflow Flag<br>Set by hardware when the FTM counter passes the value in the MOD register. The TOF bit is cleared by reading the SC register while TOF is set and then writing a 0 to TOF bit. Writing a 1 to TOF has no effect.<br>If another FTM overflow occurs between the read and write operations, the write operation has no effect; therefore, TOF remains set indicating an overflow has occurred. In this case, a TOF interrupt request is not lost due to the clearing sequence for a previous TOF.<br>0 FTM counter has not overflowed.<br>1 FTM counter has overflowed. |

*Table continues on the next page...*

**FTMx_SC field descriptions (continued)**

| Field | Description |
|---|---|
| 6<br>TOIE | Timer Overflow Interrupt Enable<br><br>Enables FTM overflow interrupts.<br><br>0 Disable TOF interrupts. Use software polling.<br>1 Enable TOF interrupts. An interrupt is generated when TOF equals one. |
| 5<br>CPWMS | Center-Aligned PWM Select<br><br>Selects CPWM mode. This mode configures the FTM to operate in Up-Down Counting mode.<br><br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br><br>0 FTM counter operates in Up Counting mode.<br>1 FTM counter operates in Up-Down Counting mode. |
| 4–3<br>CLKS | Clock Source Selection<br><br>Selects one of the three FTM counter clock sources.<br><br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br><br>00 No clock selected. This in effect disables the FTM counter.<br>01 System clock<br>10 Fixed frequency clock<br>11 External clock |
| 2–0<br>PS | Prescale Factor Selection<br><br>Selects one of 8 division factors for the clock source selected by CLKS. The new prescaler factor affects the clock source on the next system clock cycle after the new value is updated into the register bits.<br><br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br><br>000 Divide by 1<br>001 Divide by 2<br>010 Divide by 4<br>011 Divide by 8<br>100 Divide by 16<br>101 Divide by 32<br>110 Divide by 64<br>111 Divide by 128 |

## 40.3.4 Counter (FTMx_CNT)

The CNT register contains the FTM counter value.

Reset clears the CNT register. Writing any value to COUNT updates the counter with its initial value, CNTIN.

When BDM is active, the FTM counter is frozen. This is the value that you may read.

Address: Base address + 4h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | COUNT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTMx_CNT field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–0<br>COUNT | Counter Value |

## 40.3.5 Modulo (FTMx_MOD)

The Modulo register contains the modulo value for the FTM counter. After the FTM counter reaches the modulo value, the overflow flag (TOF) becomes set at the next clock, and the next value of FTM counter depends on the selected counting method; see Counter.

Writing to the MOD register latches the value into a buffer. The MOD register is updated with the value of its write buffer according to Registers updated from write buffers.

If FTMEN = 0, this write coherency mechanism may be manually reset by writing to the SC register whether BDM is active or not.

Initialize the FTM counter, by writing to CNT, before writing to the MOD register to avoid confusion about when the first counter overflow will occur.

Address: Base address + 8h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | | | | | | | | | MOD | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTMx_MOD field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved. |
| 15–0<br>MOD | Modulo Value |

## 40.3.6 Channel (n) Status And Control (FTMx_C*n*SC)

CnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function.

**Table 40-67. Mode, edge, and level selection**

| DECAPEN | COMBINE | CPWMS | MSnB:MSnA | ELSnB:ELSnA | Mode | Configuration |
|---|---|---|---|---|---|---|
| X | X | X | XX | 0 | Pin not used for FTM—revert the channel pin to general purpose I/O or other peripheral control | |
| 0 | 0 | 0 | 0 | 1 | Input Capture | Capture on Rising Edge Only |
| | | | | 10 | | Capture on Falling Edge Only |
| | | | | 11 | | Capture on Rising or Falling Edge |
| | | | 1 | 1 | Output Compare | Toggle Output on match |
| | | | | 10 | | Clear Output on match |
| | | | | 11 | | Set Output on match |
| | | | 1X | 10 | Edge-Aligned PWM | High-true pulses (clear Output on match) |
| | | | | X1 | | Low-true pulses (set Output on match) |
| | | 1 | XX | 10 | Center-Aligned PWM | High-true pulses (clear Output on match-up) |
| | | | | X1 | | Low-true pulses (set Output on match-up) |
| | 1 | 0 | XX | 10 | Combine PWM | High-true pulses (set on channel (n) match, and clear on channel (n+1) match) |
| | | | | X1 | | Low-true pulses (clear on channel (n) match, and set on channel (n +1) match) |

*Table continues on the next page...*

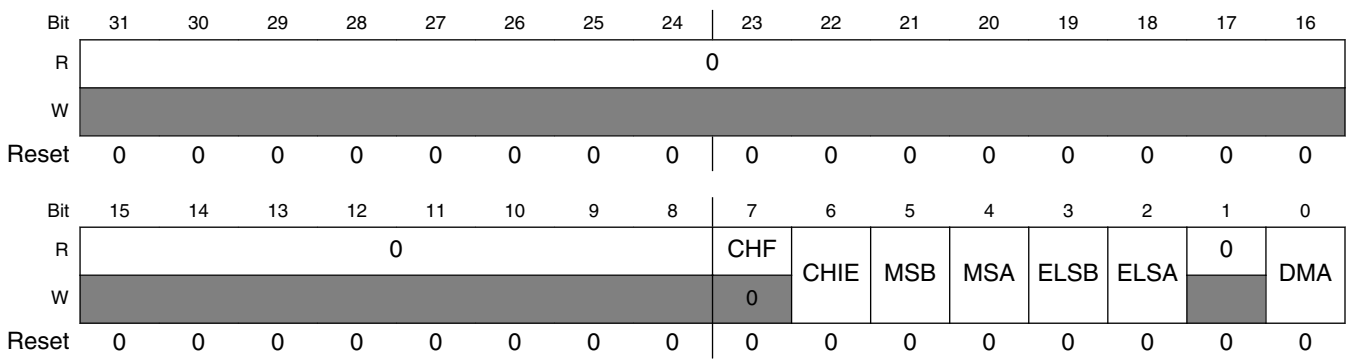
## Table 40-67. Mode, edge, and level selection (continued)

| DECAPEN | COMBINE | CPWMS | MSnB:MSnA | ELSnB:ELSnA | Mode | Configuration |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | X0 | See the following table (Table 40-8). | Dual Edge Capture | One-Shot Capture mode |
| | | | X1 | | | Continuous Capture mode |

## Table 40-68. Dual Edge Capture mode — edge polarity selection

| ELSnB | ELSnA | Channel Port Enable | Detected Edges |
|---|---|---|---|
| 0 | 0 | Disabled | No edge |
| 0 | 1 | Enabled | Rising edge |
| 1 | 0 | Enabled | Falling edge |
| 1 | 1 | Enabled | Rising and falling edges |

Address: Base address + Ch offset + (8d × i), where i=0d to 7d




## FTMx_CnSC field descriptions

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>CHF | Channel Flag<br>Set by hardware when an event occurs on the channel. CHF is cleared by reading the CSC register while CHnF is set and then writing a 0 to the CHF bit. Writing a 1 to CHF has no effect.<br>If another event occurs between the read and write operations, the write operation has no effect; therefore, CHF remains set indicating an event has occurred. In this case a CHF interrupt request is not lost due to the clearing sequence for a previous CHF.<br>0 No channel event has occurred.<br>1 A channel event has occurred. |
| 6<br>CHIE | Channel Interrupt Enable<br>Enables channel interrupts. |

*Table continues on the next page...*

**FTMx_C*n*SC field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Disable channel interrupts. Use software polling.<br>1 Enable channel interrupts. |
| 5<br>MSB | Channel Mode Select<br>Used for further selections in the channel logic. Its functionality is dependent on the channel mode. See Table 40-7.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1. |
| 4<br>MSA | Channel Mode Select<br>Used for further selections in the channel logic. Its functionality is dependent on the channel mode. See Table 40-7.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1. |
| 3<br>ELSB | Edge or Level Select<br>The functionality of ELSB and ELSA depends on the channel mode. See Table 40-7.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1. |
| 2<br>ELSA | Edge or Level Select<br>The functionality of ELSB and ELSA depends on the channel mode. See Table 40-7.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1. |
| 1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>DMA | DMA Enable<br>Enables DMA transfers for the channel.<br>0 Disable DMA transfers.<br>1 Enable DMA transfers. |

## 40.3.7 Channel (n) Value (FTMx_C*n*V)

These registers contain the captured FTM counter value for the input modes or the match value for the output modes.

In Input Capture, Capture Test, and Dual Edge Capture modes, any write to a CnV register is ignored.

In output modes, writing to a CnV register latches the value into a buffer. A CnV register is updated with the value of its write buffer according to Registers updated from write buffers.

If FTMEN = 0, this write coherency mechanism may be manually reset by writing to the CnSC register whether BDM mode is active or not.

Address: Base address + 10h offset + (8d × i), where i=0d to 7d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | VAL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTMx_C*n*V field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–0<br>VAL | Channel Value<br><br>Captured FTM counter value of the input modes or the match value for the output modes |

## 40.3.8 Counter Initial Value (FTMx_CNTIN)

The Counter Initial Value register contains the initial value for the FTM counter.

Writing to the CNTIN register latches the value into a buffer. The CNTIN register is updated with the value of its write buffer according to Registers updated from write buffers.

When the FTM clock is initially selected, by writing a non-zero value to the CLKS bits, the FTM counter starts with the value 0x0000. To avoid this behavior, before the first write to select the FTM clock, write the new value to the the CNTIN register and then initialize the FTM counter by writing any value to the CNT register.

Address: Base address + 4Ch offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | | | | | | | | | INIT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTMx_CNTIN field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved. |
| 15–0<br>INIT | Initial Value Of The FTM Counter |

## 40.3.9 Capture And Compare Status (FTMx_STATUS)

The STATUS register contains a copy of the status flag CHnF bit in CnSC for each FTM channel for software convenience.

Each CHnF bit in STATUS is a mirror of CHnF bit in CnSC. All CHnF bits can be checked using only one read of STATUS. All CHnF bits can be cleared by reading STATUS followed by writing 0x00 to STATUS.

Hardware sets the individual channel flags when an event occurs on the channel. CHnF is cleared by reading STATUS while CHnF is set and then writing a 0 to the CHnF bit. Writing a 1 to CHnF has no effect.

If another event occurs between the read and write operations, the write operation has no effect; therefore, CHnF remains set indicating an event has occurred. In this case, a CHnF interrupt request is not lost due to the clearing sequence for a previous CHnF.

**NOTE**

The STATUS register should be used only in Combine mode.

Address: Base address + 50h offset

| Bit | 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CH7F | CH6F | CH5F | CH4F | CH3F | CH2F | CH1F | CH0F |
| W | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTMx_STATUS field descriptions**

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>CH7F | Channel 7 Flag<br><br>See the register description. |

*Table continues on the next page...*

**FTMx_STATUS field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 No channel event has occurred.<br>1 A channel event has occurred. |
| 6<br>CH6F | Channel 6 Flag<br><br>See the register description.<br><br>0 No channel event has occurred.<br>1 A channel event has occurred. |
| 5<br>CH5F | Channel 5 Flag<br><br>See the register description.<br><br>0 No channel event has occurred.<br>1 A channel event has occurred. |
| 4<br>CH4F | Channel 4 Flag<br><br>See the register description.<br><br>0 No channel event has occurred.<br>1 A channel event has occurred. |
| 3<br>CH3F | Channel 3 Flag<br><br>See the register description.<br><br>0 No channel event has occurred.<br>1 A channel event has occurred. |
| 2<br>CH2F | Channel 2 Flag<br><br>See the register description.<br><br>0 No channel event has occurred.<br>1 A channel event has occurred. |
| 1<br>CH1F | Channel 1 Flag<br><br>See the register description.<br><br>0 No channel event has occurred.<br>1 A channel event has occurred. |
| 0<br>CH0F | Channel 0 Flag<br><br>See the register description.<br><br>0 No channel event has occurred.<br>1 A channel event has occurred. |

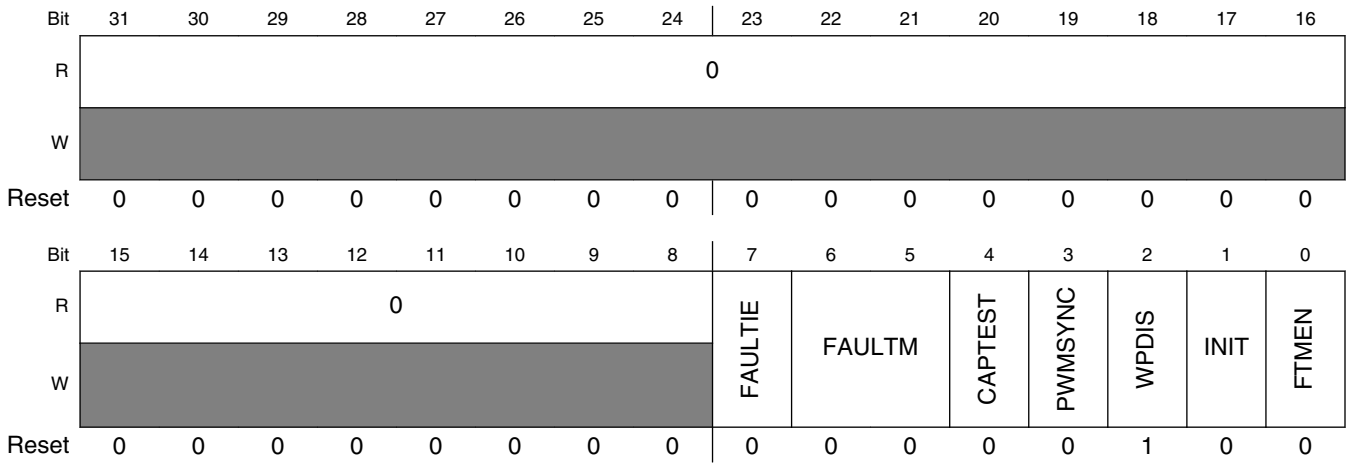
## 40.3.10 Features Mode Selection (FTMx_MODE)

This register contains the global enable bit for FTM-specific features and the control bits used to configure:

- Fault control mode and interrupt
- Capture Test mode
- PWM synchronization
- Write protection
- Channel output initialization

These controls relate to all channels within this module.

Address: Base address + 54h offset




**FTMx_MODE field descriptions**

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>FAULTIE | Fault Interrupt Enable<br><br>Enables the generation of an interrupt when a fault is detected by FTM and the FTM fault control is enabled.<br><br>0 Fault control interrupt is disabled.<br>1 Fault control interrupt is enabled. |
| 6–5<br>FAULTM | Fault Control Mode<br><br>Defines the FTM fault control mode.<br><br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br><br>00 Fault control is disabled for all channels.<br>01 Fault control is enabled for even channels only (channels 0, 2, 4, and 6), and the selected mode is the manual fault clearing.<br>10 Fault control is enabled for all channels, and the selected mode is the manual fault clearing.<br>11 Fault control is enabled for all channels, and the selected mode is the automatic fault clearing. |
| 4<br>CAPTEST | Capture Test Mode Enable<br><br>Enables the capture test mode.<br><br>This field is write protected. It can be written only when MODE[WPDIS] = 1. |

*Table continues on the next page...*

**FTMx_MODE field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Capture test mode is disabled.<br>1 Capture test mode is enabled. |
| 3<br>PWMSYNC | PWM Synchronization Mode<br><br>Selects which triggers can be used by MOD, CnV, OUTMASK, and FTM counter synchronization. See PWM synchronization. The PWMSYNC bit configures the synchronization when SYNCMODE is zero.<br><br>0 No restrictions. Software and hardware triggers can be used by MOD, CnV, OUTMASK, and FTM counter synchronization.<br>1 Software trigger can only be used by MOD and CnV synchronization, and hardware triggers can only be used by OUTMASK and FTM counter synchronization. |
| 2<br>WPDIS | Write Protection Disable<br><br>When write protection is enabled (WPDIS = 0), write protected bits cannot be written. When write protection is disabled (WPDIS = 1), write protected bits can be written. The WPDIS bit is the negation of the WPEN bit. WPDIS is cleared when 1 is written to WPEN. WPDIS is set when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPDIS has no effect.<br><br>0 Write protection is enabled.<br>1 Write protection is disabled. |
| 1<br>INIT | Initialize The Channels Output<br><br>When a 1 is written to INIT bit the channels output is initialized according to the state of their corresponding bit in the OUTINIT register. Writing a 0 to INIT bit has no effect.<br><br>The INIT bit is always read as 0. |
| 0<br>FTMEN | FTM Enable<br><br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br><br>0 Only the TPM-compatible registers (first set of registers) can be used without any restriction. Do not use the FTM-specific registers.<br>1 All registers including the FTM-specific registers (second set of registers) are available for use with no restrictions. |

## 40.3.11 Synchronization (FTMx_SYNC)

This register configures the PWM synchronization.

A synchronization event can perform the synchronized update of MOD, CV, and OUTMASK registers with the value of their write buffer and the FTM counter initialization.

## NOTE

The software trigger, SWSYNC bit, and hardware triggers TRIG0, TRIG1, and TRIG2 bits have a potential conflict if used together when SYNCMODE = 0. Use only hardware or software triggers but not both at the same time, otherwise unpredictable behavior is likely to happen.

The selection of the loading point, CNTMAX and CNTMIN bits, is intended to provide the update of MOD, CNTIN, and CnV registers across all enabled channels simultaneously. The use of the loading point selection together with SYNCMODE = 0 and hardware trigger selection, TRIG0, TRIG1, or TRIG2 bits, is likely to result in unpredictable behavior.

The synchronization event selection also depends on the PWMSYNC (MODE register) and SYNCMODE (SYNCONF register) bits. See PWM synchronization.

Address: Base address + 58h offset

| Bit | 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | SWSYNC | TRIG2 | TRIG1 | TRIG0 | SYNCHOM | REINIT | CNTMAX | CNTMIN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### FTMx_SYNC field descriptions

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>SWSYNC | PWM Synchronization Software Trigger<br><br>Selects the software trigger as the PWM synchronization trigger. The software trigger happens when a 1 is written to SWSYNC bit.<br><br>0 Software trigger is not selected.<br>1 Software trigger is selected. |
| 6<br>TRIG2 | PWM Synchronization Hardware Trigger 2 |

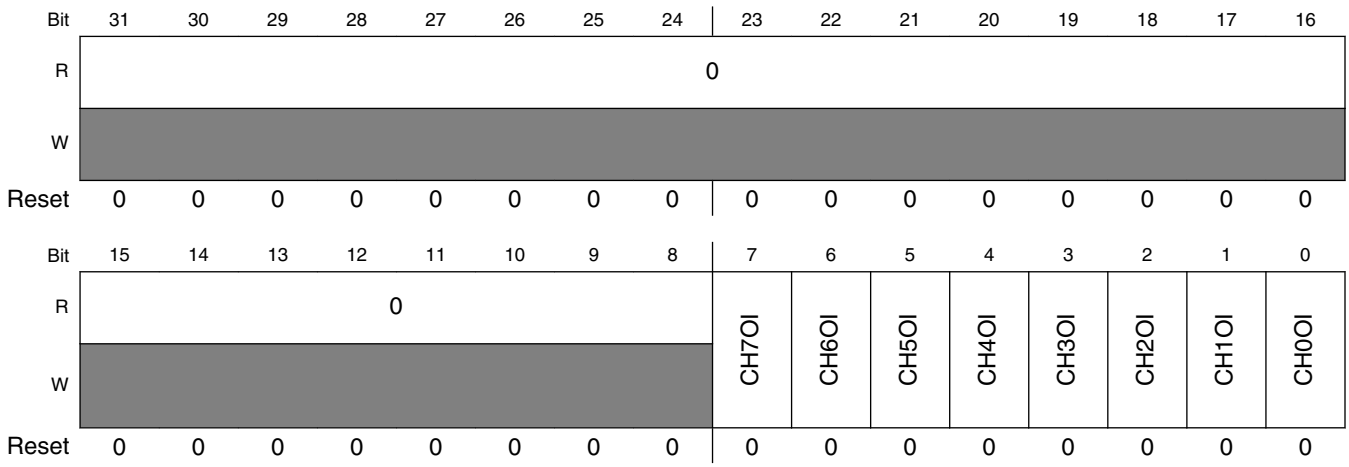
*Table continues on the next page...*

## FTMx_SYNC field descriptions (continued)

| Field | Description |
|---|---|
| | Enables hardware trigger 2 to the PWM synchronization. Hardware trigger 2 happens when a rising edge is detected at the trigger 2 input signal.<br><br>0 Trigger is disabled.<br>1 Trigger is enabled. |
| 5<br>TRIG1 | PWM Synchronization Hardware Trigger 1<br><br>Enables hardware trigger 1 to the PWM synchronization. Hardware trigger 1 happens when a rising edge is detected at the trigger 1 input signal.<br><br>0 Trigger is disabled.<br>1 Trigger is enabled. |
| 4<br>TRIG0 | PWM Synchronization Hardware Trigger 0<br><br>Enables hardware trigger 0 to the PWM synchronization. Hardware trigger 0 happens when a rising edge is detected at the trigger 0 input signal.<br><br>0 Trigger is disabled.<br>1 Trigger is enabled. |
| 3<br>SYNCHOM | Output Mask Synchronization<br><br>Selects when the OUTMASK register is updated with the value of its buffer.<br><br>0 OUTMASK register is updated with the value of its buffer in all rising edges of the system clock.<br>1 OUTMASK register is updated with the value of its buffer only by the PWM synchronization. |
| 2<br>REINIT | FTM Counter Reinitialization By Synchronization (FTM counter synchronization)<br><br>Determines if the FTM counter is reinitialized when the selected trigger for the synchronization is detected. The REINIT bit configures the synchronization when SYNCMODE is zero.<br><br>0 FTM counter continues to count normally.<br>1 FTM counter is updated with its initial value when the selected trigger is detected. |
| 1<br>CNTMAX | Maximum Loading Point Enable<br><br>Selects the maximum loading point to PWM synchronization. See Boundary cycle and loading points. If CNTMAX is one, the selected loading point is when the FTM counter reaches its maximum value (MOD register).<br><br>0 The maximum loading point is disabled.<br>1 The maximum loading point is enabled. |
| 0<br>CNTMIN | Minimum Loading Point Enable<br><br>Selects the minimum loading point to PWM synchronization. See Boundary cycle and loading points. If CNTMIN is one, the selected loading point is when the FTM counter reaches its minimum value (CNTIN register).<br><br>0 The minimum loading point is disabled.<br>1 The minimum loading point is enabled. |

## 40.3.12 Initial State For Channels Output (FTMx_OUTINIT)

Address: Base address + 5Ch offset




### FTMx_OUTINIT field descriptions

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>CH7OI | Channel 7 Output Initialization Value<br><br>Selects the value that is forced into the channel output when the initialization occurs.<br><br>0 The initialization value is 0.<br>1 The initialization value is 1. |
| 6<br>CH6OI | Channel 6 Output Initialization Value<br><br>Selects the value that is forced into the channel output when the initialization occurs.<br><br>0 The initialization value is 0.<br>1 The initialization value is 1. |
| 5<br>CH5OI | Channel 5 Output Initialization Value<br><br>Selects the value that is forced into the channel output when the initialization occurs.<br><br>0 The initialization value is 0.<br>1 The initialization value is 1. |
| 4<br>CH4OI | Channel 4 Output Initialization Value<br><br>Selects the value that is forced into the channel output when the initialization occurs.<br><br>0 The initialization value is 0.<br>1 The initialization value is 1. |
| 3<br>CH3OI | Channel 3 Output Initialization Value<br><br>Selects the value that is forced into the channel output when the initialization occurs. |

*Table continues on the next page...*

**FTMx_OUTINIT field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 The initialization value is 0.<br>1 The initialization value is 1. |
| 2<br>CH2OI | Channel 2 Output Initialization Value<br><br>Selects the value that is forced into the channel output when the initialization occurs.<br><br>0 The initialization value is 0.<br>1 The initialization value is 1. |
| 1<br>CH1OI | Channel 1 Output Initialization Value<br><br>Selects the value that is forced into the channel output when the initialization occurs.<br><br>0 The initialization value is 0.<br>1 The initialization value is 1. |
| 0<br>CH0OI | Channel 0 Output Initialization Value<br><br>Selects the value that is forced into the channel output when the initialization occurs.<br><br>0 The initialization value is 0.<br>1 The initialization value is 1. |

## 40.3.13 Output Mask (FTMx_OUTMASK)

This register provides a mask for each FTM channel. The mask of a channel determines if its output responds, that is, it is masked or not, when a match occurs. This feature is used for BLDC control where the PWM signal is presented to an electric motor at specific times to provide electronic commutation.

Any write to the OUTMASK register, stores the value in its write buffer. The register is updated with the value of its write buffer according to PWM synchronization.

Address: Base address + 60h offset

| Bit | 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CH7OM | CH6OM | CH5OM | CH4OM | CH3OM | CH2OM | CH1OM | CH0OM |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## FTMx_OUTMASK field descriptions

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>CH7OM | Channel 7 Output Mask<br><br>Defines if the channel output is masked or unmasked.<br><br>0 Channel output is not masked. It continues to operate normally.<br>1 Channel output is masked. It is forced to its inactive state. |
| 6<br>CH6OM | Channel 6 Output Mask<br><br>Defines if the channel output is masked or unmasked.<br><br>0 Channel output is not masked. It continues to operate normally.<br>1 Channel output is masked. It is forced to its inactive state. |
| 5<br>CH5OM | Channel 5 Output Mask<br><br>Defines if the channel output is masked or unmasked.<br><br>0 Channel output is not masked. It continues to operate normally.<br>1 Channel output is masked. It is forced to its inactive state. |
| 4<br>CH4OM | Channel 4 Output Mask<br><br>Defines if the channel output is masked or unmasked.<br><br>0 Channel output is not masked. It continues to operate normally.<br>1 Channel output is masked. It is forced to its inactive state. |
| 3<br>CH3OM | Channel 3 Output Mask<br><br>Defines if the channel output is masked or unmasked.<br><br>0 Channel output is not masked. It continues to operate normally.<br>1 Channel output is masked. It is forced to its inactive state. |
| 2<br>CH2OM | Channel 2 Output Mask<br><br>Defines if the channel output is masked or unmasked.<br><br>0 Channel output is not masked. It continues to operate normally.<br>1 Channel output is masked. It is forced to its inactive state. |
| 1<br>CH1OM | Channel 1 Output Mask<br><br>Defines if the channel output is masked or unmasked.<br><br>0 Channel output is not masked. It continues to operate normally.<br>1 Channel output is masked. It is forced to its inactive state. |
| 0<br>CH0OM | Channel 0 Output Mask<br><br>Defines if the channel output is masked or unmasked.<br><br>0 Channel output is not masked. It continues to operate normally.<br>1 Channel output is masked. It is forced to its inactive state. |

## 40.3.14 Function For Linked Channels (FTMx_COMBINE)

This register contains the control bits used to configure the fault control, synchronization, deadtime insertion, Dual Edge Capture mode, Complementary, and Combine mode for each pair of channels (n) and (n+1), where n equals 0, 2, 4, and 6.

Address: Base address + 64h offset

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | FAULTEN3 | SYNCEN3 | DTEN3 | DECAP3 | DECAPEN3 | COMP3 | COMBINE3 | 0 | FAULTEN2 | SYNCEN2 | DTEN2 | DECAP2 | DECAPEN2 | COMP2 | COMBINE2 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | FAULTEN1 | SYNCEN1 | DTEN1 | DECAP1 | DECAPEN1 | COMP1 | COMBINE1 | 0 | FAULTEN0 | SYNCEN0 | DTEN0 | DECAP0 | DECAPEN0 | COMP0 | COMBINE0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### FTMx_COMBINE field descriptions

| Field | Description |
|---|---|
| 31<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 30<br>FAULTEN3 | Fault Control Enable For n = 6<br>Enables the fault control in channels (n) and (n+1).<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The fault control in this pair of channels is disabled.<br>1 The fault control in this pair of channels is enabled. |
| 29<br>SYNCEN3 | Synchronization Enable For n = 6<br>Enables PWM synchronization of registers C(n)V and C(n+1)V.<br>0 The PWM synchronization in this pair of channels is disabled.<br>1 The PWM synchronization in this pair of channels is enabled. |
| 28<br>DTEN3 | Deadtime Enable For n = 6<br>Enables the deadtime insertion in the channels (n) and (n+1).<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The deadtime insertion in this pair of channels is disabled.<br>1 The deadtime insertion in this pair of channels is enabled. |

*Table continues on the next page...*

## FTMx_COMBINE field descriptions (continued)

| Field | Description |
|---|---|
| 27<br>DECAP3 | Dual Edge Capture Mode Captures For n = 6<br><br>Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits.<br><br>This field applies only when FTMEN = 1 and DECAPEN = 1.<br><br>DECAP bit is cleared automatically by hardware if dual edge capture – one-shot mode is selected and when the capture of channel (n+1) event is made.<br><br>0 The dual edge captures are inactive.<br>1 The dual edge captures are active. |
| 26<br>DECAPEN3 | Dual Edge Capture Mode Enable For n = 6<br><br>Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 40-7.<br><br>This field applies only when FTMEN = 1.<br><br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br><br>0 The Dual Edge Capture mode in this pair of channels is disabled.<br>1 The Dual Edge Capture mode in this pair of channels is enabled. |
| 25<br>COMP3 | Complement Of Channel (n) for n = 6<br><br>Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output.<br><br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br><br>0 The channel (n+1) output is the same as the channel (n) output.<br>1 The channel (n+1) output is the complement of the channel (n) output. |
| 24<br>COMBINE3 | Combine Channels For n = 6<br><br>Enables the combine feature for channels (n) and (n+1).<br><br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br><br>0 Channels (n) and (n+1) are independent.<br>1 Channels (n) and (n+1) are combined. |
| 23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22<br>FAULTEN2 | Fault Control Enable For n = 4<br><br>Enables the fault control in channels (n) and (n+1).<br><br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br><br>0 The fault control in this pair of channels is disabled.<br>1 The fault control in this pair of channels is enabled. |
| 21<br>SYNCEN2 | Synchronization Enable For n = 4<br><br>Enables PWM synchronization of registers C(n)V and C(n+1)V.<br><br>0 The PWM synchronization in this pair of channels is disabled.<br>1 The PWM synchronization in this pair of channels is enabled. |

*Table continues on the next page...*

## FTMx_COMBINE field descriptions (continued)

| Field | Description |
|---|---|
| 20<br>DTEN2 | Deadtime Enable For n = 4<br><br>Enables the deadtime insertion in the channels (n) and (n+1).<br><br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br><br>0 The deadtime insertion in this pair of channels is disabled.<br>1 The deadtime insertion in this pair of channels is enabled. |
| 19<br>DECAP2 | Dual Edge Capture Mode Captures For n = 4<br><br>Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits.<br><br>This field applies only when FTMEN = 1 and DECAPEN = 1.<br><br>DECAP bit is cleared automatically by hardware if dual edge capture – one-shot mode is selected and when the capture of channel (n+1) event is made.<br><br>0 The dual edge captures are inactive.<br>1 The dual edge captures are active. |
| 18<br>DECAPEN2 | Dual Edge Capture Mode Enable For n = 4<br><br>Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 40-7.<br><br>This field applies only when FTMEN = 1.<br><br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br><br>0 The Dual Edge Capture mode in this pair of channels is disabled.<br>1 The Dual Edge Capture mode in this pair of channels is enabled. |
| 17<br>COMP2 | Complement Of Channel (n) For n = 4<br><br>Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output.<br><br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br><br>0 The channel (n+1) output is the same as the channel (n) output.<br>1 The channel (n+1) output is the complement of the channel (n) output. |
| 16<br>COMBINE2 | Combine Channels For n = 4<br><br>Enables the combine feature for channels (n) and (n+1).<br><br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br><br>0 Channels (n) and (n+1) are independent.<br>1 Channels (n) and (n+1) are combined. |
| 15<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 14<br>FAULTEN1 | Fault Control Enable For n = 2<br><br>Enables the fault control in channels (n) and (n+1).<br><br>This field is write protected. It can be written only when MODE[WPDIS] = 1. |

*Table continues on the next page...*

## FTMx_COMBINE field descriptions (continued)

| Field | Description |
|---|---|
| | 0 The fault control in this pair of channels is disabled.<br>1 The fault control in this pair of channels is enabled. |
| 13<br>SYNCEN1 | Synchronization Enable For n = 2<br>Enables PWM synchronization of registers C(n)V and C(n+1)V.<br>0 The PWM synchronization in this pair of channels is disabled.<br>1 The PWM synchronization in this pair of channels is enabled. |
| 12<br>DTEN1 | Deadtime Enable For n = 2<br>Enables the deadtime insertion in the channels (n) and (n+1).<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The deadtime insertion in this pair of channels is disabled.<br>1 The deadtime insertion in this pair of channels is enabled. |
| 11<br>DECAP1 | Dual Edge Capture Mode Captures For n = 2<br>Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits.<br>This field applies only when FTMEN = 1 and DECAPEN = 1.<br>DECAP bit is cleared automatically by hardware if Dual Edge Capture – One-Shot mode is selected and when the capture of channel (n+1) event is made.<br>0 The dual edge captures are inactive.<br>1 The dual edge captures are active. |
| 10<br>DECAPEN1 | Dual Edge Capture Mode Enable For n = 2<br>Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 40-7.<br>This field applies only when FTMEN = 1.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The Dual Edge Capture mode in this pair of channels is disabled.<br>1 The Dual Edge Capture mode in this pair of channels is enabled. |
| 9<br>COMP1 | Complement Of Channel (n) For n = 2<br>Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The channel (n+1) output is the same as the channel (n) output.<br>1 The channel (n+1) output is the complement of the channel (n) output. |
| 8<br>COMBINE1 | Combine Channels For n = 2<br>Enables the combine feature for channels (n) and (n+1).<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 Channels (n) and (n+1) are independent.<br>1 Channels (n) and (n+1) are combined. |

*Table continues on the next page...*

## FTMx_COMBINE field descriptions (continued)

| Field | Description |
|---|---|
| 7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6<br>FAULTEN0 | Fault Control Enable For n = 0<br>Enables the fault control in channels (n) and (n+1).<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The fault control in this pair of channels is disabled.<br>1 The fault control in this pair of channels is enabled. |
| 5<br>SYNCEN0 | Synchronization Enable For n = 0<br>Enables PWM synchronization of registers C(n)V and C(n+1)V.<br>0 The PWM synchronization in this pair of channels is disabled.<br>1 The PWM synchronization in this pair of channels is enabled. |
| 4<br>DTEN0 | Deadtime Enable For n = 0<br>Enables the deadtime insertion in the channels (n) and (n+1).<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The deadtime insertion in this pair of channels is disabled.<br>1 The deadtime insertion in this pair of channels is enabled. |
| 3<br>DECAP0 | Dual Edge Capture Mode Captures For n = 0<br>Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits.<br>This field applies only when FTMEN = 1 and DECAPEN = 1.<br>DECAP bit is cleared automatically by hardware if dual edge capture – one-shot mode is selected and when the capture of channel (n+1) event is made.<br>0 The dual edge captures are inactive.<br>1 The dual edge captures are active. |
| 2<br>DECAPEN0 | Dual Edge Capture Mode Enable For n = 0<br>Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 40-7.<br>This field applies only when FTMEN = 1.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The Dual Edge Capture mode in this pair of channels is disabled.<br>1 The Dual Edge Capture mode in this pair of channels is enabled. |
| 1<br>COMP0 | Complement Of Channel (n) For n = 0<br>Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The channel (n+1) output is the same as the channel (n) output.<br>1 The channel (n+1) output is the complement of the channel (n) output. |

*Table continues on the next page...*

**FTMx_COMBINE field descriptions (continued)**

| Field | Description |
|---|---|
| 0<br>COMBINE0 | Combine Channels For n = 0<br>Enables the combine feature for channels (n) and (n+1).<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 Channels (n) and (n+1) are independent.<br>1 Channels (n) and (n+1) are combined. |

## 40.3.15 Deadtime Insertion Control (FTMx_DEADTIME)

This register selects the deadtime prescaler factor and deadtime value. All FTM channels use this clock prescaler and this deadtime value for the deadtime insertion.

Address: Base address + 68h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | | | | | | | | | DTPS | | DTVAL | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTMx_DEADTIME field descriptions**

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7–6<br>DTPS | Deadtime Prescaler Value<br>Selects the division factor of the system clock. This prescaled clock is used by the deadtime counter.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0x Divide the system clock by 1.<br>10 Divide the system clock by 4.<br>11 Divide the system clock by 16. |
| 5–0<br>DTVAL | Deadtime Value<br>Selects the deadtime insertion value for the deadtime counter. The deadtime counter is clocked by a scaled version of the system clock. See the description of DTPS.<br>Deadtime insert value = (DTPS × DTVAL).<br>DTVAL selects the number of deadtime counts inserted as follows:<br>When DTVAL is 0, no counts are inserted.<br>When DTVAL is 1, 1 count is inserted.<br>When DTVAL is 2, 2 counts are inserted.<br>This pattern continues up to a possible 63 counts.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1. |

## 40.3.16 FTM External Trigger (FTMx_EXTTRIG)

This register:

- Indicates when a channel trigger was generated
- Enables the generation of a trigger when the FTM counter is equal to its initial value
- Selects which channels are used in the generation of the channel triggers

Several channels can be selected to generate multiple triggers in one PWM period.

Channels 6 and 7 are not used to generate channel triggers.

Address: Base address + 6Ch offset

| Bit | 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | TRIGF | INITTRIGEN | CH1TRIG | CH0TRIG | CH5TRIG | CH4TRIG | CH3TRIG | CH2TRIG |
| W | | | | | | | | | 0 | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## FTMx_EXTTRIG field descriptions

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved. |
| 7<br>TRIGF | Channel Trigger Flag<br><br>Set by hardware when a channel trigger is generated. Clear TRIGF by reading EXTTRIG while TRIGF is set and then writing a 0 to TRIGF. Writing a 1 to TRIGF has no effect.<br><br>If another channel trigger is generated before the clearing sequence is completed, the sequence is reset so TRIGF remains set after the clear sequence is completed for the earlier TRIGF.<br><br>0 No channel trigger was generated.<br>1 A channel trigger was generated. |
| 6<br>INITTRIGEN | Initialization Trigger Enable<br><br>Enables the generation of the trigger when the FTM counter is equal to the CNTIN register.<br><br>0 The generation of initialization trigger is disabled.<br>1 The generation of initialization trigger is enabled. |
| 5<br>CH1TRIG | Channel 1 Trigger Enable<br><br>Enable the generation of the channel trigger when the FTM counter is equal to the CnV register.<br><br>0 The generation of the channel trigger is disabled.<br>1 The generation of the channel trigger is enabled. |
| 4<br>CH0TRIG | Channel 0 Trigger Enable<br><br>Enable the generation of the channel trigger when the FTM counter is equal to the CnV register.<br><br>0 The generation of the channel trigger is disabled.<br>1 The generation of the channel trigger is enabled. |
| 3<br>CH5TRIG | Channel 5 Trigger Enable<br><br>Enable the generation of the channel trigger when the FTM counter is equal to the CnV register.<br><br>0 The generation of the channel trigger is disabled.<br>1 The generation of the channel trigger is enabled. |
| 2<br>CH4TRIG | Channel 4 Trigger Enable<br><br>Enable the generation of the channel trigger when the FTM counter is equal to the CnV register.<br><br>0 The generation of the channel trigger is disabled.<br>1 The generation of the channel trigger is enabled. |
| 1<br>CH3TRIG | Channel 3 Trigger Enable<br><br>Enable the generation of the channel trigger when the FTM counter is equal to the CnV register.<br><br>0 The generation of the channel trigger is disabled.<br>1 The generation of the channel trigger is enabled. |
| 0<br>CH2TRIG | Channel 2 Trigger Enable<br><br>Enable the generation of the channel trigger when the FTM counter is equal to the CnV register. |

*Table continues on the next page...*

**FTMx_EXTTRIG field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 The generation of the channel trigger is disabled.<br>1 The generation of the channel trigger is enabled. |

## 40.3.17 Channels Polarity (FTMx_POL)

This register defines the output polarity of the FTM channels.

**NOTE**

The safe value that is driven in a channel output when the fault control is enabled and a fault condition is detected is the inactive state of the channel. That is, the safe value of a channel is the value of its POL bit.

Address: Base address + 70h offset

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

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | POL7 | POL6 | POL5 | POL4 | POL3 | POL2 | POL1 | POL0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTMx_POL field descriptions**

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved. |
| 7<br>POL7 | Channel 7 Polarity<br>Defines the polarity of the channel output.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The channel polarity is active high.<br>1 The channel polarity is active low. |
| 6<br>POL6 | Channel 6 Polarity<br>Defines the polarity of the channel output.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The channel polarity is active high.<br>1 The channel polarity is active low. |
| 5<br>POL5 | Channel 5 Polarity<br>Defines the polarity of the channel output.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1. |

*Table continues on the next page...*

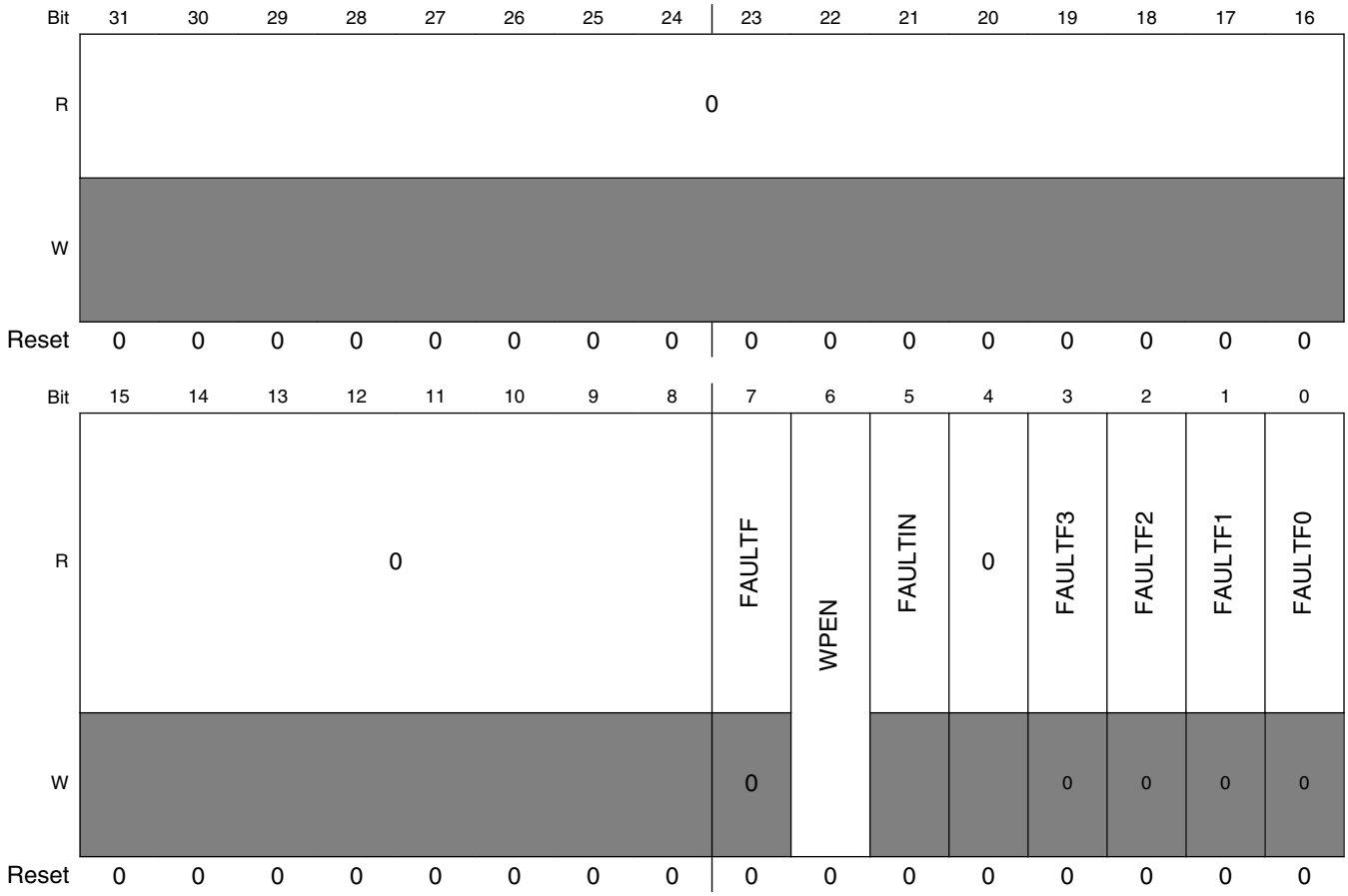
## FTMx_POL field descriptions (continued)

| Field | Description |
|---|---|
| | 0 The channel polarity is active high.<br>1 The channel polarity is active low. |
| 4<br>POL4 | Channel 4 Polarity<br>Defines the polarity of the channel output.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The channel polarity is active high.<br>1 The channel polarity is active low. |
| 3<br>POL3 | Channel 3 Polarity<br>Defines the polarity of the channel output.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The channel polarity is active high.<br>1 The channel polarity is active low. |
| 2<br>POL2 | Channel 2 Polarity<br>Defines the polarity of the channel output.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The channel polarity is active high.<br>1 The channel polarity is active low. |
| 1<br>POL1 | Channel 1 Polarity<br>Defines the polarity of the channel output.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The channel polarity is active high.<br>1 The channel polarity is active low. |
| 0<br>POL0 | Channel 0 Polarity<br>Defines the polarity of the channel output.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The channel polarity is active high.<br>1 The channel polarity is active low. |

## 40.3.18 Fault Mode Status (FTMx_FMS)

This register contains the fault detection flags, write protection enable bit, and the logic OR of the enabled fault inputs.

Address: Base address + 74h offset




**FTMx_FMS field descriptions**

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>FAULTF | Fault Detection Flag<br><br>Represents the logic OR of the individual FAULTFj bits where j = 3, 2, 1, 0. Clear FAULTF by reading the FMS register while FAULTF is set and then writing a 0 to FAULTF while there is no existing fault condition at the enabled fault inputs. Writing a 1 to FAULTF has no effect.<br><br>If another fault condition is detected in an enabled fault input before the clearing sequence is completed, the sequence is reset so FAULTF remains set after the clearing sequence is completed for the earlier fault condition. FAULTF is also cleared when FAULTFj bits are cleared individually.<br><br>0 No fault condition was detected.<br>1 A fault condition was detected. |

*Table continues on the next page...*

## FTMx_FMS field descriptions (continued)

| Field | Description |
|---|---|
| 6<br>WPEN | Write Protection Enable<br><br>The WPEN bit is the negation of the WPDIS bit. WPEN is set when 1 is written to it. WPEN is cleared when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPEN has no effect.<br><br>0 Write protection is disabled. Write protected bits can be written.<br>1 Write protection is enabled. Write protected bits cannot be written. |
| 5<br>FAULTIN | Fault Inputs<br><br>Represents the logic OR of the enabled fault inputs after their filter (if their filter is enabled) when fault control is enabled.<br><br>0 The logic OR of the enabled fault inputs is 0.<br>1 The logic OR of the enabled fault inputs is 1. |
| 4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3<br>FAULTF3 | Fault Detection Flag 3<br><br>Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input.<br><br>Clear FAULTF3 by reading the FMS register while FAULTF3 is set and then writing a 0 to FAULTF3 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTF3 has no effect. FAULTF3 bit is also cleared when FAULTF bit is cleared.<br><br>If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTF3 remains set after the clearing sequence is completed for the earlier fault condition.<br><br>0 No fault condition was detected at the fault input.<br>1 A fault condition was detected at the fault input. |
| 2<br>FAULTF2 | Fault Detection Flag 2<br><br>Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input.<br><br>Clear FAULTF2 by reading the FMS register while FAULTF2 is set and then writing a 0 to FAULTF2 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTF2 has no effect. FAULTF2 bit is also cleared when FAULTF bit is cleared.<br><br>If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTF2 remains set after the clearing sequence is completed for the earlier fault condition.<br><br>0 No fault condition was detected at the fault input.<br>1 A fault condition was detected at the fault input. |
| 1<br>FAULTF1 | Fault Detection Flag 1<br><br>Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input.<br><br>Clear FAULTF1 by reading the FMS register while FAULTF1 is set and then writing a 0 to FAULTF1 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTF1 has no effect. FAULTF1 bit is also cleared when FAULTF bit is cleared. |

*Table continues on the next page...*

**FTMx_FMS field descriptions (continued)**

| Field | Description |
|---|---|
| | If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTF1 remains set after the clearing sequence is completed for the earlier fault condition.<br><br>0 No fault condition was detected at the fault input.<br>1 A fault condition was detected at the fault input. |
| 0<br>FAULTF0 | Fault Detection Flag 0<br><br>Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input.<br><br>Clear FAULTF0 by reading the FMS register while FAULTF0 is set and then writing a 0 to FAULTF0 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTF0 has no effect. FAULTF0 bit is also cleared when FAULTF bit is cleared.<br><br>If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTF0 remains set after the clearing sequence is completed for the earlier fault condition.<br><br>0 No fault condition was detected at the fault input.<br>1 A fault condition was detected at the fault input. |

## 40.3.19 Input Capture Filter Control (FTMx_FILTER)

This register selects the filter value for the inputs of channels.

Channels 4, 5, 6 and 7 do not have an input filter.

**NOTE**

Writing to the FILTER register has immediate effect and must be done only when the channels 0, 1, 2, and 3 are not in input modes. Failure to do this could result in a missing valid signal.

Address: Base address + 78h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | | CH3FVAL | | | | CH2FVAL | | | | CH1FVAL | | | | CH0FVAL | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTMx_FILTER field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved. |
| 15–12<br>CH3FVAL | Channel 3 Input Filter<br><br>Selects the filter value for the channel input.<br><br>The filter is disabled when the value is zero. |

*Table continues on the next page...*

**FTMx_FILTER field descriptions (continued)**

| Field | Description |
|---|---|
| 11–8<br>CH2FVAL | Channel 2 Input Filter<br>Selects the filter value for the channel input.<br>The filter is disabled when the value is zero. |
| 7–4<br>CH1FVAL | Channel 1 Input Filter<br>Selects the filter value for the channel input.<br>The filter is disabled when the value is zero. |
| 3–0<br>CH0FVAL | Channel 0 Input Filter<br>Selects the filter value for the channel input.<br>The filter is disabled when the value is zero. |

## 40.3.20 Fault Control (FTMx_FLTCTRL)

This register selects the filter value for the fault inputs, enables the fault inputs and the fault inputs filter.

Address: Base address + 7Ch offset

| Bit | 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | FFVAL | | | | FFLTR3EN | FFLTR2EN | FFLTR1EN | FFLTR0EN | FAULT3EN | FAULT2EN | FAULT1EN | FAULT0EN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTMx_FLTCTRL field descriptions**

| Field | Description |
|---|---|
| 31–12<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 11–8<br>FFVAL | Fault Input Filter<br>Selects the filter value for the fault inputs.<br>The fault filter is disabled when the value is zero. |

*Table continues on the next page...*

## FTMx_FLTCTRL field descriptions (continued)

| Field | Description |
|---|---|
| | **NOTE:** Writing to this field has immediate effect and must be done only when the fault control or all fault inputs are disabled. Failure to do this could result in a missing fault detection. |
| 7<br>FFLTR3EN | Fault Input 3 Filter Enable<br>Enables the filter for the fault input.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 Fault input filter is disabled.<br>1 Fault input filter is enabled. |
| 6<br>FFLTR2EN | Fault Input 2 Filter Enable<br>Enables the filter for the fault input.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 Fault input filter is disabled.<br>1 Fault input filter is enabled. |
| 5<br>FFLTR1EN | Fault Input 1 Filter Enable<br>Enables the filter for the fault input.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 Fault input filter is disabled.<br>1 Fault input filter is enabled. |
| 4<br>FFLTR0EN | Fault Input 0 Filter Enable<br>Enables the filter for the fault input.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 Fault input filter is disabled.<br>1 Fault input filter is enabled. |
| 3<br>FAULT3EN | Fault Input 3 Enable<br>Enables the fault input.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 Fault input is disabled.<br>1 Fault input is enabled. |
| 2<br>FAULT2EN | Fault Input 2 Enable<br>Enables the fault input.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 Fault input is disabled.<br>1 Fault input is enabled. |
| 1<br>FAULT1EN | Fault Input 1 Enable<br>Enables the fault input.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1. |

*Table continues on the next page...*

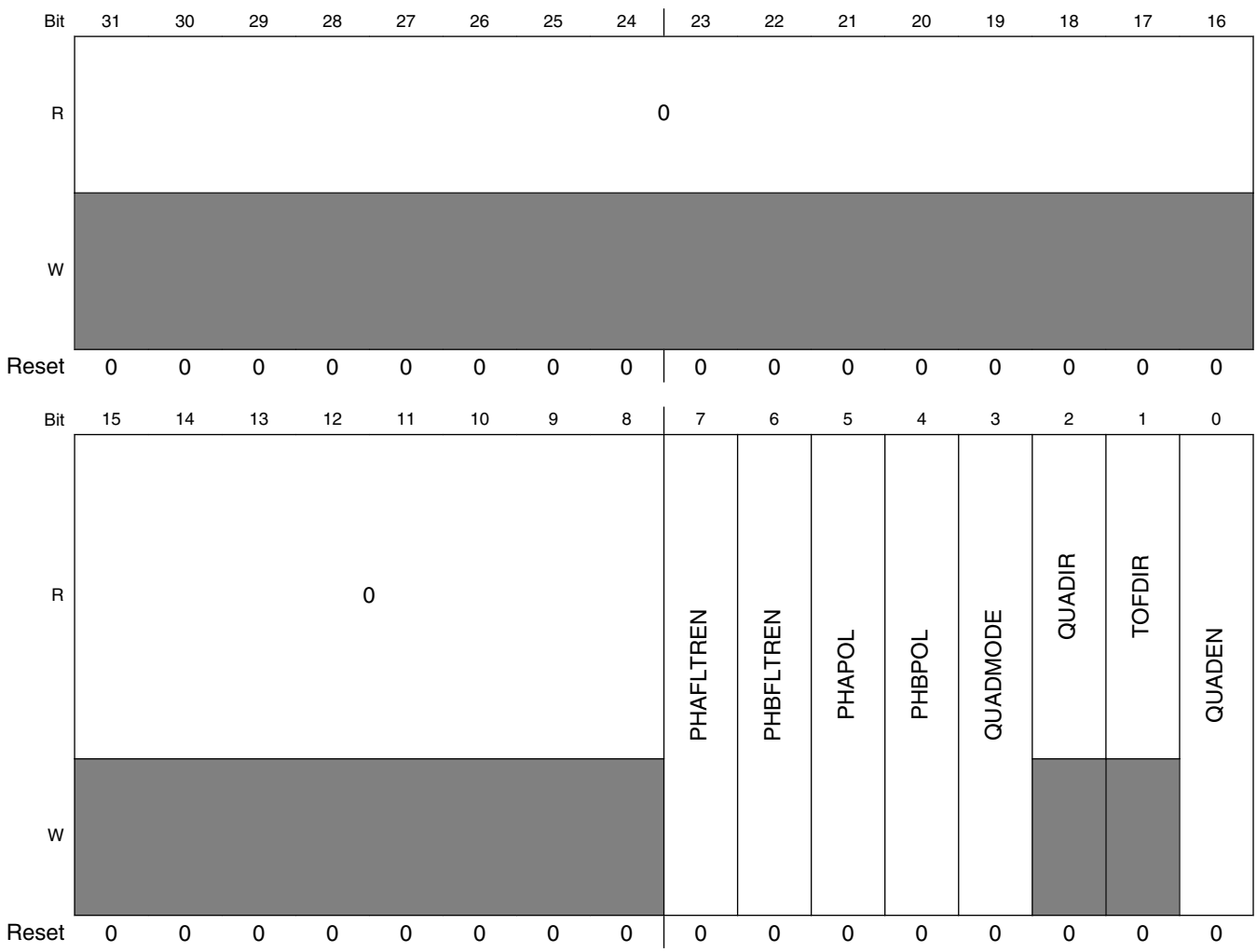
**FTMx_FLTCTRL field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Fault input is disabled.<br>1 Fault input is enabled. |
| 0<br>FAULT0EN | Fault Input 0 Enable<br>Enables the fault input.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 Fault input is disabled.<br>1 Fault input is enabled. |

## 40.3.21 Quadrature Decoder Control And Status (FTMx_QDCTRL)

This register has the control and status bits for the Quadrature Decoder mode.

Address: Base address + 80h offset



## FTMx_QDCTRL field descriptions

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>PHAFLTREN | Phase A Input Filter Enable<br><br>Enables the filter for the quadrature decoder phase A input. The filter value for the phase A input is defined by the CH0FVAL field of FILTER. The phase A filter is also disabled when CH0FVAL is zero.<br><br>0 Phase A input filter is disabled.<br>1 Phase A input filter is enabled. |
| 6<br>PHBFLTREN | Phase B Input Filter Enable<br><br>Enables the filter for the quadrature decoder phase B input. The filter value for the phase B input is defined by the CH1FVAL field of FILTER. The phase B filter is also disabled when CH1FVAL is zero.<br><br>0 Phase B input filter is disabled.<br>1 Phase B input filter is enabled. |
| 5<br>PHAPOL | Phase A Input Polarity<br><br>Selects the polarity for the quadrature decoder phase A input.<br><br>0 Normal polarity. Phase A input signal is not inverted before identifying the rising and falling edges of this signal.<br>1 Inverted polarity. Phase A input signal is inverted before identifying the rising and falling edges of this signal. |
| 4<br>PHBPOL | Phase B Input Polarity<br><br>Selects the polarity for the quadrature decoder phase B input.<br><br>0 Normal polarity. Phase B input signal is not inverted before identifying the rising and falling edges of this signal.<br>1 Inverted polarity. Phase B input signal is inverted before identifying the rising and falling edges of this signal. |
| 3<br>QUADMODE | Quadrature Decoder Mode<br><br>Selects the encoding mode used in the Quadrature Decoder mode.<br><br>0 Phase A and phase B encoding mode.<br>1 Count and direction encoding mode. |
| 2<br>QUADIR | FTM Counter Direction In Quadrature Decoder Mode<br><br>Indicates the counting direction.<br><br>0 Counting direction is decreasing (FTM counter decrement).<br>1 Counting direction is increasing (FTM counter increment). |
| 1<br>TOFDIR | Timer Overflow Direction In Quadrature Decoder Mode<br><br>Indicates if the TOF bit was set on the top or the bottom of counting.<br><br>0 TOF bit was set on the bottom of counting. There was an FTM counter decrement and FTM counter changes from its minimum value (CNTIN register) to its maximum value (MOD register).<br>1 TOF bit was set on the top of counting. There was an FTM counter increment and FTM counter changes from its maximum value (MOD register) to its minimum value (CNTIN register). |

*Table continues on the next page...*

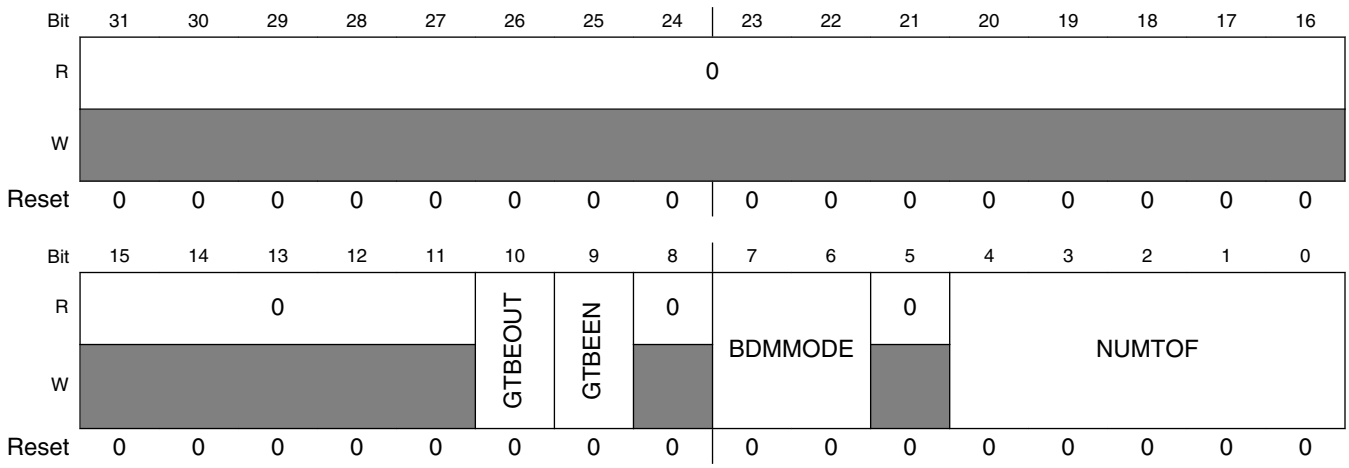
**FTMx_QDCTRL field descriptions (continued)**

| Field | Description |
|---|---|
| 0<br>QUADEN | Quadrature Decoder Mode Enable<br><br>Enables the Quadrature Decoder mode. In this mode, the phase A and B input signals control the FTM counter direction. The Quadrature Decoder mode has precedence over the other modes. See Table 40-7.<br><br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br><br>0 Quadrature Decoder mode is disabled.<br>1 Quadrature Decoder mode is enabled. |

## 40.3.22 Configuration (FTMx_CONF)

This register selects the number of times that the FTM counter overflow should occur before the TOF bit to be set, the FTM behavior in BDM modes, the use of an external global time base, and the global time base signal generation.

Address: Base address + 84h offset




**FTMx_CONF field descriptions**

| Field | Description |
|---|---|
| 31–11<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 10<br>GTBEOUT | Global Time Base Output<br><br>Enables the global time base signal generation to other FTMs.<br><br>0 A global time base signal generation is disabled.<br>1 A global time base signal generation is enabled. |
| 9<br>GTBEEN | Global Time Base Enable<br><br>Configures the FTM to use an external global time base signal that is generated by another FTM. |

*Table continues on the next page...*

**FTMx_CONF field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Use of an external global time base is disabled.<br>1 Use of an external global time base is enabled. |
| 8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7–6<br>BDMMODE | BDM Mode<br>Selects the FTM behavior in BDM mode. See BDM mode. |
| 5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4–0<br>NUMTOF | TOF Frequency<br>Selects the ratio between the number of counter overflows to the number of times the TOF bit is set.<br>NUMTOF = 0: The TOF bit is set for each counter overflow.<br>NUMTOF = 1: The TOF bit is set for the first counter overflow but not for the next overflow.<br>NUMTOF = 2: The TOF bit is set for the first counter overflow but not for the next 2 overflows.<br>NUMTOF = 3: The TOF bit is set for the first counter overflow but not for the next 3 overflows.<br>This pattern continues up to a maximum of 31. |

## 40.3.23 FTM Fault Input Polarity (FTMx_FLTPOL)

This register defines the fault inputs polarity.

Address: Base address + 88h offset

| Bit | 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | FLT3POL | FLT2POL | FLT1POL | FLT0POL |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTMx_FLTPOL field descriptions**

| Field | Description |
|---|---|
| 31–4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

*Table continues on the next page...*

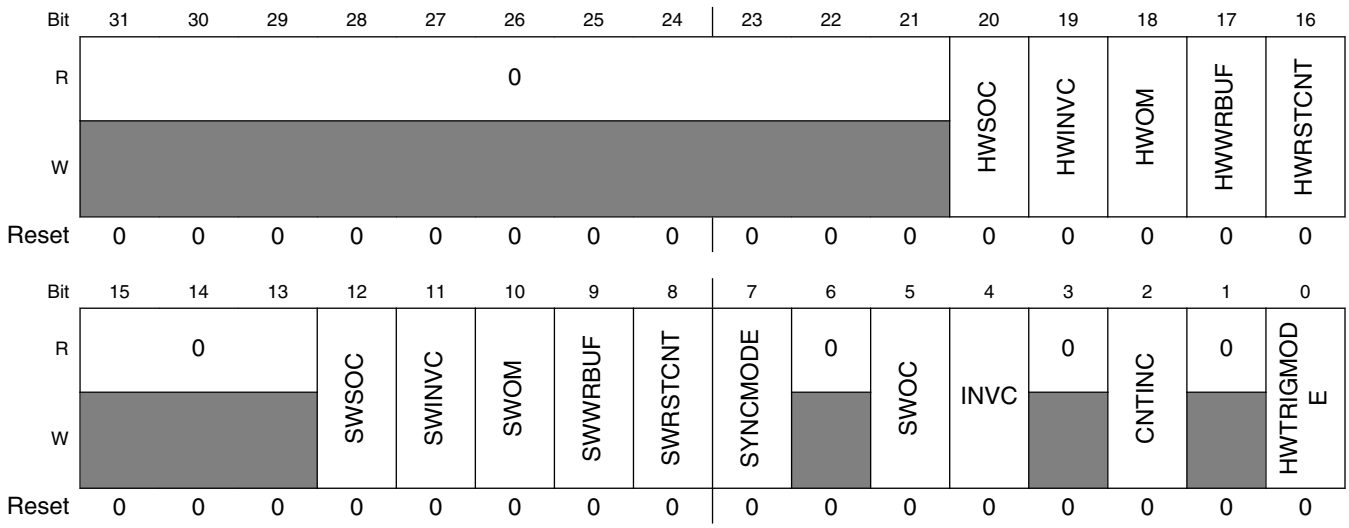
## FTMx_FLTPOL field descriptions (continued)

| Field | Description |
|---|---|
| 3<br>FLT3POL | Fault Input 3 Polarity<br>Defines the polarity of the fault input.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The fault input polarity is active high. A one at the fault input indicates a fault.<br>1 The fault input polarity is active low. A zero at the fault input indicates a fault. |
| 2<br>FLT2POL | Fault Input 2 Polarity<br>Defines the polarity of the fault input.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The fault input polarity is active high. A one at the fault input indicates a fault.<br>1 The fault input polarity is active low. A zero at the fault input indicates a fault. |
| 1<br>FLT1POL | Fault Input 1 Polarity<br>Defines the polarity of the fault input.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The fault input polarity is active high. A one at the fault input indicates a fault.<br>1 The fault input polarity is active low. A zero at the fault input indicates a fault. |
| 0<br>FLT0POL | Fault Input 0 Polarity<br>Defines the polarity of the fault input.<br>This field is write protected. It can be written only when MODE[WPDIS] = 1.<br>0 The fault input polarity is active high. A one at the fault input indicates a fault.<br>1 The fault input polarity is active low. A zero at the fault input indicates a fault. |

## 40.3.24 Synchronization Configuration (FTMx_SYNCONF)

This register selects the PWM synchronization configuration, SWOCTRL, INVCTRL and CNTIN registers synchronization, if FTM clears the TRIGj bit, where j = 0, 1, 2, when the hardware trigger j is detected.

Address: Base address + 8Ch offset




**FTMx_SYNCONF field descriptions**

| Field | Description |
|---|---|
| 31–21<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 20<br>HWSOC | Software output control synchronization is activated by a hardware trigger.<br><br>0 A hardware trigger does not activate the SWOCTRL register synchronization.<br>1 A hardware trigger activates the SWOCTRL register synchronization. |
| 19<br>HWINVC | Inverting control synchronization is activated by a hardware trigger.<br><br>0 A hardware trigger does not activate the INVCTRL register synchronization.<br>1 A hardware trigger activates the INVCTRL register synchronization. |
| 18<br>HWOM | Output mask synchronization is activated by a hardware trigger.<br><br>0 A hardware trigger does not activate the OUTMASK register synchronization.<br>1 A hardware trigger activates the OUTMASK register synchronization. |
| 17<br>HWWRBUF | MOD, CNTIN, and CV registers synchronization is activated by a hardware trigger.<br><br>0 A hardware trigger does not activate MOD, CNTIN, and CV registers synchronization.<br>1 A hardware trigger activates MOD, CNTIN, and CV registers synchronization. |
| 16<br>HWRSTCNT | FTM counter synchronization is activated by a hardware trigger.<br><br>0 A hardware trigger does not activate the FTM counter synchronization.<br>1 A hardware trigger activates the FTM counter synchronization. |

*Table continues on the next page...*

## FTMx_SYNCONF field descriptions (continued)

| Field | Description |
|---|---|
| 15–13<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 12<br>SWSOC | Software output control synchronization is activated by the software trigger.<br><br>0 The software trigger does not activate the SWOCTRL register synchronization.<br>1 The software trigger activates the SWOCTRL register synchronization. |
| 11<br>SWINVC | Inverting control synchronization is activated by the software trigger.<br><br>0 The software trigger does not activate the INVCTRL register synchronization.<br>1 The software trigger activates the INVCTRL register synchronization. |
| 10<br>SWOM | Output mask synchronization is activated by the software trigger.<br><br>0 The software trigger does not activate the OUTMASK register synchronization.<br>1 The software trigger activates the OUTMASK register synchronization. |
| 9<br>SWWRBUF | MOD, CNTIN, and CV registers synchronization is activated by the software trigger.<br><br>0 The software trigger does not activate MOD, CNTIN, and CV registers synchronization.<br>1 The software trigger activates MOD, CNTIN, and CV registers synchronization. |
| 8<br>SWRSTCNT | FTM counter synchronization is activated by the software trigger.<br><br>0 The software trigger does not activate the FTM counter synchronization.<br>1 The software trigger activates the FTM counter synchronization. |
| 7<br>SYNCMODE | Synchronization Mode<br><br>Selects the PWM Synchronization mode.<br><br>0 Legacy PWM synchronization is selected.<br>1 Enhanced PWM synchronization is selected. |
| 6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5<br>SWOC | SWOCTRL Register Synchronization<br><br>0 SWOCTRL register is updated with its buffer value at all rising edges of system clock.<br>1 SWOCTRL register is updated with its buffer value by the PWM synchronization. |
| 4<br>INVC | INVCTRL Register Synchronization<br><br>0 INVCTRL register is updated with its buffer value at all rising edges of system clock.<br>1 INVCTRL register is updated with its buffer value by the PWM synchronization. |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>CNTINC | CNTIN Register Synchronization<br><br>0 CNTIN register is updated with its buffer value at all rising edges of system clock.<br>1 CNTIN register is updated with its buffer value by the PWM synchronization. |
| 1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>HWTRIGMODE | Hardware Trigger Mode |

*Table continues on the next page...*

**FTMx_SYNCONF field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 FTM clears the TRIGj bit when the hardware trigger j is detected, where j = 0, 1,2.<br>1 FTM does not clear the TRIGj bit when the hardware trigger j is detected, where j = 0, 1,2. |

## 40.3.25 FTM Inverting Control (FTMx_INVCTRL)

This register controls when the channel (n) output becomes the channel (n+1) output, and channel (n+1) output becomes the channel (n) output. Each INVmEN bit enables the inverting operation for the corresponding pair channels m.

This register has a write buffer. The INVmEN bit is updated by the INVCTRL register synchronization.

Address: Base address + 90h offset

| Bit | 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | INV3EN | INV2EN | INV1EN | INV0EN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTMx_INVCTRL field descriptions**

| Field | Description |
|---|---|
| 31–4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3<br>INV3EN | Pair Channels 3 Inverting Enable<br><br>0 Inverting is disabled.<br>1 Inverting is enabled. |
| 2<br>INV2EN | Pair Channels 2 Inverting Enable<br><br>0 Inverting is disabled.<br>1 Inverting is enabled. |
| 1<br>INV1EN | Pair Channels 1 Inverting Enable<br><br>0 Inverting is disabled.<br>1 Inverting is enabled. |

*Table continues on the next page...*

**FTMx_INVCTRL field descriptions (continued)**

| Field | Description |
|---|---|
| 0<br>INV0EN | Pair Channels 0 Inverting Enable<br><br>0 Inverting is disabled.<br>1 Inverting is enabled. |

## 40.3.26 FTM Software Output Control (FTMx_SWOCTRL)

This register enables software control of channel (n) output and defines the value forced to the channel (n) output:

- The CHnOC bits enable the control of the corresponding channel (n) output by software.
- The CHnOCV bits select the value that is forced at the corresponding channel (n) output.

This register has a write buffer. The fields are updated by the SWOCTRL register synchronization.

Address: Base address + 94h offset

| Bit | 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CH7OCV | CH6OCV | CH5OCV | CH4OCV | CH3OCV | CH2OCV | CH1OCV | CH0OCV | CH7OC | CH6OC | CH5OC | CH4OC | CH3OC | CH2OC | CH1OC | CH0OC |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTMx_SWOCTRL field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15<br>CH7OCV | Channel 7 Software Output Control Value<br><br>0 The software output control forces 0 to the channel output.<br>1 The software output control forces 1 to the channel output. |
| 14<br>CH6OCV | Channel 6 Software Output Control Value |

*Table continues on the next page...*

## FTMx_SWOCTRL field descriptions (continued)

| Field | Description |
|---|---|
| | 0 The software output control forces 0 to the channel output.<br>1 The software output control forces 1 to the channel output. |
| 13<br>CH5OCV | Channel 5 Software Output Control Value<br><br>0 The software output control forces 0 to the channel output.<br>1 The software output control forces 1 to the channel output. |
| 12<br>CH4OCV | Channel 4 Software Output Control Value<br><br>0 The software output control forces 0 to the channel output.<br>1 The software output control forces 1 to the channel output. |
| 11<br>CH3OCV | Channel 3 Software Output Control Value<br><br>0 The software output control forces 0 to the channel output.<br>1 The software output control forces 1 to the channel output. |
| 10<br>CH2OCV | Channel 2 Software Output Control Value<br><br>0 The software output control forces 0 to the channel output.<br>1 The software output control forces 1 to the channel output. |
| 9<br>CH1OCV | Channel 1 Software Output Control Value<br><br>0 The software output control forces 0 to the channel output.<br>1 The software output control forces 1 to the channel output. |
| 8<br>CH0OCV | Channel 0 Software Output Control Value<br><br>0 The software output control forces 0 to the channel output.<br>1 The software output control forces 1 to the channel output. |
| 7<br>CH7OC | Channel 7 Software Output Control Enable<br><br>0 The channel output is not affected by software output control.<br>1 The channel output is affected by software output control. |
| 6<br>CH6OC | Channel 6 Software Output Control Enable<br><br>0 The channel output is not affected by software output control.<br>1 The channel output is affected by software output control. |
| 5<br>CH5OC | Channel 5 Software Output Control Enable<br><br>0 The channel output is not affected by software output control.<br>1 The channel output is affected by software output control. |
| 4<br>CH4OC | Channel 4 Software Output Control Enable<br><br>0 The channel output is not affected by software output control.<br>1 The channel output is affected by software output control. |
| 3<br>CH3OC | Channel 3 Software Output Control Enable<br><br>0 The channel output is not affected by software output control.<br>1 The channel output is affected by software output control. |
| 2<br>CH2OC | Channel 2 Software Output Control Enable |

*Table continues on the next page...*

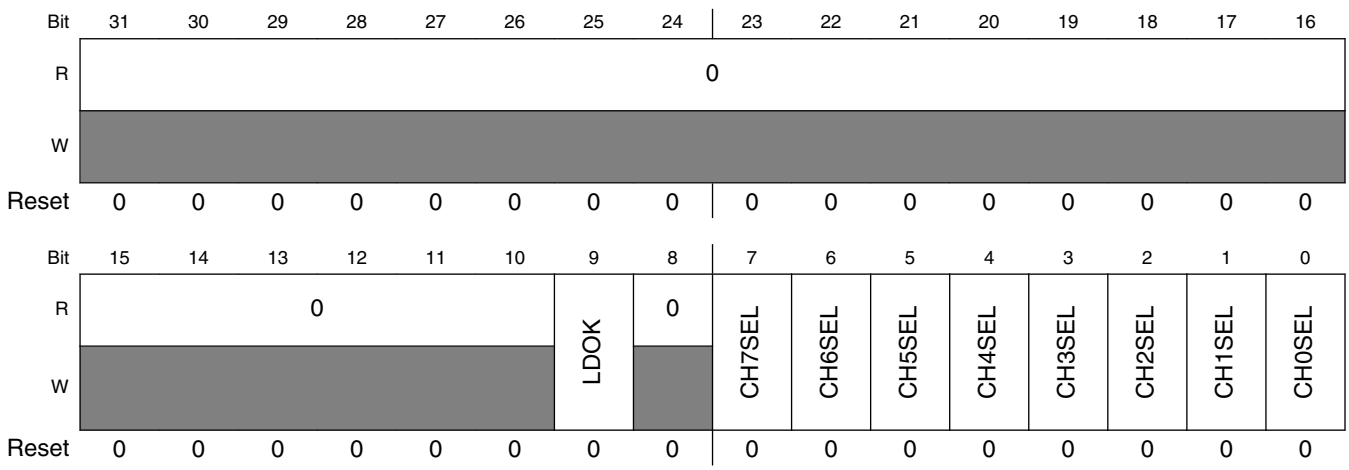
**FTMx_SWOCTRL field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 The channel output is not affected by software output control.<br>1 The channel output is affected by software output control. |
| 1<br>CH1OC | Channel 1 Software Output Control Enable<br><br>0 The channel output is not affected by software output control.<br>1 The channel output is affected by software output control. |
| 0<br>CH0OC | Channel 0 Software Output Control Enable<br><br>0 The channel output is not affected by software output control.<br>1 The channel output is affected by software output control. |

## 40.3.27 FTM PWM Load (FTMx_PWMLOAD)

Enables the loading of the MOD, CNTIN, C(n)V, and C(n+1)V registers with the values of their write buffers when the FTM counter changes from the MOD register value to its next value or when a channel (j) match occurs. A match occurs for the channel (j) when FTM counter = C(j)V.

Address: Base address + 98h offset




**FTMx_PWMLOAD field descriptions**

| Field | Description |
|---|---|
| 31–10<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 9<br>LDOK | Load Enable<br><br>Enables the loading of the MOD, CNTIN, and CV registers with the values of their write buffers.<br><br>0 Loading updated values is disabled.<br>1 Loading updated values is enabled. |

*Table continues on the next page...*

**FTMx_PWMLOAD field descriptions (continued)**

| Field | Description |
|---|---|
| 8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>CH7SEL | Channel 7 Select<br><br>0 Do not include the channel in the matching process.<br>1 Include the channel in the matching process. |
| 6<br>CH6SEL | Channel 6 Select<br><br>0 Do not include the channel in the matching process.<br>1 Include the channel in the matching process. |
| 5<br>CH5SEL | Channel 5 Select<br><br>0 Do not include the channel in the matching process.<br>1 Include the channel in the matching process. |
| 4<br>CH4SEL | Channel 4 Select<br><br>0 Do not include the channel in the matching process.<br>1 Include the channel in the matching process. |
| 3<br>CH3SEL | Channel 3 Select<br><br>0 Do not include the channel in the matching process.<br>1 Include the channel in the matching process. |
| 2<br>CH2SEL | Channel 2 Select<br><br>0 Do not include the channel in the matching process.<br>1 Include the channel in the matching process. |
| 1<br>CH1SEL | Channel 1 Select<br><br>0 Do not include the channel in the matching process.<br>1 Include the channel in the matching process. |
| 0<br>CH0SEL | Channel 0 Select<br><br>0 Do not include the channel in the matching process.<br>1 Include the channel in the matching process. |

## 40.4 Functional description

The notation used in this document to represent the counters and the generation of the signals is shown in the following figure.

FTM counting is up.
Channel (n) is in high-true EPWM mode.

PS[2:0] = 001
CNTIN = 0x0000
MOD = 0x0004
CnV = 0x0002

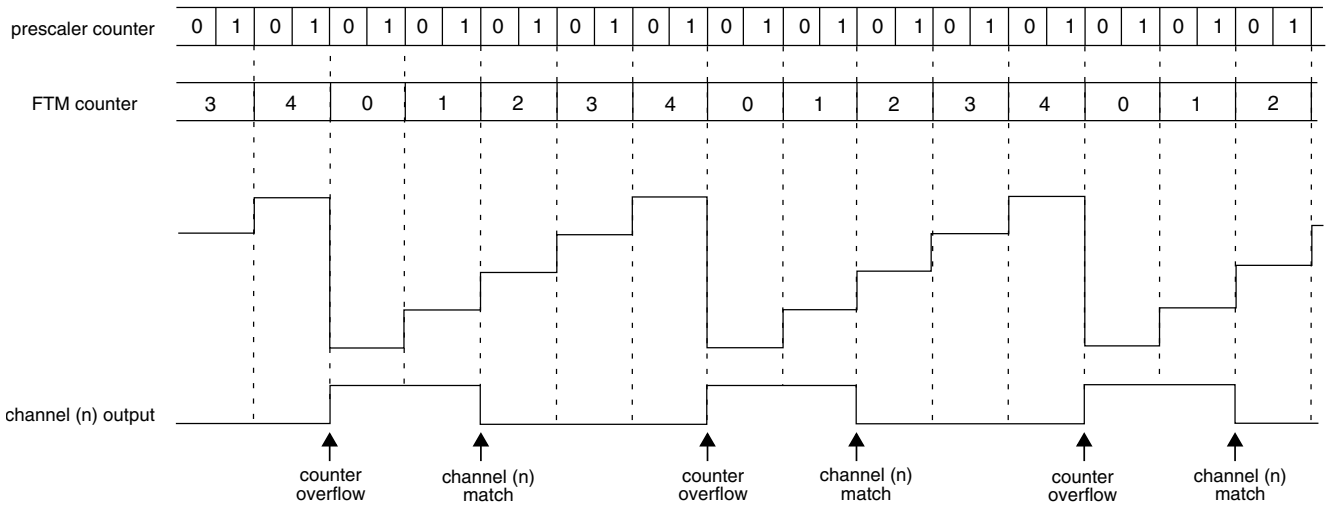



**Figure 40-207. Notation used**

## 40.4.1 Clock source

The FTM has only one clock domain: the system clock.

### 40.4.1.1 Counter clock source

The CLKS[1:0] bits in the SC register select one of three possible clock sources for the FTM counter or disable the FTM counter. After any MCU reset, CLKS[1:0] = 0:0 so no clock source is selected.

The CLKS[1:0] bits may be read or written at any time. Disabling the FTM counter by writing 0:0 to the CLKS[1:0] bits does not affect the FTM counter value or other registers.

The fixed frequency clock is an alternative clock source for the FTM counter that allows the selection of a clock other than the system clock or an external clock. This clock input is defined by chip integration. Refer to the chip specific documentation for further information. Due to FTM hardware implementation limitations, the frequency of the fixed frequency clock must not exceed 1/2 of the system clock frequency.

The external clock passes through a synchronizer clocked by the system clock to assure that counter transitions are properly aligned to system clock transitions.Therefore, to meet Nyquist criteria considering also jitter, the frequency of the external clock source must not exceed 1/4 of the system clock frequency.

## 40.4.2 Prescaler

The selected counter clock source passes through a prescaler that is a 7-bit counter. The value of the prescaler is selected by the PS[2:0] bits. The following figure shows an example of the prescaler counter and FTM counter.

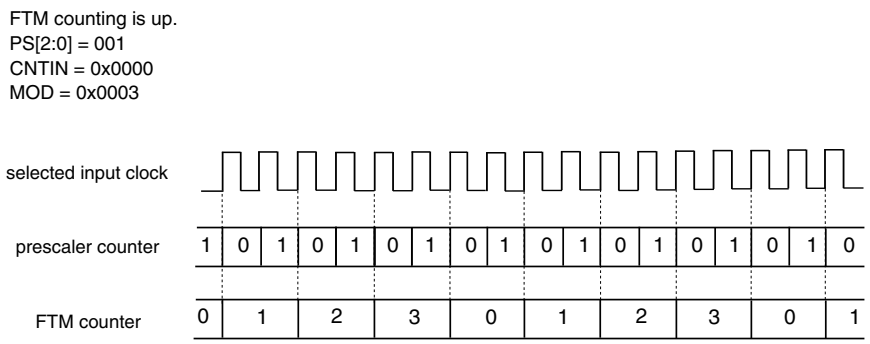



**Figure 40-208. Example of the prescaler counter**

## 40.4.3 Counter

The FTM has a 16-bit counter that is used by the channels either for input or output modes. The FTM counter clock is the selected clock divided by the prescaler.

The FTM counter has these modes of operation:

- Up counting
- Up-down counting
- Quadrature Decoder mode

### 40.4.3.1 Up counting

Up counting is selected when:

- QUADEN = 0, and
- CPWMS = 0

CNTIN defines the starting value of the count and MOD defines the final value of the count, see the following figure. The value of CNTIN is loaded into the FTM counter, and the counter increments until the value of MOD is reached, at which point the counter is reloaded with the value of CNTIN.

The FTM period when using up counting is (MOD – CNTIN + 0x0001) × period of the FTM counter clock.

The TOF bit is set when the FTM counter changes from MOD to CNTIN.

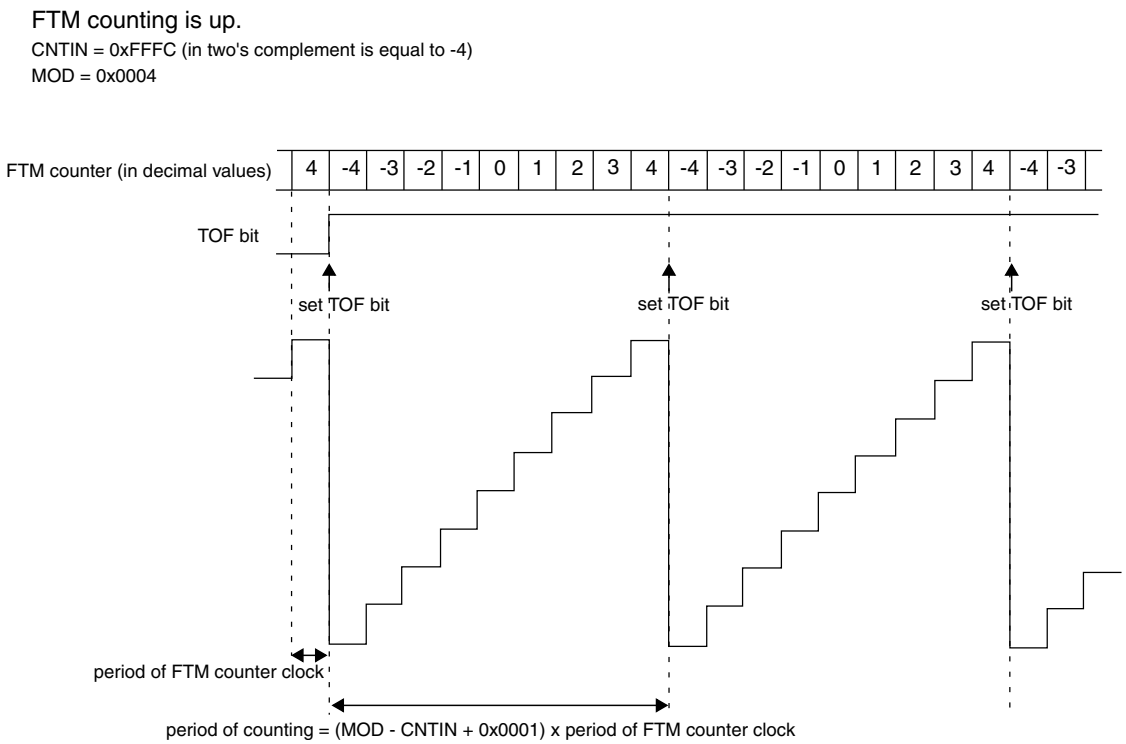



**Figure 40-209. Example of FTM up and signed counting**

**Table 40-302. FTM counting based on CNTIN value**

| When | Then |
|---|---|
| CNTIN = 0x0000 | The FTM counting is equivalent to TPM up counting, that is, up and unsigned counting. See the following figure. |
| CNTIN[15] = 1 | The initial value of the FTM counter is a negative number in two's complement, so the FTM counting is up and signed. |
| CNTIN[15] = 0 and CNTIN ≠ 0x0000 | The initial value of the FTM counter is a positive number, so the FTM counting is up and unsigned. |

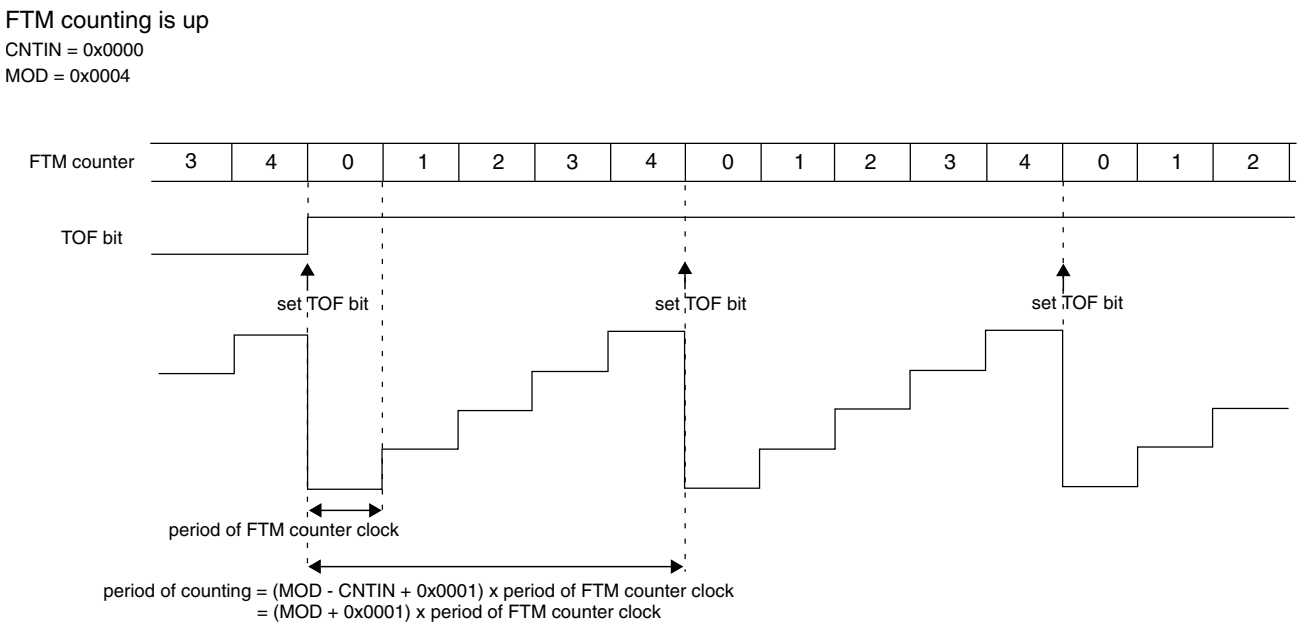



**Figure 40-210. Example of FTM up counting with CNTIN = 0x0000**

## Note

- FTM operation is only valid when the value of the CNTIN register is less than the value of the MOD register, either in the unsigned counting or signed counting. It is the responsibility of the software to ensure that the values in the CNTIN and MOD registers meet this requirement. Any values of CNTIN and MOD that do not satisfy this criteria can result in unpredictable behavior.
- MOD = CNTIN is a redundant condition. In this case, the FTM counter is always equal to MOD and the TOF bit is set in each rising edge of the FTM counter clock.
- When MOD = 0x0000, CNTIN = 0x0000, for example after reset, and FTMEN = 1, the FTM counter remains stopped at 0x0000 until a non-zero value is written into the MOD or CNTIN registers.
- Setting CNTIN to be greater than the value of MOD is not recommended as this unusual setting may make the FTM operation difficult to comprehend. However, there is no restriction on this configuration, and an example is shown in the following figure.

FTM counting is up
MOD = 0x0005
CNTIN = 0x0015

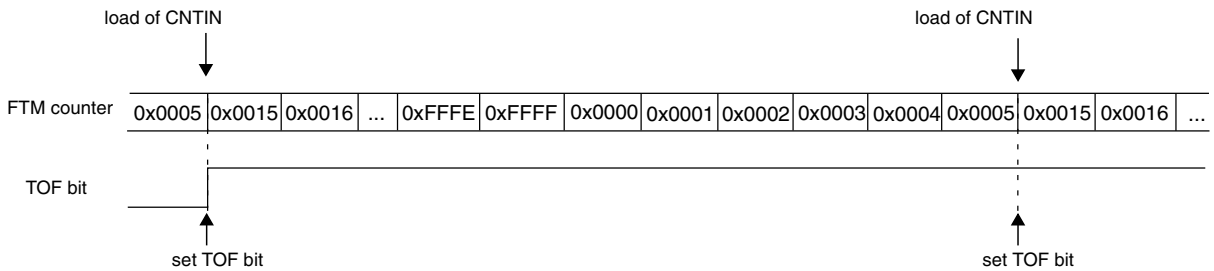



**Figure 40-211. Example of up counting when the value of CNTIN is greater than the value of MOD**

### 40.4.3.2 Up-down counting

Up-down counting is selected when:

- QUADEN = 0, and
- CPWMS = 1

CNTIN defines the starting value of the count and MOD defines the final value of the count. The value of CNTIN is loaded into the FTM counter, and the counter increments until the value of MOD is reached, at which point the counter is decremented until it returns to the value of CNTIN and the up-down counting restarts.

The FTM period when using up-down counting is 2 × (MOD – CNTIN) × period of the FTM counter clock.

The TOF bit is set when the FTM counter changes from MOD to (MOD – 1).

If (CNTIN = 0x0000), the FTM counting is equivalent to TPM up-down counting, that is, up-down and unsigned counting. See the following figure.

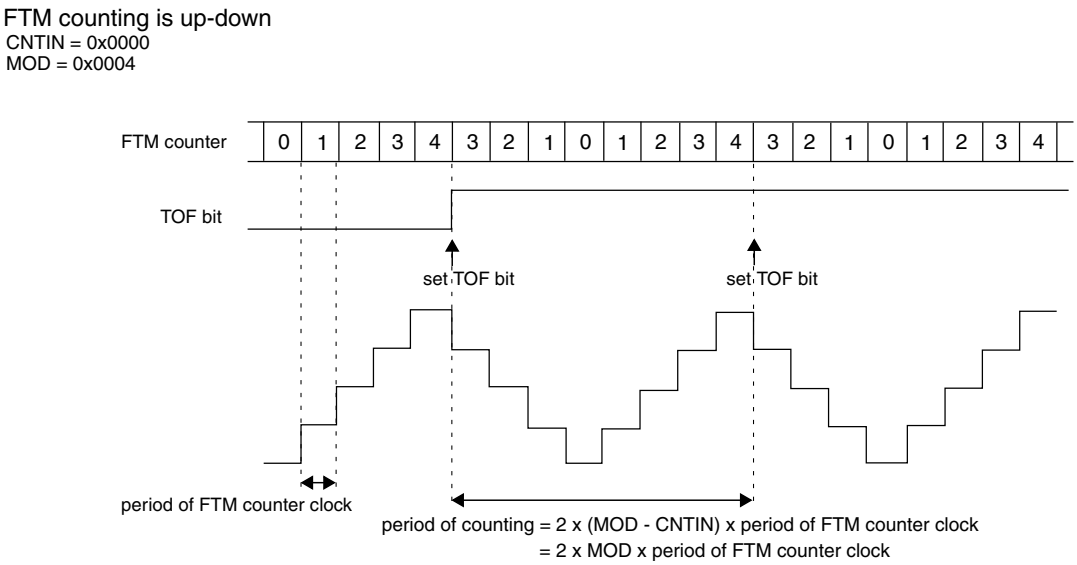



**Figure 40-212. Example of up-down counting when CNTIN = 0x0000**

### Note

It is expected that the up-down counting be used only with CNTIN = 0x0000.

## 40.4.3.3 Free running counter

If (FTMEN = 0) and (MOD = 0x0000 or MOD = 0xFFFF), the FTM counter is a free running counter. In this case, the FTM counter runs free from 0x0000 through 0xFFFF and the TOF bit is set when the FTM counter changes from 0xFFFF to 0x0000. See the following figure.

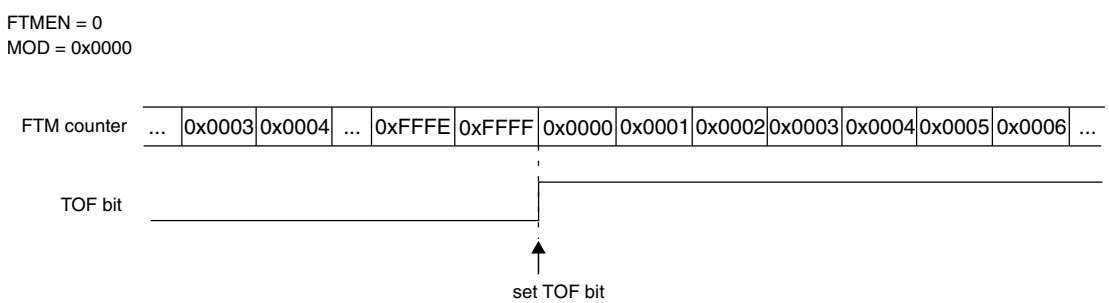



**Figure 40-213. Example when the FTM counter is free running**

The FTM counter is also a free running counter when:

- FTMEN = 1
- QUADEN = 0
- CPWMS = 0

- CNTIN = 0x0000, and
- MOD = 0xFFFF

### 40.4.3.4 Counter reset

Any one of the following cases resets the FTM counter to the value in the CNTIN register and the channels output to its initial value, except for channels in Output Compare mode.

- Any write to CNT.
- FTM counter synchronization.

### 40.4.3.5 When the TOF bit is set

The NUMTOF[4:0] bits define the number of times that the FTM counter overflow should occur before the TOF bit to be set. If NUMTOF[4:0] = 0x00, then the TOF bit is set at each FTM counter overflow.

Initialize the FTM counter, by writing to CNT, after writing to the NUMTOF[4:0] bits to avoid confusion about when the first counter overflow will occur.

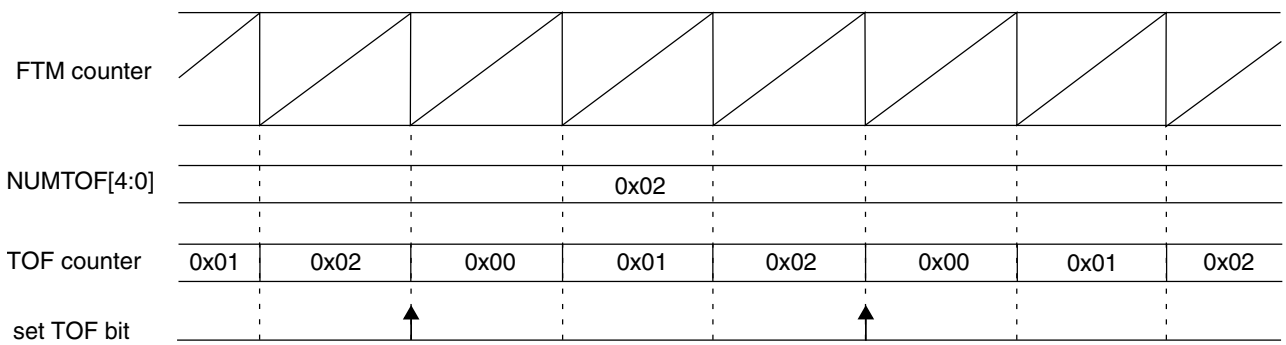



**Figure 40-214. Periodic TOF when NUMTOF = 0x02**

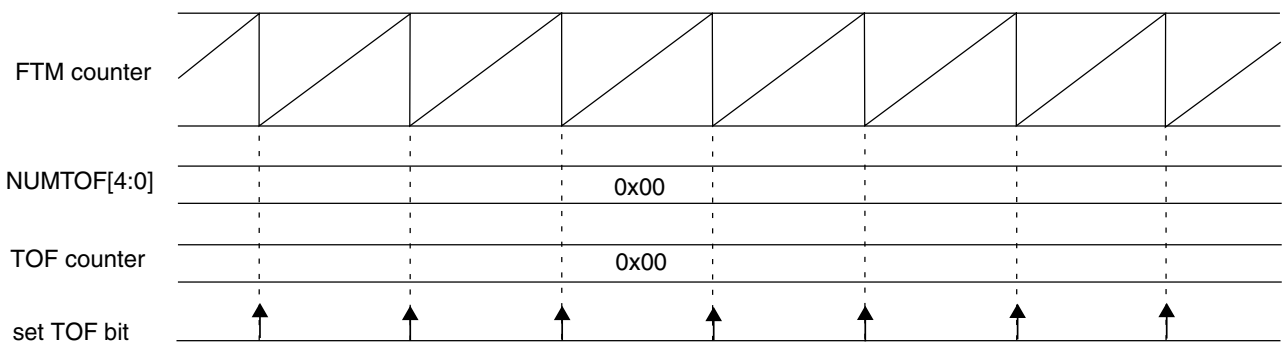



**Figure 40-215. Periodic TOF when NUMTOF = 0x00**

### 40.4.4 Input Capture mode

The Input Capture mode is selected when:

- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0
- MSnB:MSnA = 0:0, and
- ELSnB:ELSnA ≠ 0:0

When a selected edge occurs on the channel input, the current value of the FTM counter is captured into the CnV register, at the same time the CHnF bit is set and the channel interrupt is generated if enabled by CHnIE = 1. See the following figure.

When a channel is configured for input capture, the FTMxCHn pin is an edge-sensitive input. ELSnB:ELSnA control bits determine which edge, falling or rising, triggers input-capture event. Note that the maximum frequency for the channel input signal to be detected correctly is system clock divided by 4, which is required to meet Nyquist criteria for signal sampling.

Writes to the CnV register is ignored in Input Capture mode.

While in BDM, the input capture function works as configured. When a selected edge event occurs, the FTM counter value, which is frozen because of BDM, is captured into the CnV register and the CHnF bit is set.

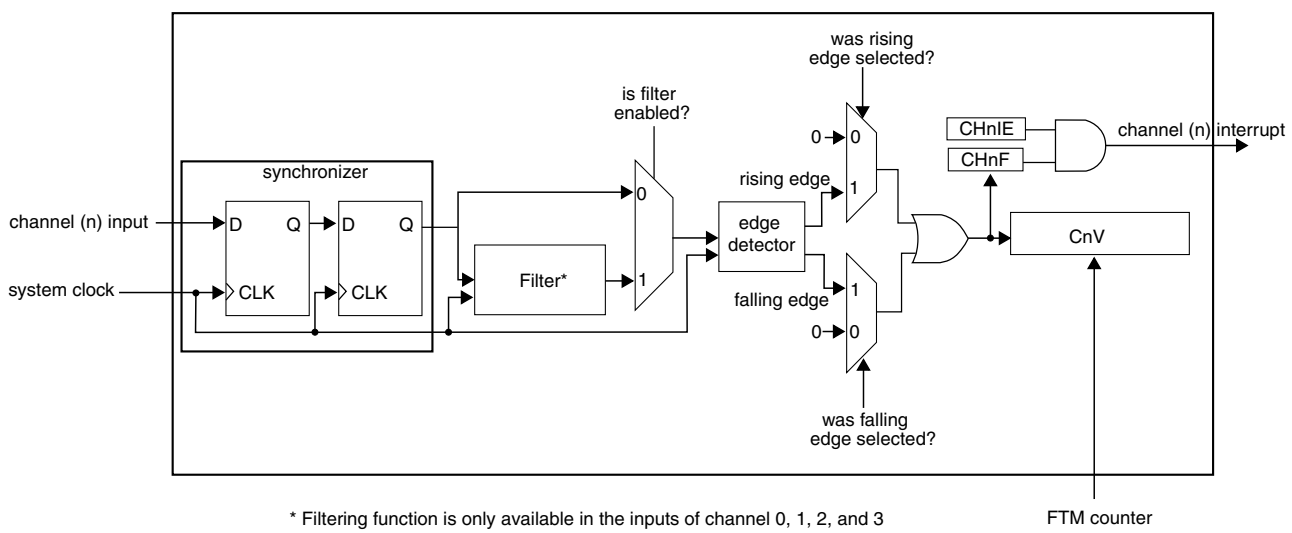



**Figure 40-216. Input Capture mode**

If the channel input does not have a filter enabled, then the input signal is always delayed 3 rising edges of the system clock, that is, two rising edges to the synchronizer plus one more rising edge to the edge detector. In other words, the CHnF bit is set on the third rising edge of the system clock after a valid edge occurs on the channel input.

**Note**

The Input Capture mode must be used only with CNTIN = 0x0000.

### 40.4.4.1 Filter for Input Capture mode

The filter function is only available on channels 0, 1, 2, and 3.

First, the input signal is synchronized by the system clock. Following synchronization, the input signal enters the filter block. See the following figure.

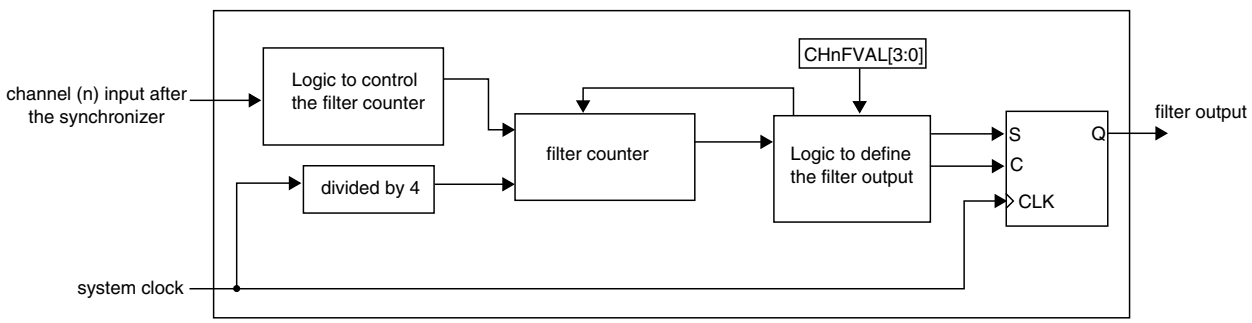



**Figure 40-217. Channel input filter**

When there is a state change in the input signal, the counter is reset and starts counting up. As long as the new state is stable on the input, the counter continues to increment. When the counter is equal to CHnFVAL[3:0], the state change of the input signal is validated. It is then transmitted as a pulse edge to the edge detector.

If the opposite edge appears on the input signal before it can be validated, the counter is reset. At the next input transition, the counter starts counting again. Any pulse that is shorter than the minimum value selected by CHnFVAL[3:0] (× 4 system clocks) is regarded as a glitch and is not passed on to the edge detector. A timing diagram of the input filter is shown in the following figure.

The filter function is disabled when CHnFVAL[3:0] bits are zero. In this case, the input signal is delayed 3 rising edges of the system clock. If (CHnFVAL[3:0] ≠ 0000), then the input signal is delayed by the minimum pulse width (CHnFVAL[3:0] × 4 system clocks) plus a further 4 rising edges of the system clock: two rising edges to the synchronizer,

one rising edge to the filter output, plus one more to the edge detector. In other words, CHnF is set (4 + 4 × CHnFVAL[3:0]) system clock periods after a valid edge occurs on the channel input.

The clock for the counter in the channel input filter is the system clock divided by 4.

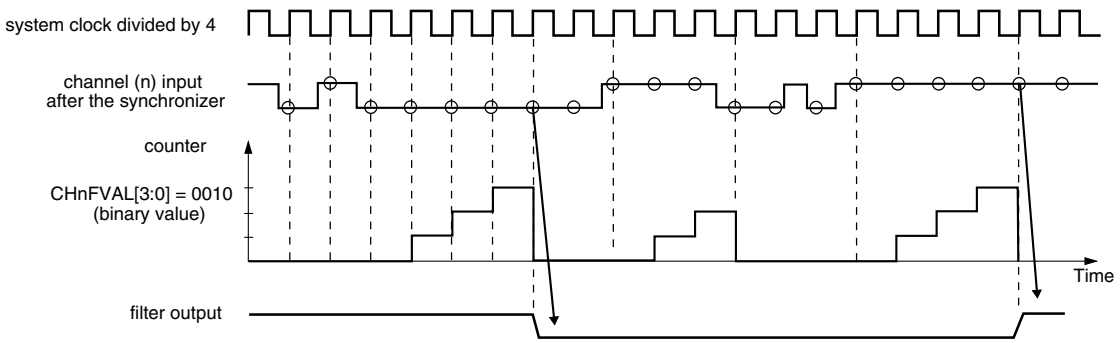



**Figure 40-218. Channel input filter example**

## 40.4.5 Output Compare mode

The Output Compare mode is selected when:

- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0, and
- MSnB:MSnA = 0:1

In Output Compare mode, the FTM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CnV register of an output compare channel, the channel (n) output can be set, cleared, or toggled.

When a channel is initially configured to Toggle mode, the previous value of the channel output is held until the first output compare event occurs.

The CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1 at the channel (n) match (FTM counter = CnV).

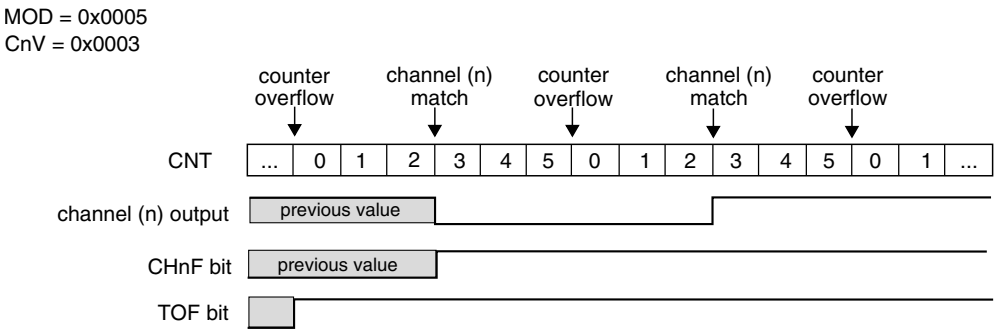



**Figure 40-219. Example of the Output Compare mode when the match toggles the channel output**

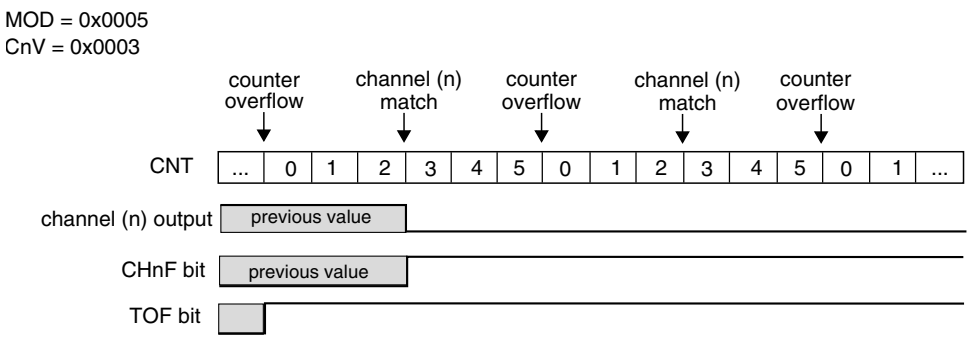



**Figure 40-220. Example of the Output Compare mode when the match clears the channel output**

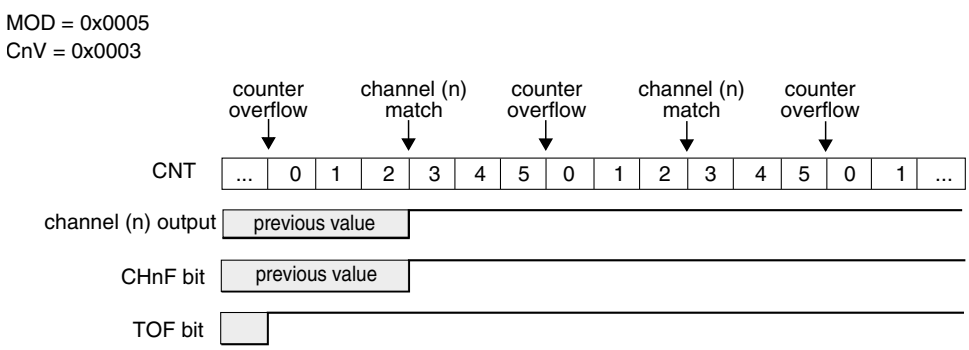



**Figure 40-221. Example of the Output Compare mode when the match sets the channel output**

If (ELSnB:ELSnA = 0:0) when the counter reaches the value in the CnV register, the CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1, however the channel (n) output is not modified and controlled by FTM.

## Note

The Output Compare mode must be used only with CNTIN = 0x0000.

### 40.4.6 Edge-Aligned PWM (EPWM) mode

The Edge-Aligned mode is selected when:

- QUADEN = 0
- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0, and
- MSnB = 1

The EPWM period is determined by (MOD − CNTIN + 0x0001) and the pulse width (duty cycle) is determined by (CnV − CNTIN).

The CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1 at the channel (n) match (FTM counter = CnV), that is, at the end of the pulse width.

This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within an FTM.

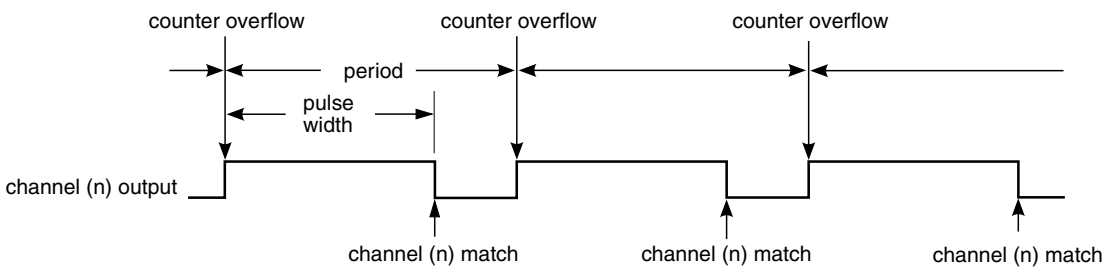



**Figure 40-222. EPWM period and pulse width with ELSnB:ELSnA = 1:0**

If (ELSnB:ELSnA = 0:0) when the counter reaches the value in the CnV register, the CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1, however the channel (n) output is not controlled by FTM.

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced high at the counter overflow when the CNTIN register value is loaded into the FTM counter, and it is forced low at the channel (n) match (FTM counter = CnV). See the following figure.

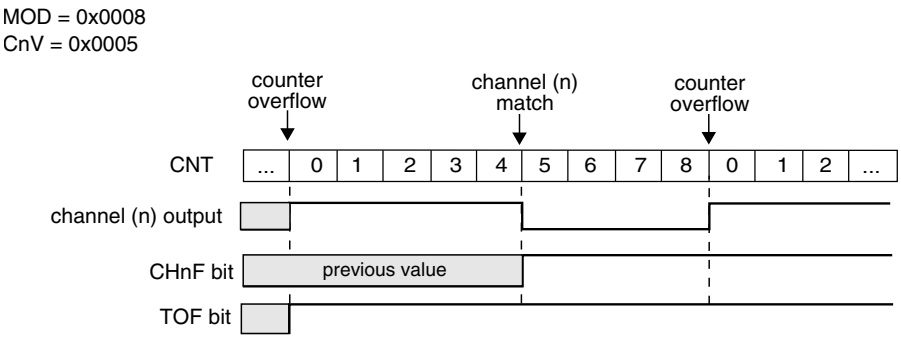



**Figure 40-223. EPWM signal with ELSnB:ELSnA = 1:0**

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced low at the counter overflow when the CNTIN register value is loaded into the FTM counter, and it is forced high at the channel (n) match (FTM counter = CnV). See the following figure.

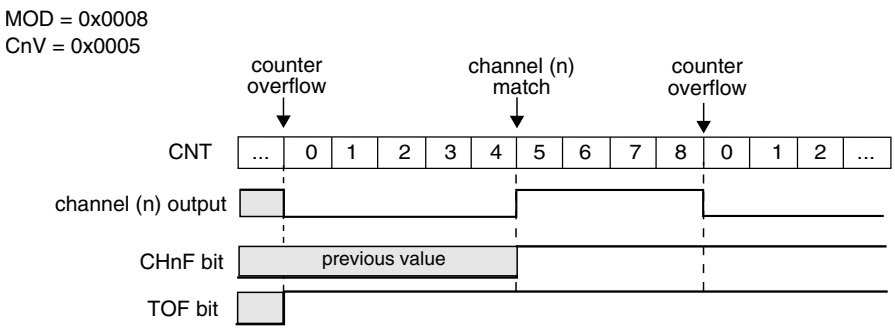



**Figure 40-224. EPWM signal with ELSnB:ELSnA = X:1**

If (CnV = 0x0000), then the channel (n) output is a 0% duty cycle EPWM signal and CHnF bit is not set even when there is the channel (n) match. If (CnV > MOD), then the channel (n) output is a 100% duty cycle EPWM signal and CHnF bit is not set even when there is the channel (n) match. Therefore, MOD must be less than 0xFFFF in order to get a 100% duty cycle EPWM signal.

**Note**

The EPWM mode must be used only with CNTIN = 0x0000.

## 40.4.7 Center-Aligned PWM (CPWM) mode

The Center-Aligned mode is selected when:

- QUADEN = 0
- DECAPEN = 0
- COMBINE = 0, and
- CPWMS = 1

The CPWM pulse width (duty cycle) is determined by 2 × (CnV − CNTIN) and the period is determined by 2 × (MOD − CNTIN). See the following figure. MOD must be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results.

In the CPWM mode, the FTM counter counts up until it reaches MOD and then counts down until it reaches CNTIN.

The CHnF bit is set and channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (FTM counter = CnV) when the FTM counting is down (at the begin of the pulse width) and when the FTM counting is up (at the end of the pulse width).

This type of PWM signal is called center-aligned because the pulse width centers for all channels are aligned with the value of CNTIN.

The other channel modes are not compatible with the up-down counter (CPWMS = 1). Therefore, all FTM channels must be used in CPWM mode when (CPWMS = 1).

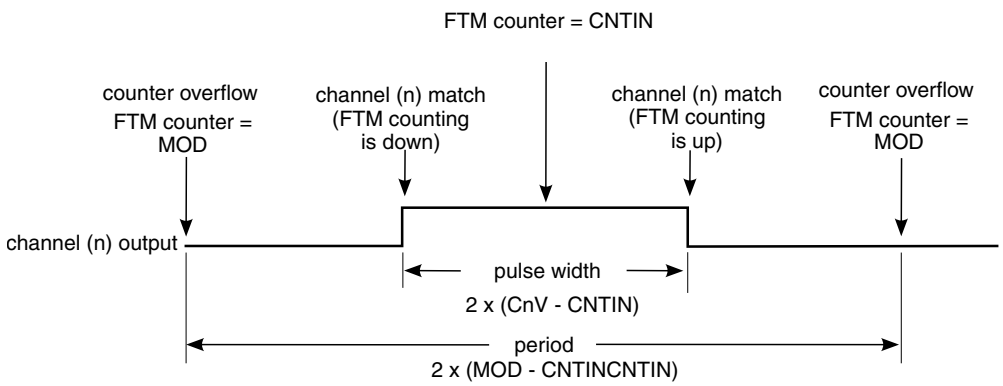



**Figure 40-225. CPWM period and pulse width with ELSnB:ELSnA = 1:0**

If (ELSnB:ELSnA = 0:0) when the FTM counter reaches the value in the CnV register, the CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1), however the channel (n) output is not controlled by FTM.

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced high at the channel (n) match (FTM counter = CnV) when counting down, and it is forced low at the channel (n) match when counting up. See the following figure.

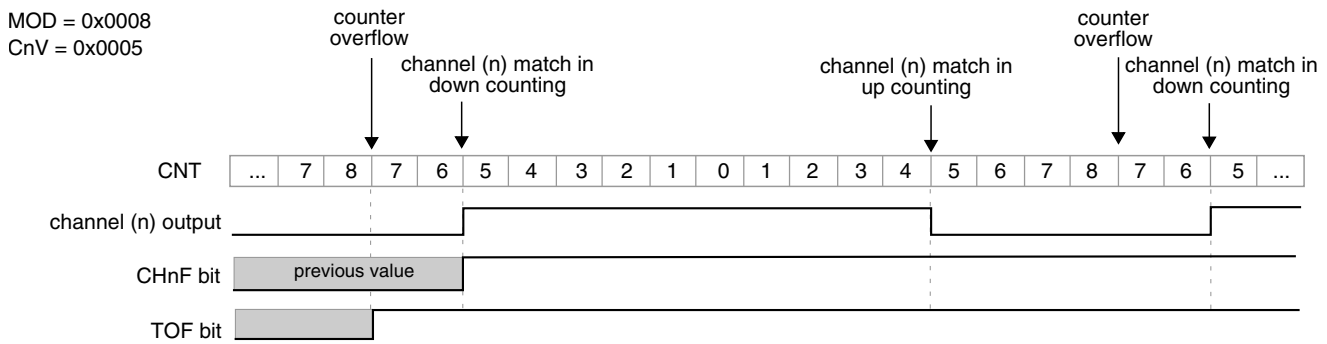



**Figure 40-226. CPWM signal with ELSnB:ELSnA = 1:0**

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced low at the channel (n) match (FTM counter = CnV) when counting down, and it is forced high at the channel (n) match when counting up. See the following figure.

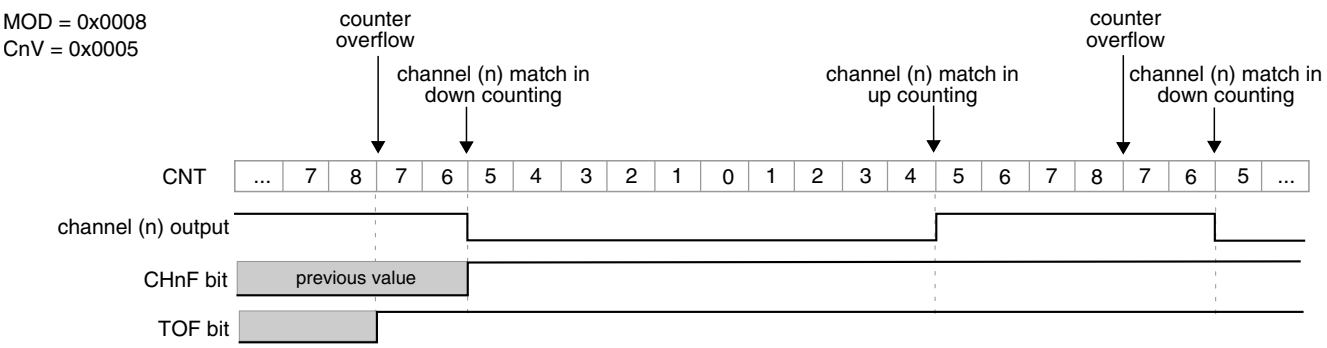



**Figure 40-227. CPWM signal with ELSnB:ELSnA = X:1**

If (CnV = 0x0000) or CnV is a negative value, that is (CnV[15] = 1), then the channel (n) output is a 0% duty cycle CPWM signal and CHnF bit is not set even when there is the channel (n) match.

If CnV is a positive value, that is (CnV[15] = 0), (CnV ≥ MOD), and (MOD ≠ 0x0000), then the channel (n) output is a 100% duty cycle CPWM signal and CHnF bit is not set even when there is the channel (n) match. This implies that the usable range of periods set by MOD is 0x0001 through 0x7FFE, 0x7FFF if you do not need to generate a 100% duty cycle CPWM signal. This is not a significant limitation because the resulting period is much longer than required for normal applications.

The CPWM mode must not be used when the FTM counter is a free running counter.

**Note**

The CPWM mode must be used only with CNTIN = 0x0000.

## 40.4.8 Combine mode

The Combine mode is selected when:

- FTMEN = 1
- QUADEN = 0
- DECAPEN = 0
- COMBINE = 1, and
- CPWMS = 0

In Combine mode, an even channel (n) and adjacent odd channel (n+1) are combined to generate a PWM signal in the channel (n) output.

In the Combine mode, the PWM period is determined by (MOD − CNTIN + 0x0001) and the PWM pulse width (duty cycle) is determined by (|C(n+1)V − C(n)V|).

The CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (FTM counter = C(n)V). The CH(n+1)F bit is set and the channel (n +1) interrupt is generated, if CH(n+1)IE = 1, at the channel (n+1) match (FTM counter = C(n+1)V).

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced low at the beginning of the period (FTM counter = CNTIN) and at the channel (n+1) match (FTM counter = C(n +1)V). It is forced high at the channel (n) match (FTM counter = C(n)V). See the following figure.

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced high at the beginning of the period (FTM counter = CNTIN) and at the channel (n+1) match (FTM counter = C(n +1)V). It is forced low at the channel (n) match (FTM counter = C(n)V). See the following figure.

In Combine mode, the ELS(n+1)B and ELS(n+1)A bits are not used in the generation of the channels (n) and (n+1) output. However, if (ELSnB:ELSnA = 0:0) then the channel (n) output is not controlled by FTM, and if (ELS(n+1)B:ELS(n+1)A = 0:0) then the channel (n+1) output is not controlled by FTM.

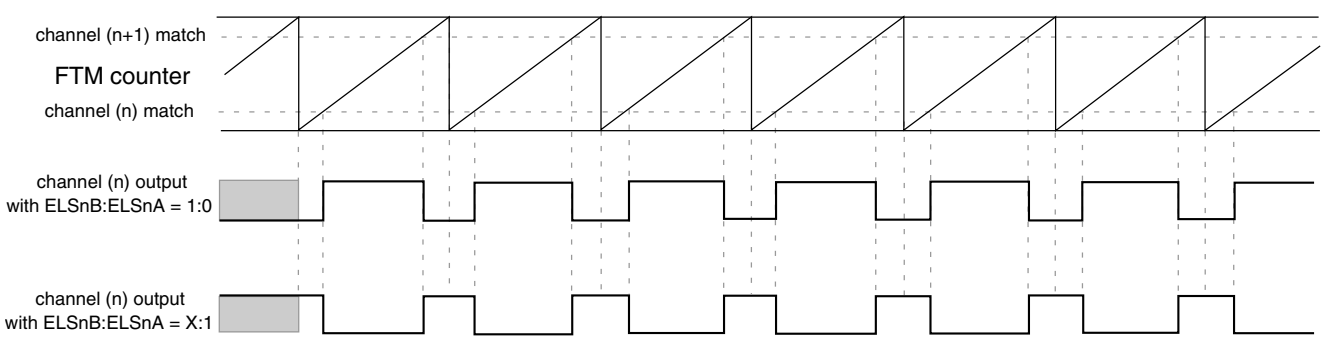



**Figure 40-228. Combine mode**

The following figures illustrate the PWM signals generation using Combine mode.

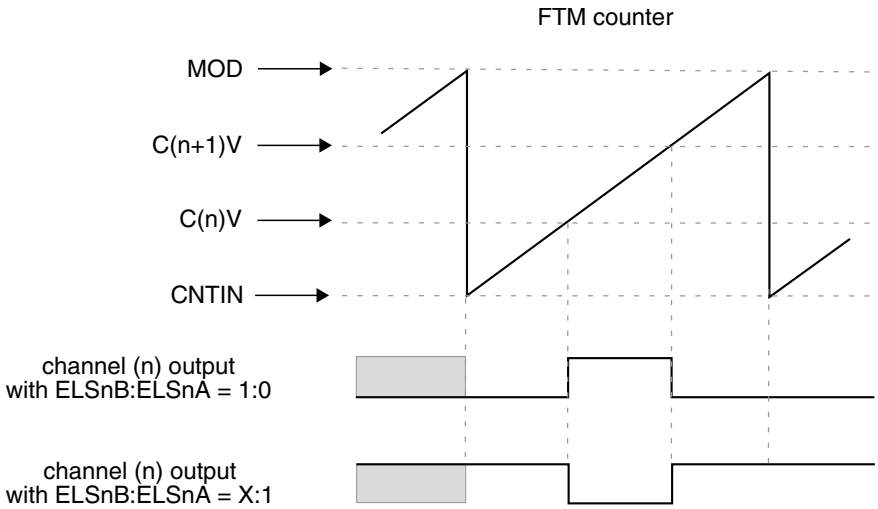



**Figure 40-229. Channel (n) output if (CNTIN < C(n)V < MOD) and (CNTIN < C(n+1)V < MOD) and (C(n)V < C(n+1)V)**

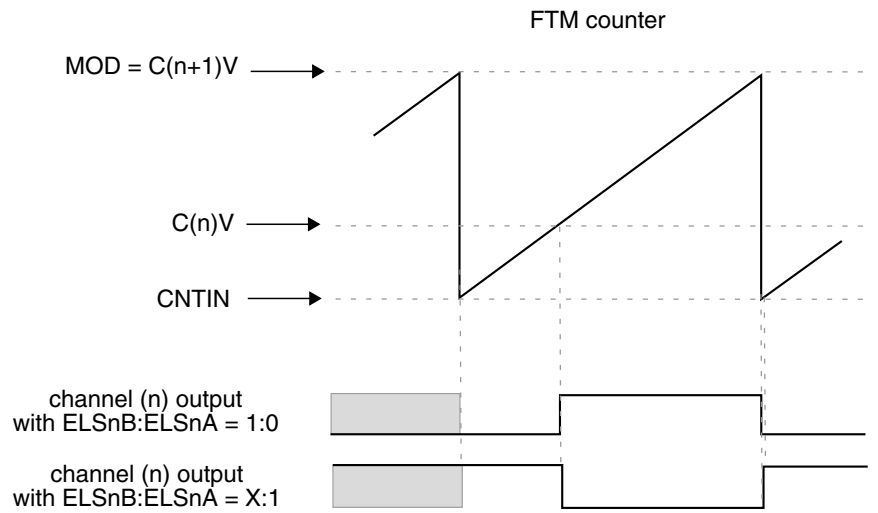



**Figure 40-230. Channel (n) output if (CNTIN < C(n)V < MOD) and (C(n+1)V = MOD)**

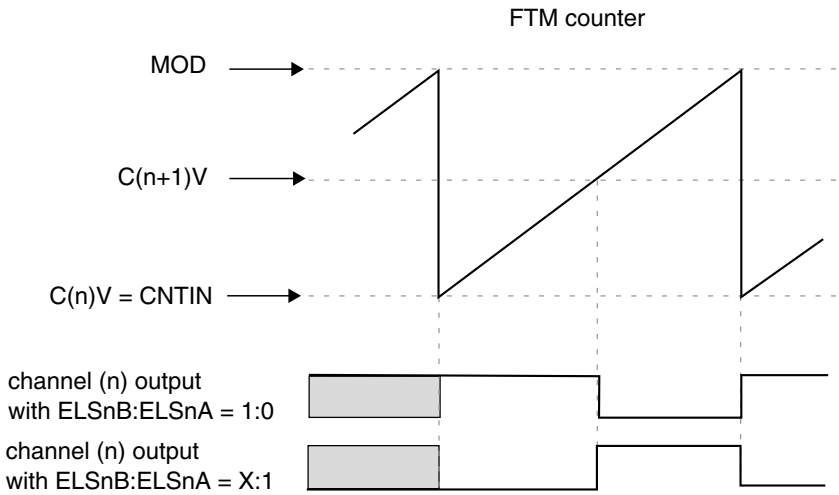



**Figure 40-231. Channel (n) output if (C(n)V = CNTIN) and (CNTIN < C(n+1)V < MOD)**

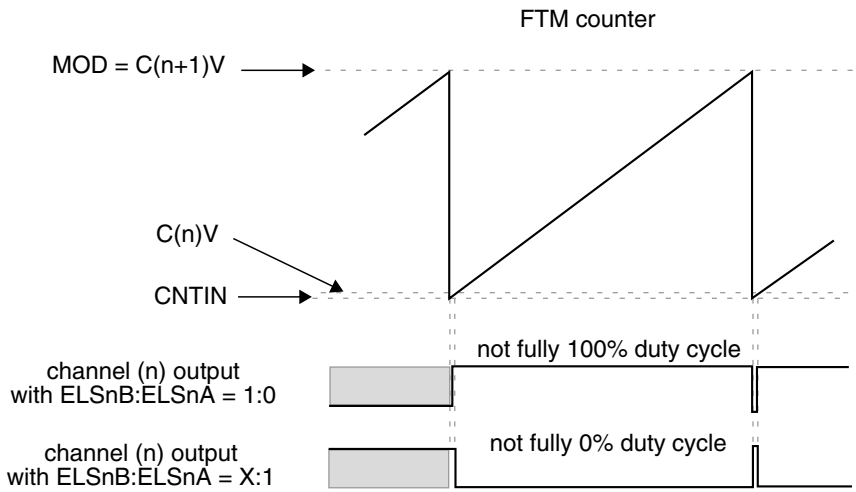



**Figure 40-232. Channel (n) output if (CNTIN < C(n)V < MOD) and (C(n)V is Almost Equal to CNTIN) and (C(n+1)V = MOD)**

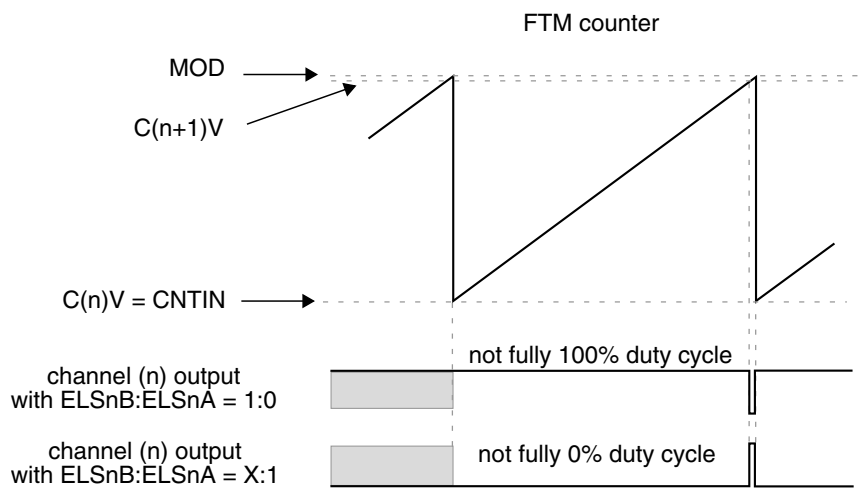



**Figure 40-233. Channel (n) output if (C(n)V = CNTIN) and (CNTIN < C(n+1)V < MOD) and (C(n+1)V is Almost Equal to MOD)**

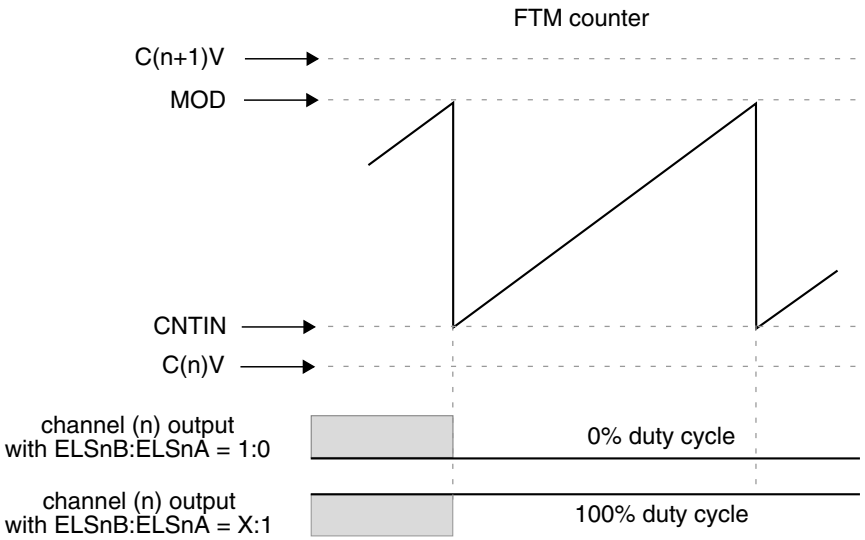



**Figure 40-234. Channel (n) output if C(n)V and C(n+1)V are not between CNTIN and MOD**

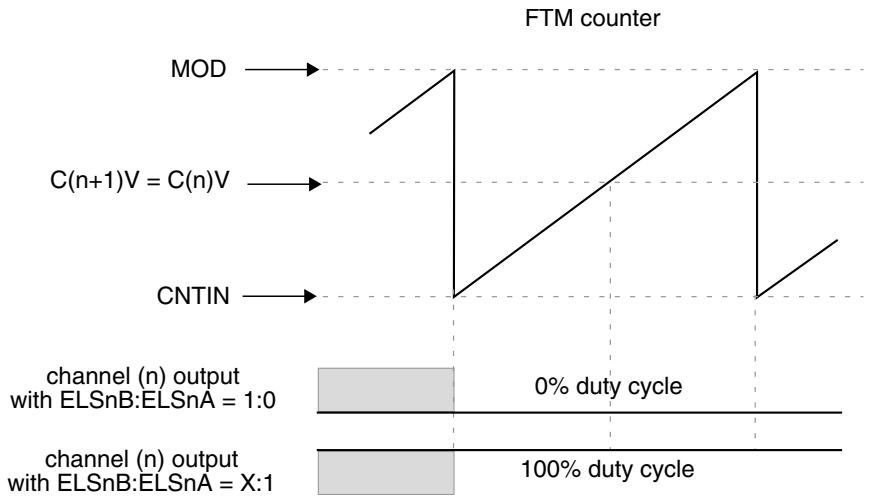



**Figure 40-235. Channel (n) output if (CNTIN < C(n)V < MOD) and (CNTIN < C(n+1)V < MOD) and (C(n)V = C(n+1)V)**

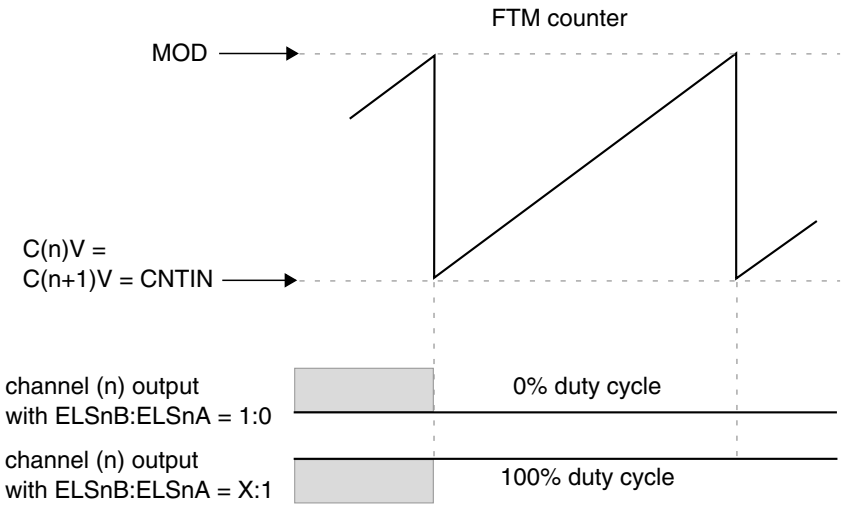



**Figure 40-236. Channel (n) output if (C(n)V = C(n+1)V = CNTIN)**

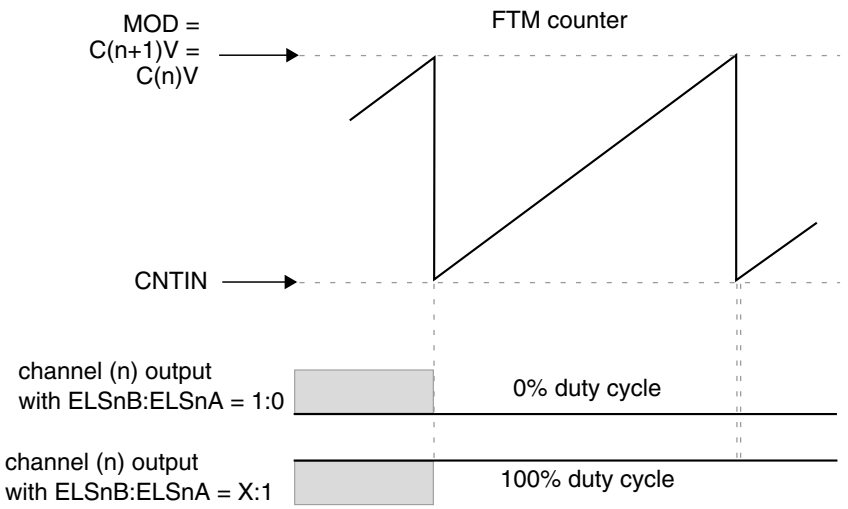



**Figure 40-237. Channel (n) output if (C(n)V = C(n+1)V = MOD)**

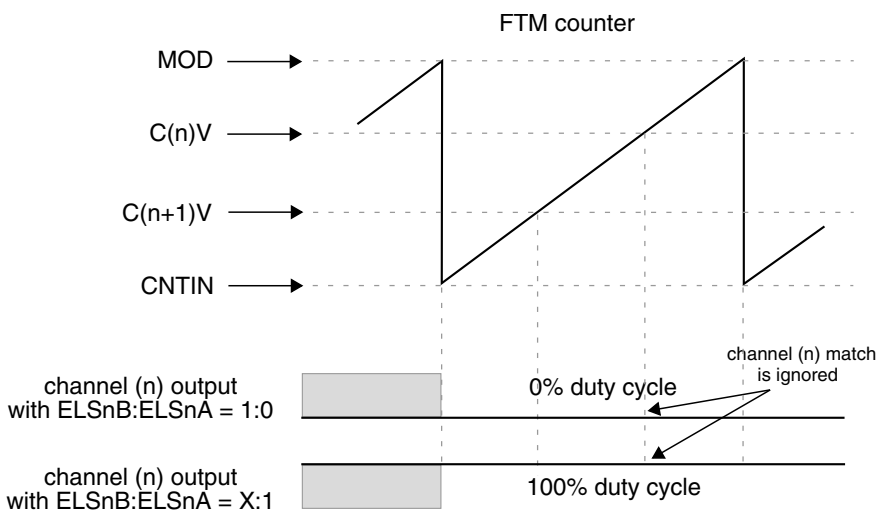



**Figure 40-238. Channel (n) output if (CNTIN < C(n)V < MOD) and (CNTIN < C(n+1)V < MOD) and (C(n)V > C(n+1)V)**

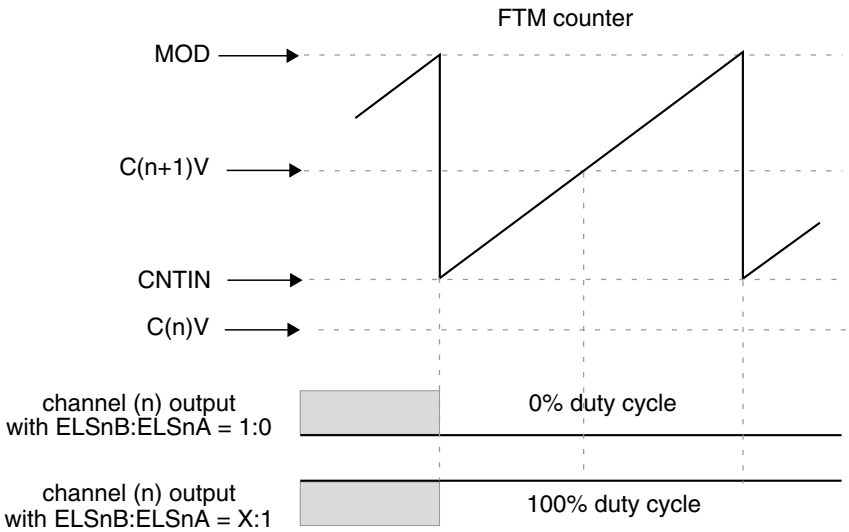



**Figure 40-239. Channel (n) output if (C(n)V < CNTIN) and (CNTIN < C(n+1)V < MOD)**

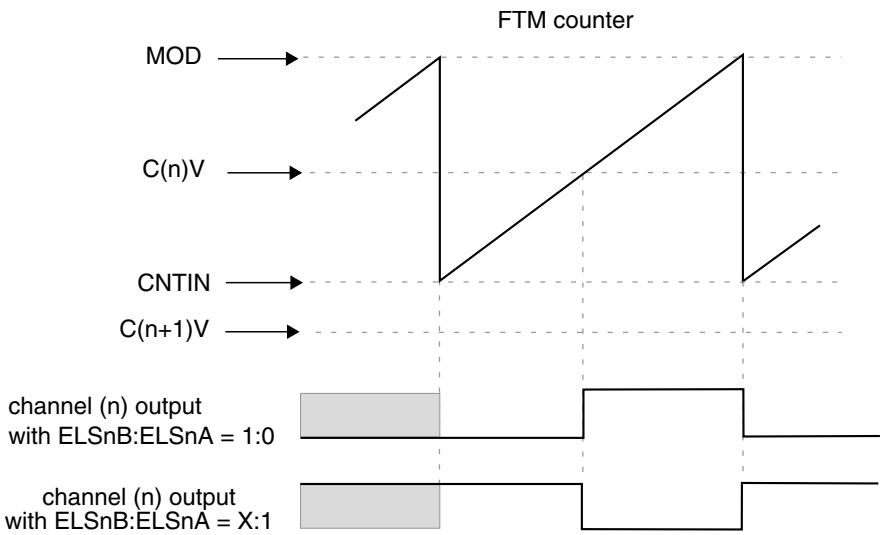



**Figure 40-240. Channel (n) output if (C(n+1)V < CNTIN) and (CNTIN < C(n)V < MOD)**

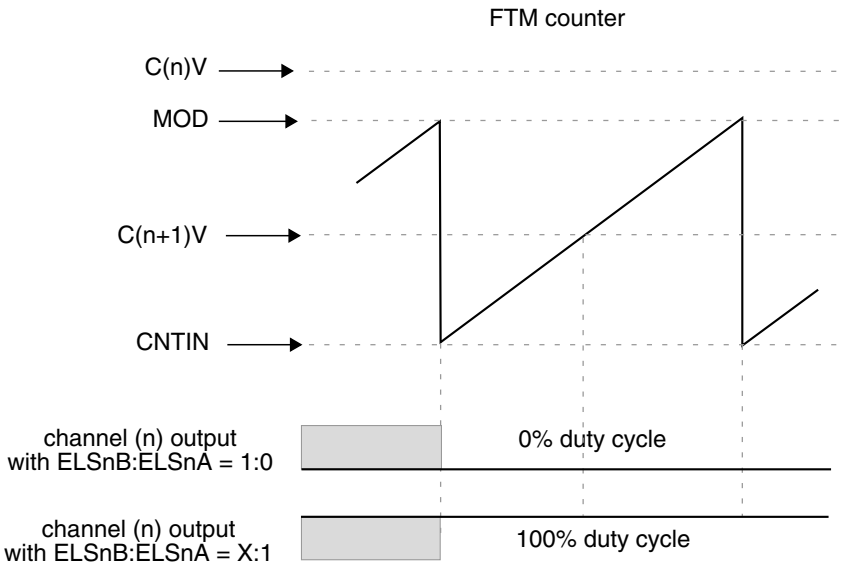



**Figure 40-241. Channel (n) output if (C(n)V > MOD) and (CNTIN < C(n+1)V < MOD)**

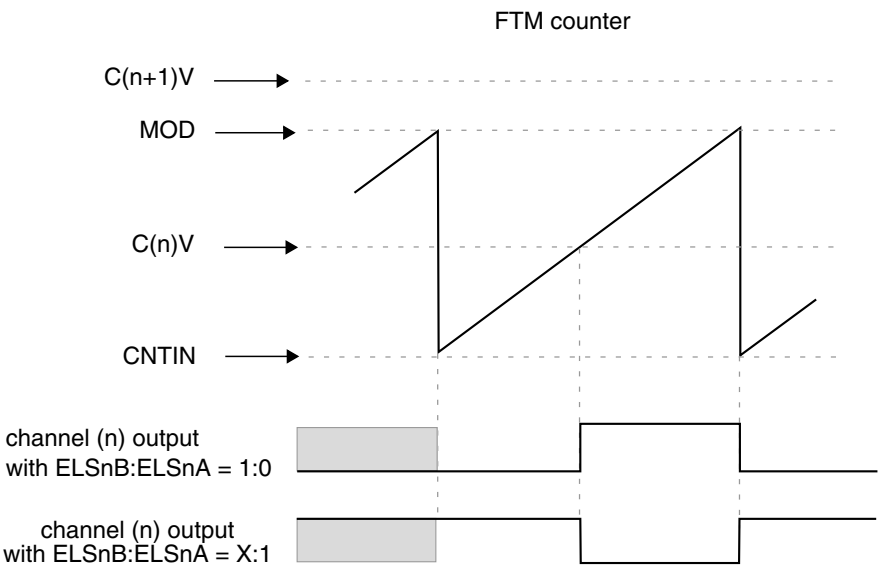



**Figure 40-242. Channel (n) output if (C(n+1)V > MOD) and (CNTIN < C(n)V < MOD)**

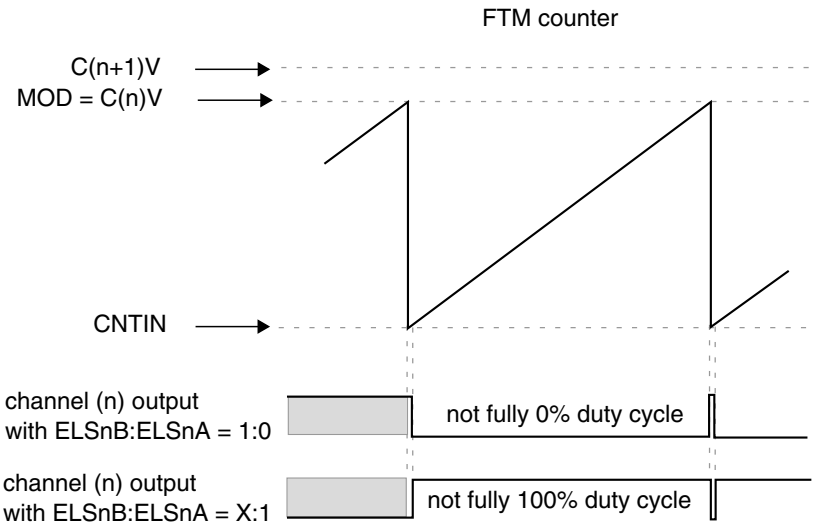



**Figure 40-243. Channel (n) output if (C(n+1)V > MOD) and (CNTIN < C(n)V = MOD)**

### 40.4.8.1 Asymmetrical PWM

In Combine mode, the control of the PWM signal first edge, when the channel (n) match occurs, that is, FTM counter = C(n)V, is independent of the control of the PWM signal second edge, when the channel (n+1) match occurs, that is, FTM counter = C(n+1)V. So, Combine mode allows the generation of asymmetrical PWM signals.

## 40.4.9 Complementary mode

The Complementary mode is selected when:

- FTMEN = 1
- QUADEN = 0
- DECAPEN = 0
- COMBINE = 1
- CPWMS = 0, and
- COMP = 1

In Complementary mode, the channel (n+1) output is the inverse of the channel (n) output.

So, the channel (n+1) output is the same as the channel (n) output when:

- FTMEN = 1
- QUADEN = 0
- DECAPEN = 0
- COMBINE = 1

- CPWMS = 0, and
- COMP = 0

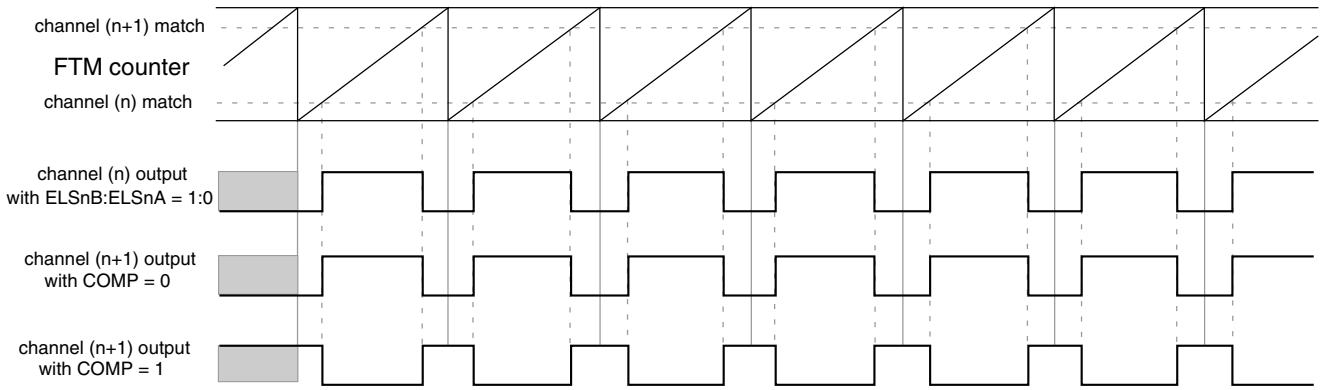



**Figure 40-244. Channel (n+1) output in Complementary mode with (ELSnB:ELSnA = 1:0)**

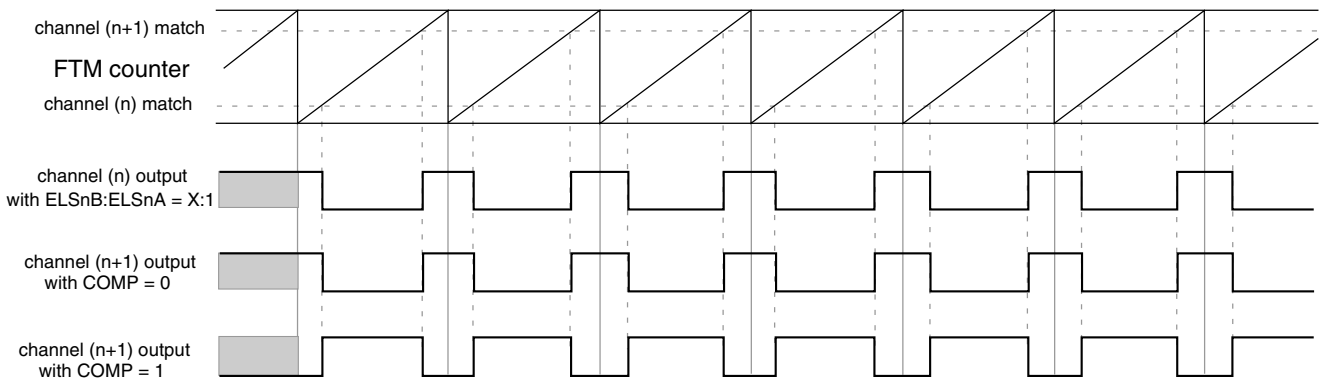



**Figure 40-245. Channel (n+1) output in Complementary mode with (ELSnB:ELSnA = X:1)**

## 40.4.10 Registers updated from write buffers

### 40.4.10.1 CNTIN register update

The following table describes when CNTIN register is updated:

**Table 40-303. CNTIN register update**

| When | Then CNTIN register is updated |
|---|---|
| CLKS[1:0] = 0:0 | When CNTIN register is written, independent of FTMEN bit. |
| • FTMEN = 0, or<br>• CNTINC = 0 | At the next system clock after CNTIN was written. |
| • FTMEN = 1,<br>• SYNCMODE = 1, and<br>• CNTINC = 1 | By the CNTIN register synchronization. |

### 40.4.10.2 MOD register update

The following table describes when MOD register is updated:

**Table 40-304. MOD register update**

| When | Then MOD register is updated |
|---|---|
| CLKS[1:0] = 0:0 | When MOD register is written, independent of FTMEN bit. |
| • CLKS[1:0] ≠ 0:0, and<br>• FTMEN = 0 | According to the CPWMS bit, that is:<br>• If the selected mode is not CPWM then MOD register is updated after MOD register was written and the FTM counter changes from MOD to CNTIN. If the FTM counter is at free-running counter mode then this update occurs when the FTM counter changes from 0xFFFF to 0x0000.<br>• If the selected mode is CPWM then MOD register is updated after MOD register was written and the FTM counter changes from MOD to (MOD – 0x0001). |
| • CLKS[1:0] ≠ 0:0, and<br>• FTMEN = 1 | By the MOD register synchronization. |

### 40.4.10.3 CnV register update

The following table describes when CnV register is updated:

**Table 40-305. CnV register update**

| When | Then CnV register is updated |
|---|---|
| CLKS[1:0] = 0:0 | When CnV register is written, independent of FTMEN bit. |
| • CLKS[1:0] ≠ 0:0, and<br>• FTMEN = 0 | According to the selected mode, that is:<br>• If the selected mode is Output Compare, then CnV register is updated on the next FTM counter change, end of the prescaler counting, after CnV register was written.<br>• If the selected mode is EPWM, then CnV register is updated after CnV register was written and the FTM counter changes from MOD to CNTIN. If the FTM counter is at free-running counter mode then this update occurs when the FTM counter changes from 0xFFFF to 0x0000.<br>• If the selected mode is CPWM, then CnV register is updated after CnV register was written and the FTM counter changes from MOD to (MOD – 0x0001). |
| • CLKS[1:0] ≠ 0:0, and<br>• FTMEN = 1 | According to the selected mode, that is:<br>• If the selected mode is output compare then CnV register is updated according to the SYNCEN bit. If (SYNCEN = 0) then CnV register is updated after CnV register was written at the next change of the FTM counter, the end of the prescaler counting. If (SYNCEN = 1) then CnV register is updated by the C(n)V and C(n+1)V register synchronization.<br>• If the selected mode is not output compare and (SYNCEN = 1) then CnV register is updated by the C(n)V and C(n+1)V register synchronization. |

### 40.4.11 PWM synchronization

The PWM synchronization provides an opportunity to update the MOD, CNTIN, CnV, OUTMASK, INVCTRL and SWOCTRL registers with their buffered value and force the FTM counter to the CNTIN register value.

**Note**

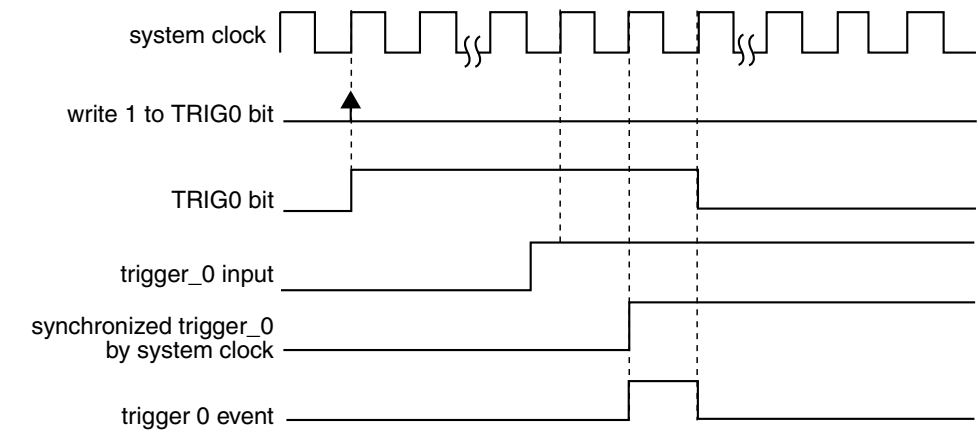
- The PWM synchronization must be used only in Combine mode.
- The legacy PWM synchronization (SYNCMODE = 0) is a subset of the enhanced PWM synchronization (SYNCMODE = 1). Thus, only the enhanced PWM synchronization must be used.

#### 40.4.11.1 Hardware trigger

Three hardware trigger signal inputs of the FTM module are enabled when TRIGn = 1, where n = 0, 1 or 2 corresponding to each one of the input signals, respectively. The hardware trigger input n is synchronized by the system clock. The PWM synchronization with hardware trigger is initiated when a rising edge is detected at the enabled hardware trigger inputs.

If (HWTRIGMODE = 0) then the TRIGn bit is cleared when 0 is written to it or when the trigger n event is detected.

In this case, if two or more hardware triggers are enabled (for example, TRIG0 and TRIG1 = 1) and only trigger 1 event occurs, then only TRIG1 bit is cleared. If a trigger n event occurs together with a write setting TRIGn bit, then the synchronization is initiated, but TRIGn bit remains set due to the write operation.




Note
All hardware trigger inputs have the same behavior.

**Figure 40-246. Hardware trigger event with HWTRIGMODE = 0**

If HWTRIGMODE = 1, then the TRIGn bit is only cleared when 0 is written to it.

**NOTE**

The HWTRIGMODE bit must be 1 only with enhanced PWM synchronization (SYNCMODE = 1).

### 40.4.11.2 Software trigger

A software trigger event occurs when 1 is written to the SYNC[SWSYNC] bit. The SWSYNC bit is cleared when 0 is written to it or when the PWM synchronization, initiated by the software event, is completed.

If another software trigger event occurs (by writing another 1 to the SWSYNC bit) at the same time the PWM synchronization initiated by the previous software trigger event is ending, a new PWM synchronization is started and the SWSYNC bit remains equal to 1.

If SYNCMODE = 0 then the SWSYNC bit is also cleared by FTM according to PWMSYNC and REINIT bits. In this case if (PWMSYNC = 1) or (PWMSYNC = 0 and REINIT = 0) then SWSYNC bit is cleared at the next selected loading point after that the software trigger event occurred; see Boundary cycle and loading points and the following figure. If (PWMSYNC = 0) and (REINIT = 1) then SWSYNC bit is cleared when the software trigger event occurs.

If SYNCMODE = 1 then the SWSYNC bit is also cleared by FTM according to the SWRSTCNT bit. If SWRSTCNT = 0 then SWSYNC bit is cleared at the next selected loading point after that the software trigger event occurred; see the following figure. If SWRSTCNT = 1 then SWSYNC bit is cleared when the software trigger event occurs.

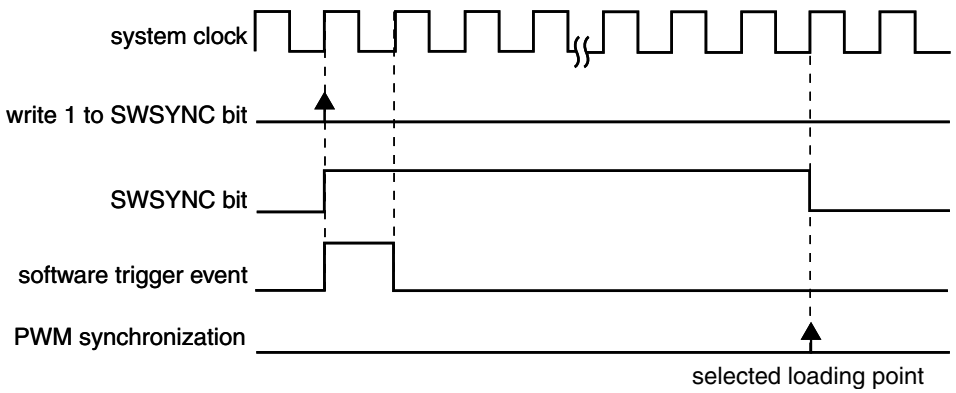



**Figure 40-247. Software trigger event**

### 40.4.11.3 Boundary cycle and loading points

The boundary cycle definition is important for the loading points for the registers MOD, CNTIN, and C(n)V.

In Up counting mode, the boundary cycle is defined as when the counter wraps to its initial value (CNTIN). If in Up-down counting mode, then the boundary cycle is defined as when the counter turns from down to up counting and when from up to down counting.

The following figure shows the boundary cycles and the loading points for the registers. In the Up Counting mode, the loading points are enabled if one of CNTMIN or CTMAX bits are 1. In the Up-Down Counting mode, the loading points are selected by CNTMIN and CNTMAX bits, as indicated in the figure. These loading points are safe places for register updates thus allowing a smooth transitions in PWM waveform generation.

For both counting modes, if neither CNTMIN nor CNTMAX are 1, then the boundary cycles are not used as loading points for registers updates. See the register synchronization descriptions in the following sections for details.

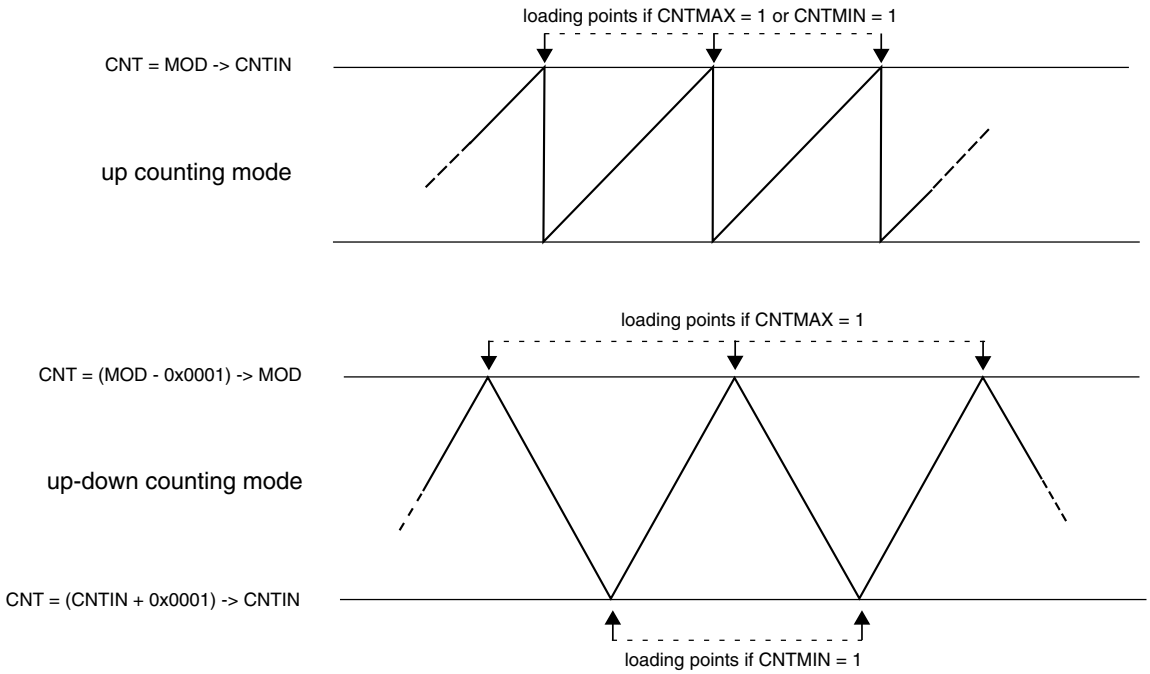



**Figure 40-248. Boundary cycles and loading points**

### 40.4.11.4 MOD register synchronization

The MOD register synchronization updates the MOD register with its buffer value. This synchronization is enabled if (FTMEN = 1).

The MOD register synchronization can be done by either the enhanced PWM synchronization (SYNCMODE = 1) or the legacy PWM synchronization (SYNCMODE = 0). However, it is expected that the MOD register be synchronized only by the enhanced PWM synchronization.

In the case of enhanced PWM synchronization, the MOD register synchronization depends on SWWRBUF, SWRSTCNT, HWWRBUF, and HWRSTCNT bits according to this flowchart:

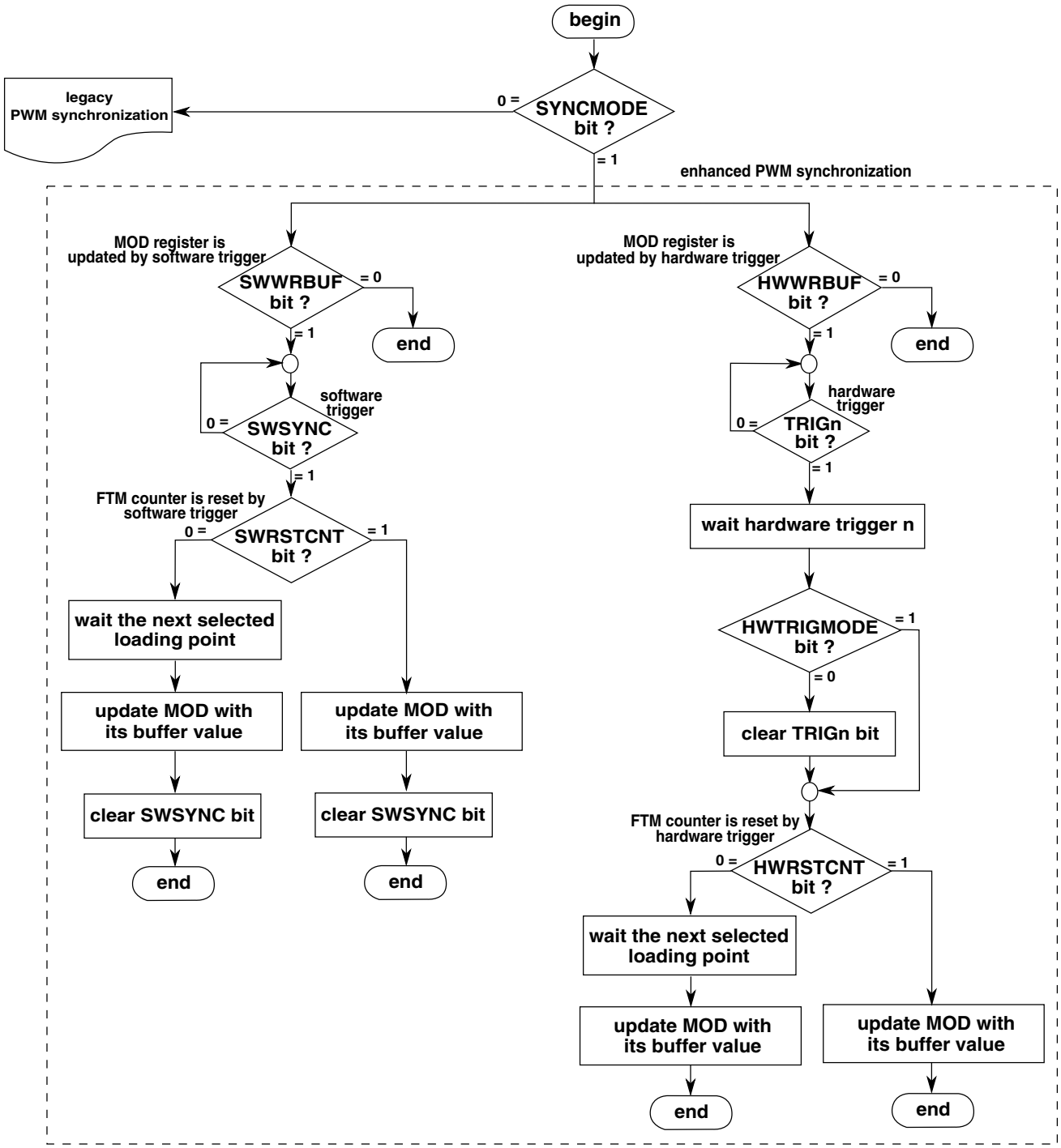



**Figure 40-249. MOD register synchronization flowchart**

In the case of legacy PWM synchronization, the MOD register synchronization depends on PWMSYNC and REINIT bits according to the following description.

If (SYNCMODE = 0), (PWMSYNC = 0), and (REINIT = 0), then this synchronization is made on the next selected loading point after an enabled trigger event takes place. If the trigger event was a software trigger, then the SWSYNC bit is cleared on the next selected

loading point. If the trigger event was a hardware trigger, then the trigger enable bit (TRIGn) is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

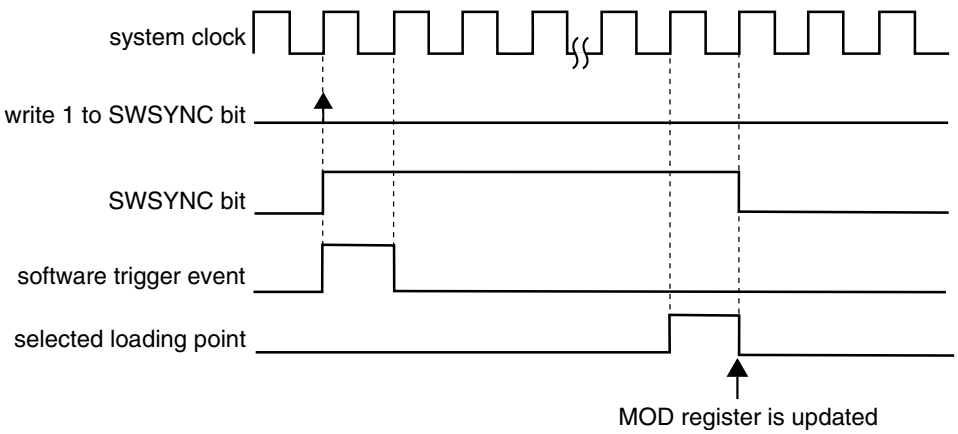



**Figure 40-250. MOD synchronization with (SYNCMODE = 0), (PWMSYNC = 0), (REINIT = 0), and software trigger was used**

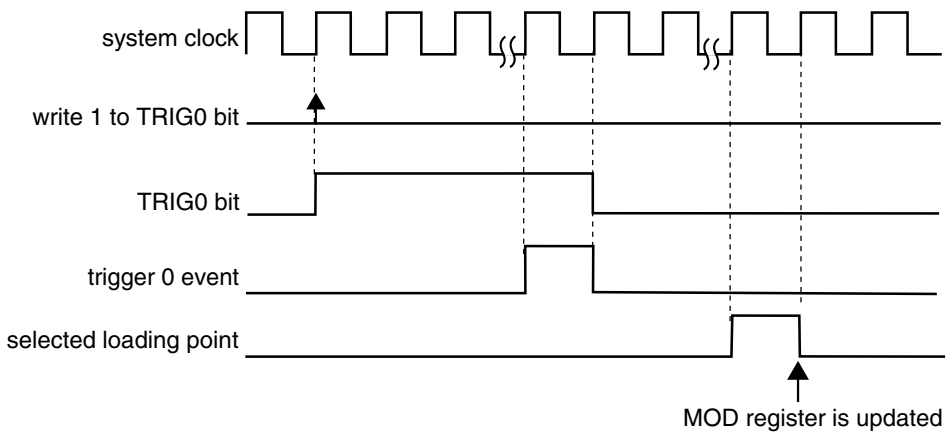



**Figure 40-251. MOD synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (PWMSYNC = 0), (REINIT = 0), and a hardware trigger was used**

If (SYNCMODE = 0), (PWMSYNC = 0), and (REINIT = 1), then this synchronization is made on the next enabled trigger event. If the trigger event was a software trigger, then the SWSYNC bit is cleared according to the following example. If the trigger event was a hardware trigger, then the TRIGn bit is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

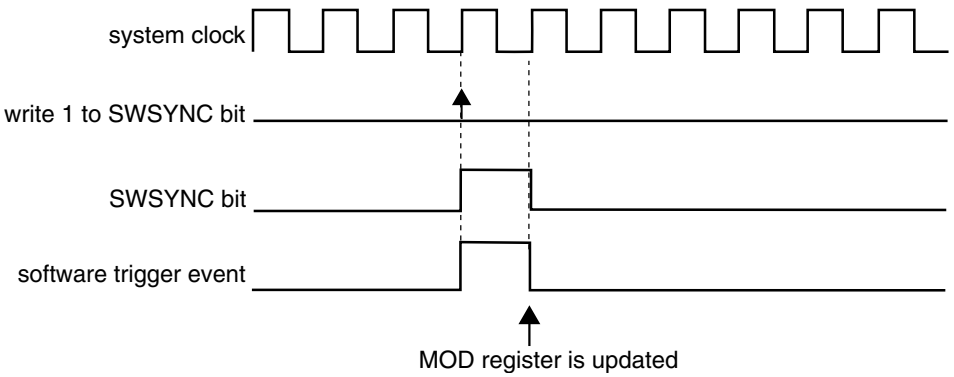



**Figure 40-252. MOD synchronization with (SYNCMODE = 0), (PWMSYNC = 0), (REINIT = 1), and software trigger was used**

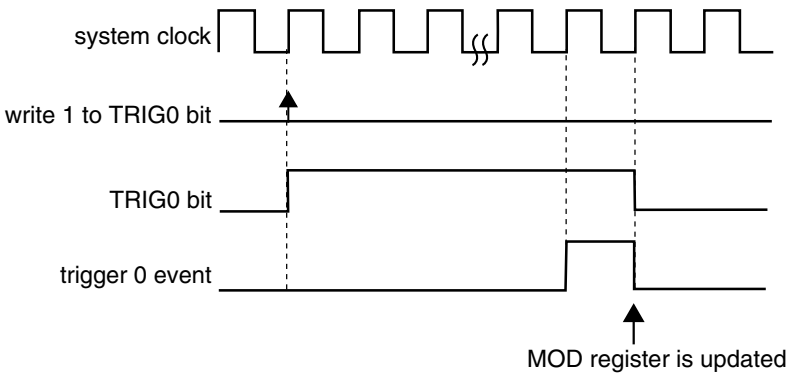



**Figure 40-253. MOD synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (PWMSYNC = 0), (REINIT = 1), and a hardware trigger was used**

If (SYNCMODE = 0) and (PWMSYNC = 1), then this synchronization is made on the next selected loading point after the software trigger event takes place. The SWSYNC bit is cleared on the next selected loading point:

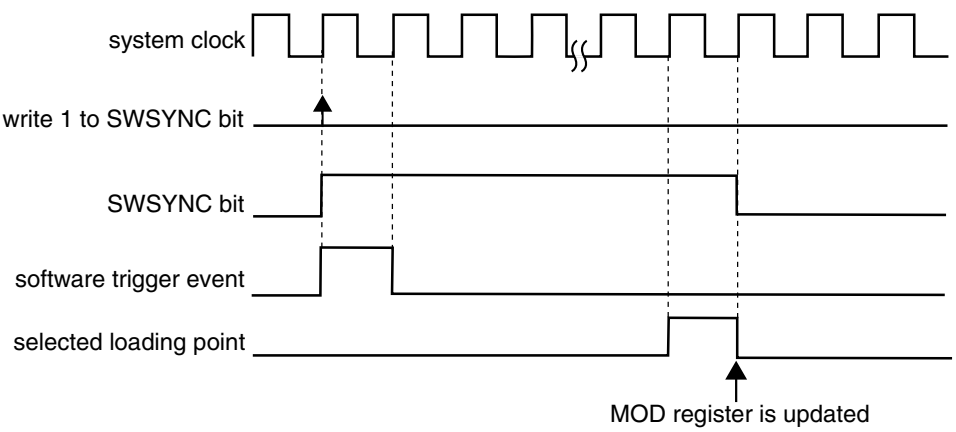



**Figure 40-254. MOD synchronization with (SYNCMODE = 0) and (PWMSYNC = 1)**

### 40.4.11.5 CNTIN register synchronization

The CNTIN register synchronization updates the CNTIN register with its buffer value.

This synchronization is enabled if (FTMEN = 1), (SYNCMODE = 1), and (CNTINC = 1). The CNTIN register synchronization can be done only by the enhanced PWM synchronization (SYNCMODE = 1). The synchronization mechanism is the same as the MOD register synchronization done by the enhanced PWM synchronization; see MOD register synchronization.

### 40.4.11.6 C(n)V and C(n+1)V register synchronization

The C(n)V and C(n+1)V registers synchronization updates the C(n)V and C(n+1)V registers with their buffer values.

This synchronization is enabled if (FTMEN = 1) and (SYNCEN = 1). The synchronization mechanism is the same as the MOD register synchronization. However, it is expected that the C(n)V and C(n+1)V registers be synchronized only by the enhanced PWM synchronization (SYNCMODE = 1).

### 40.4.11.7 OUTMASK register synchronization

The OUTMASK register synchronization updates the OUTMASK register with its buffer value.

The OUTMASK register can be updated at each rising edge of system clock (SYNCHOM = 0), by the enhanced PWM synchronization (SYNCHOM = 1 and SYNCMODE = 1) or by the legacy PWM synchronization (SYNCHOM = 1 and SYNCMODE = 0). However, it is expected that the OUTMASK register be synchronized only by the enhanced PWM synchronization.

In the case of enhanced PWM synchronization, the OUTMASK register synchronization depends on SWOM and HWOM bits. See the following flowchart:

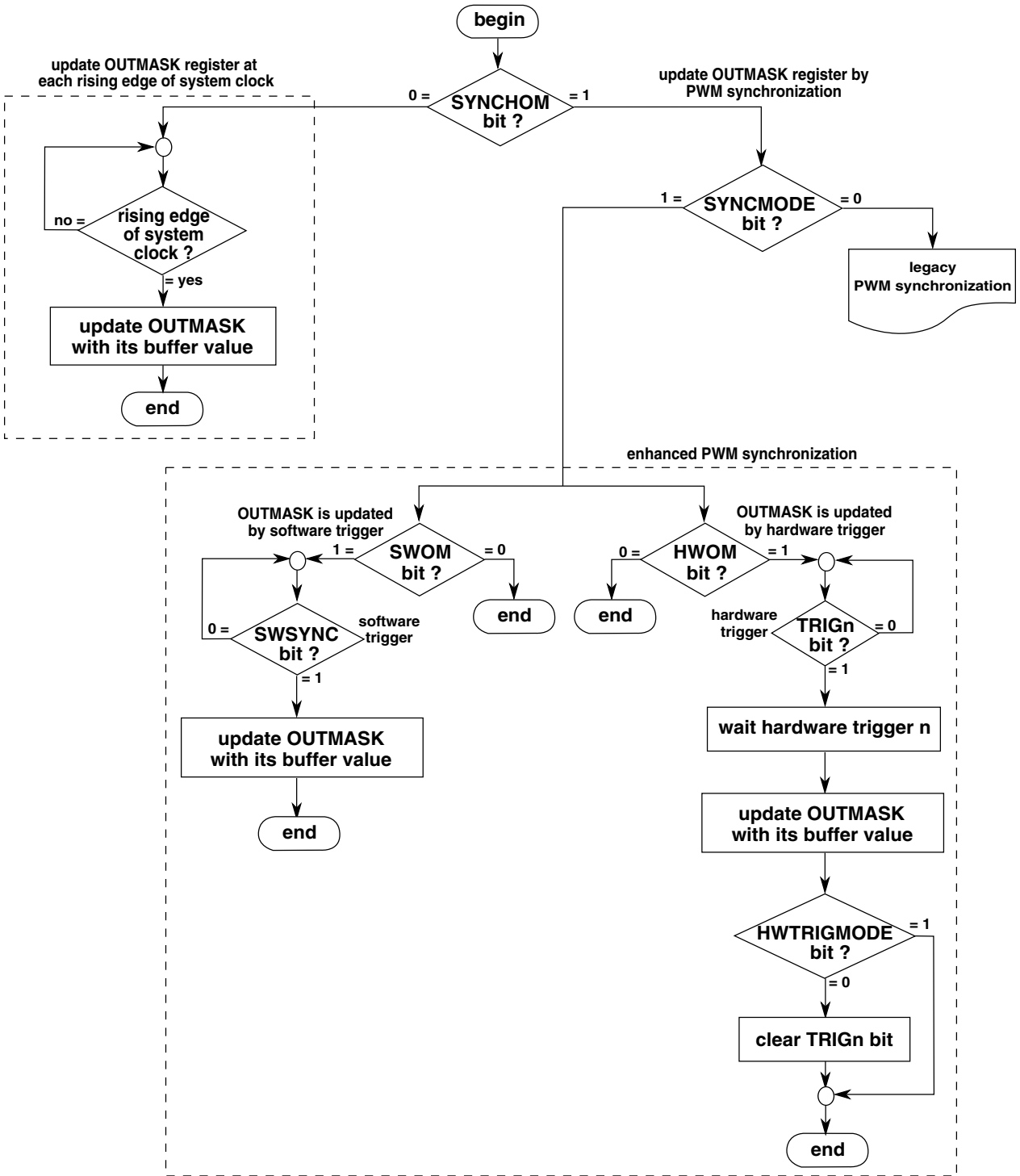



**Figure 40-255. OUTMASK register synchronization flowchart**

In the case of legacy PWM synchronization, the OUTMASK register synchronization depends on PWMSYNC bit according to the following description.

If (SYNCMODE = 0), (SYNCHOM = 1), and (PWMSYNC = 0), then this synchronization is done on the next enabled trigger event. If the trigger event was a software trigger, then the SWSYNC bit is cleared on the next selected loading point. If the trigger event was a hardware trigger, then the TRIGn bit is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

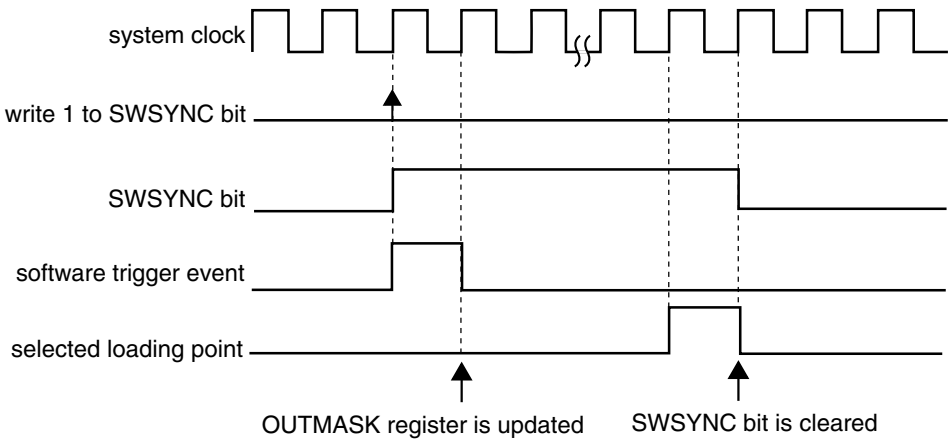



**Figure 40-256. OUTMASK synchronization with (SYNCMODE = 0), (SYNCHOM = 1), (PWMSYNC = 0) and software trigger was used**

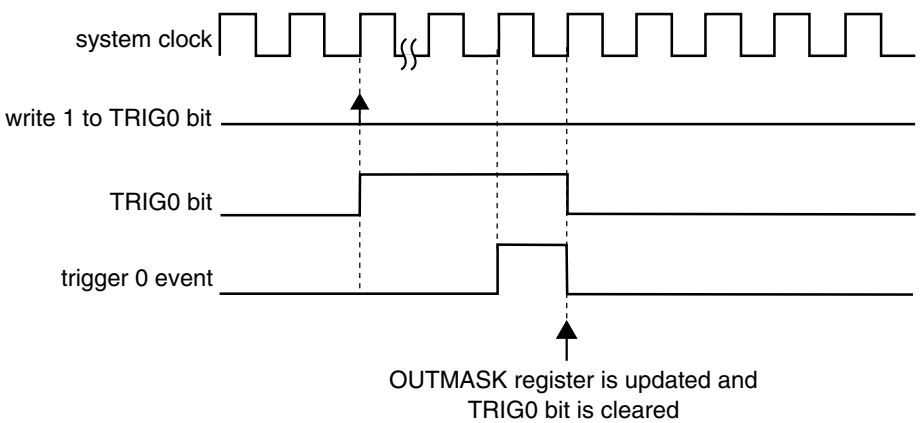



**Figure 40-257. OUTMASK synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (SYNCHOM = 1), (PWMSYNC = 0), and a hardware trigger was used**

If (SYNCMODE = 0), (SYNCHOM = 1), and (PWMSYNC = 1), then this synchronization is made on the next enabled hardware trigger. The TRIGn bit is cleared according to Hardware trigger. An example with a hardware trigger follows.

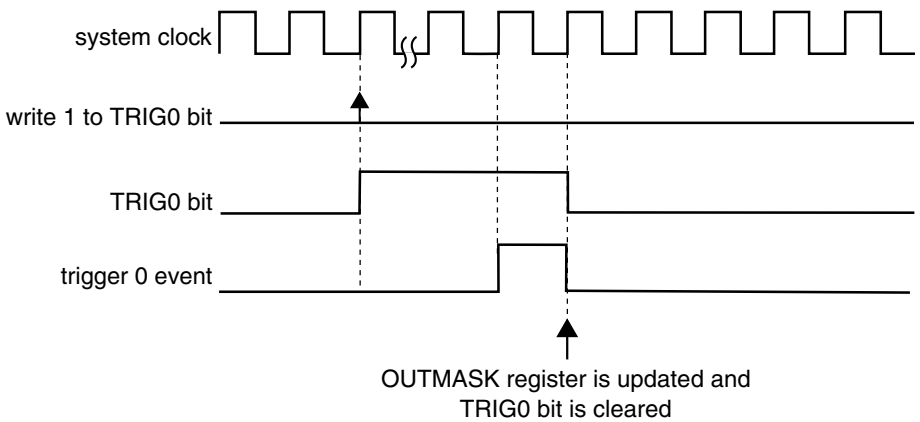



**Figure 40-258. OUTMASK synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (SYNCHOM = 1), (PWMSYNC = 1), and a hardware trigger was used**

### 40.4.11.8 INVCTRL register synchronization

The INVCTRL register synchronization updates the INVCTRL register with its buffer value.

The INVCTRL register can be updated at each rising edge of system clock (INVC = 0) or by the enhanced PWM synchronization (INVC = 1 and SYNCMODE = 1) according to the following flowchart.

In the case of enhanced PWM synchronization, the INVCTRL register synchronization depends on SWINVC and HWINVC bits.

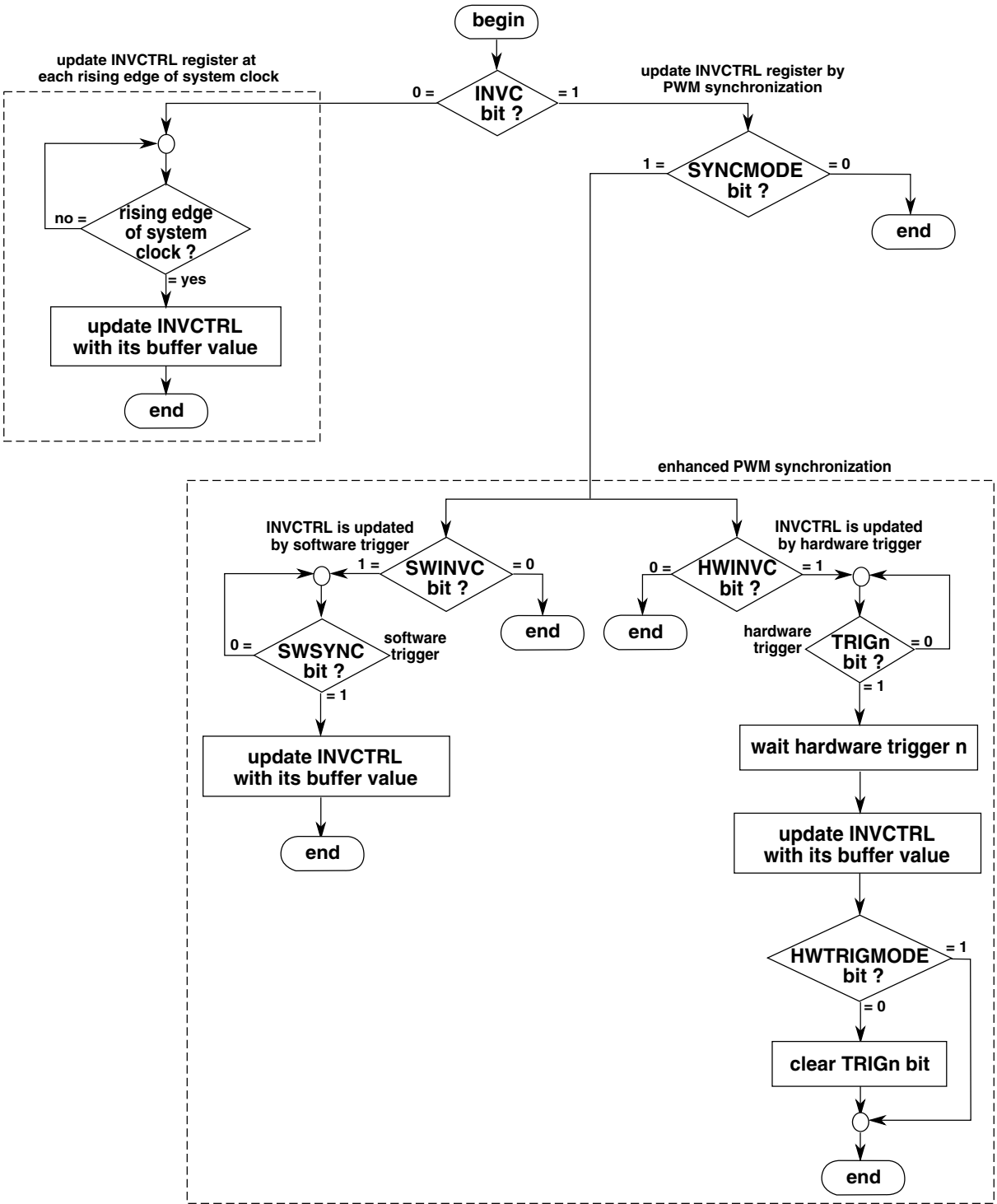



**Figure 40-259. INVCTRL register synchronization flowchart**

## 40.4.11.9 SWOCTRL register synchronization

The SWOCTRL register synchronization updates the SWOCTRL register with its buffer value.

The SWOCTRL register can be updated at each rising edge of system clock (SWOC = 0) or by the enhanced PWM synchronization (SWOC = 1 and SYNCMODE = 1) according to the following flowchart.

In the case of enhanced PWM synchronization, the SWOCTRL register synchronization depends on SWSOC and HWSOC bits.

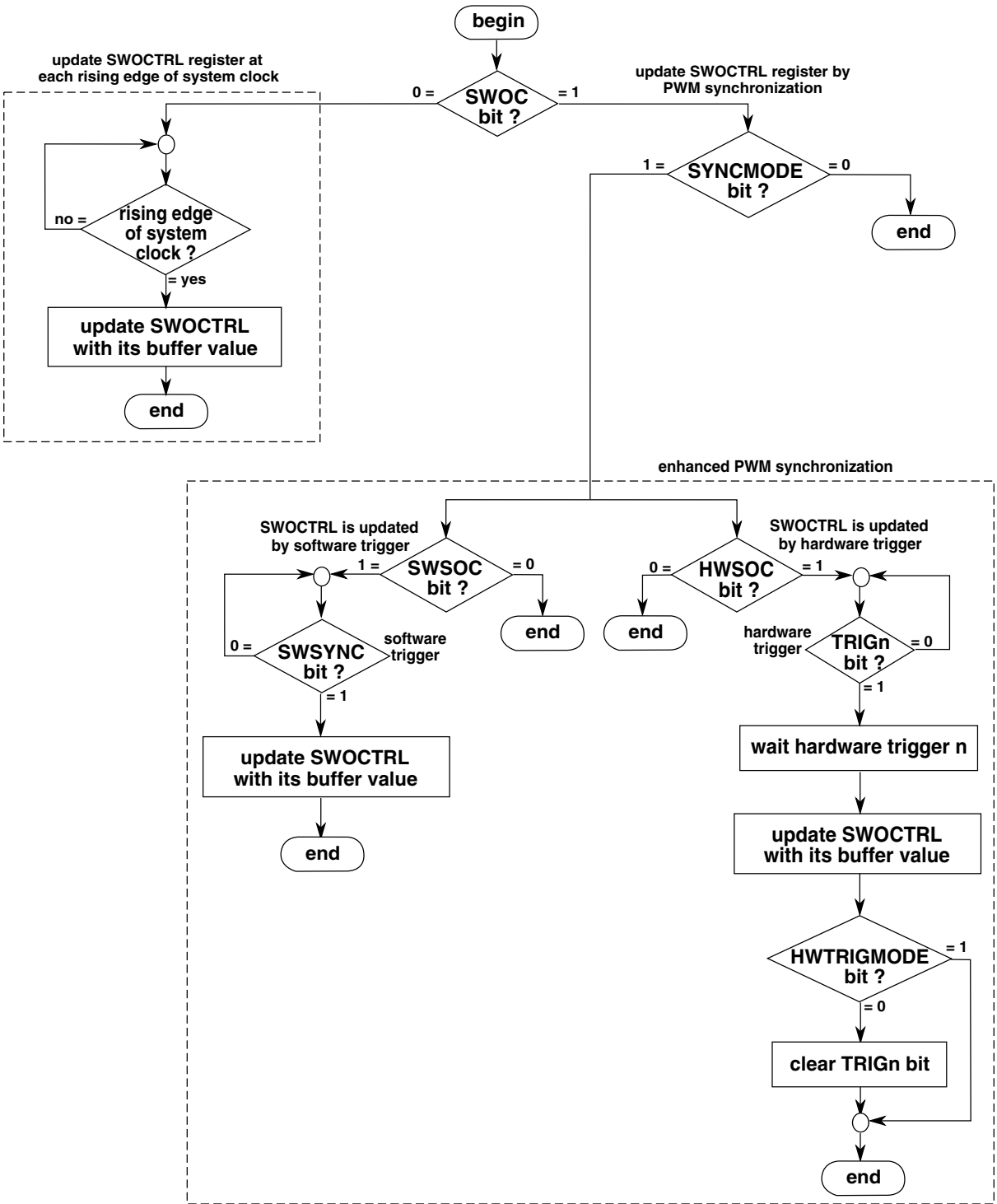



**Figure 40-260. SWOCTRL register synchronization flowchart**

### 40.4.11.10 FTM counter synchronization

The FTM counter synchronization is a mechanism that allows the FTM to restart the PWM generation at a certain point in the PWM period. The channels outputs are forced to their initial value, except for channels in Output Compare mode, and the FTM counter is forced to its initial counting value defined by CNTIN register.

The following figure shows the FTM counter synchronization. Note that after the synchronization event occurs, the channel (n) is set to its initial value and the channel (n +1) is not set to its initial value due to a specific timing of this figure in which the deadtime insertion prevents this channel output from transitioning to 1. If no deadtime insertion is selected, then the channel (n+1) transitions to logical value 1 immediately after the synchronization event occurs.

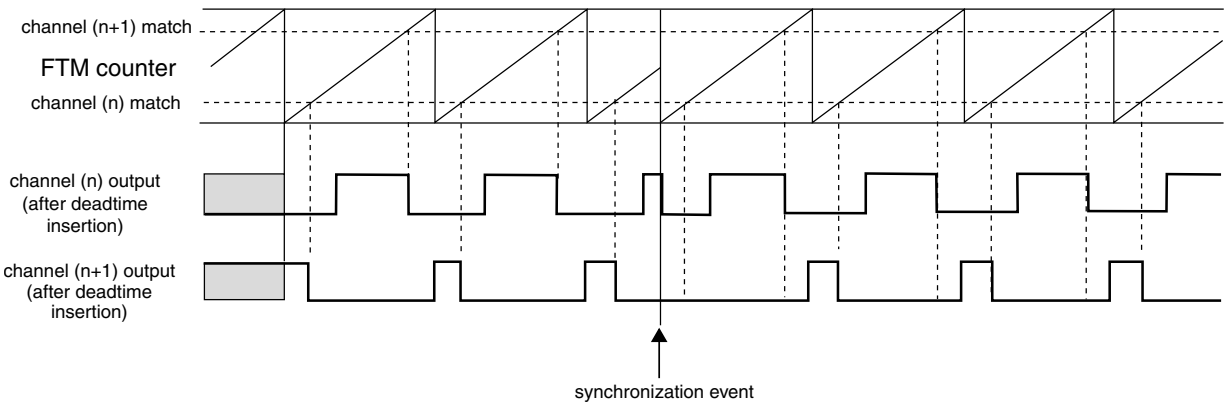



**Figure 40-261. FTM counter synchronization**

The FTM counter synchronization can be done by either the enhanced PWM synchronization (SYNCMODE = 1) or the legacy PWM synchronization (SYNCMODE = 0). However, the FTM counter must be synchronized only by the enhanced PWM synchronization.

In the case of enhanced PWM synchronization, the FTM counter synchronization depends on SWRSTCNT and HWRSTCNT bits according to the following flowchart.

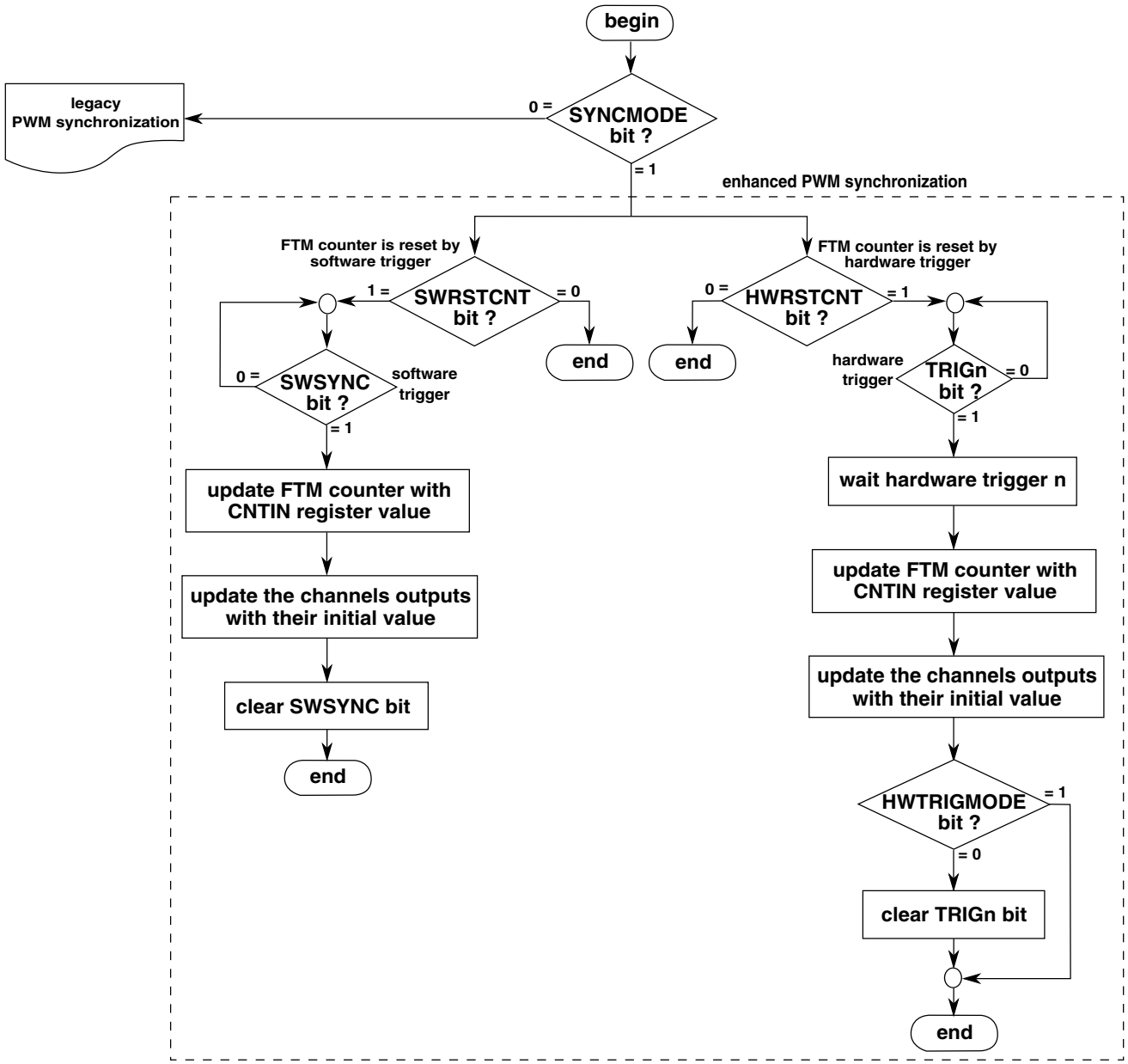



**Figure 40-262. FTM counter synchronization flowchart**

In the case of legacy PWM synchronization, the FTM counter synchronization depends on REINIT and PWMSYNC bits according to the following description.

If (SYNCMODE = 0), (REINIT = 1), and (PWMSYNC = 0) then this synchronization is made on the next enabled trigger event. If the trigger event was a software trigger then the SWSYNC bit is cleared according to the following example. If the trigger event was a hardware trigger then the TRIGn bit is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

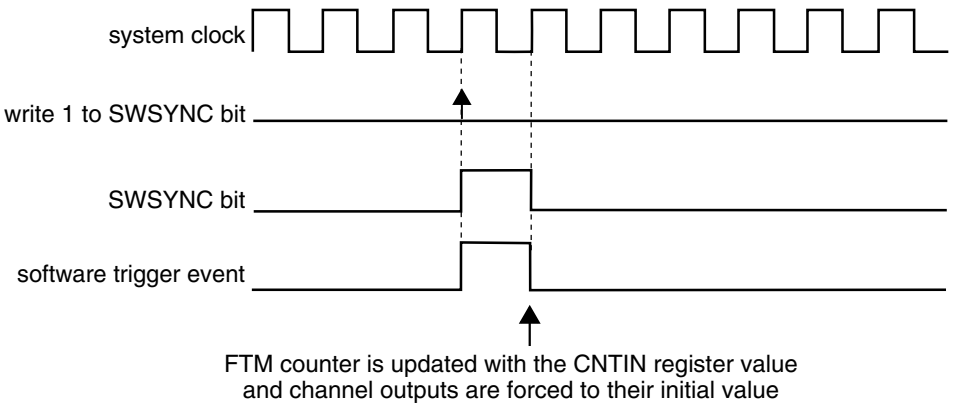



**Figure 40-263. FTM counter synchronization with (SYNCMODE = 0), (REINIT = 1), (PWMSYNC = 0), and software trigger was used**

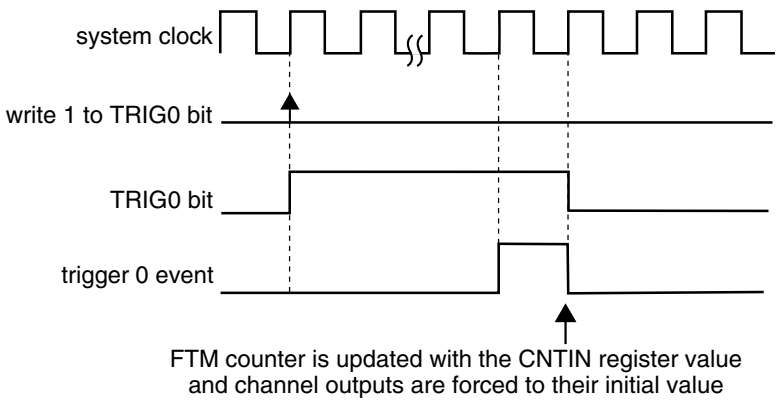



**Figure 40-264. FTM counter synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (REINIT = 1), (PWMSYNC = 0), and a hardware trigger was used**

If (SYNCMODE = 0), (REINIT = 1), and (PWMSYNC = 1) then this synchronization is made on the next enabled hardware trigger. The TRIGn bit is cleared according to Hardware trigger.

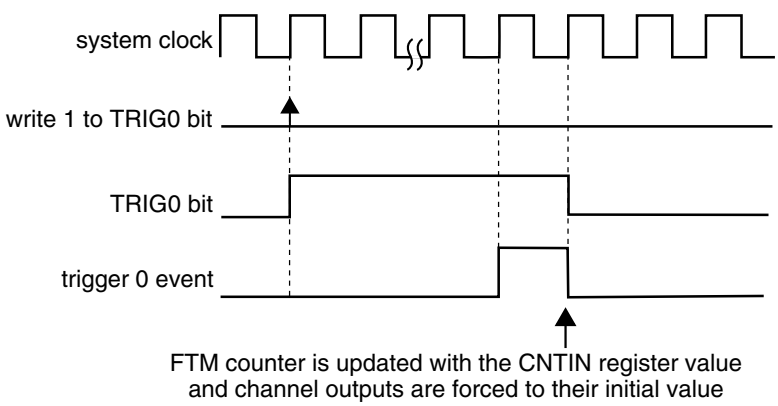



**Figure 40-265. FTM counter synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (REINIT = 1), (PWMSYNC = 1), and a hardware trigger was used**

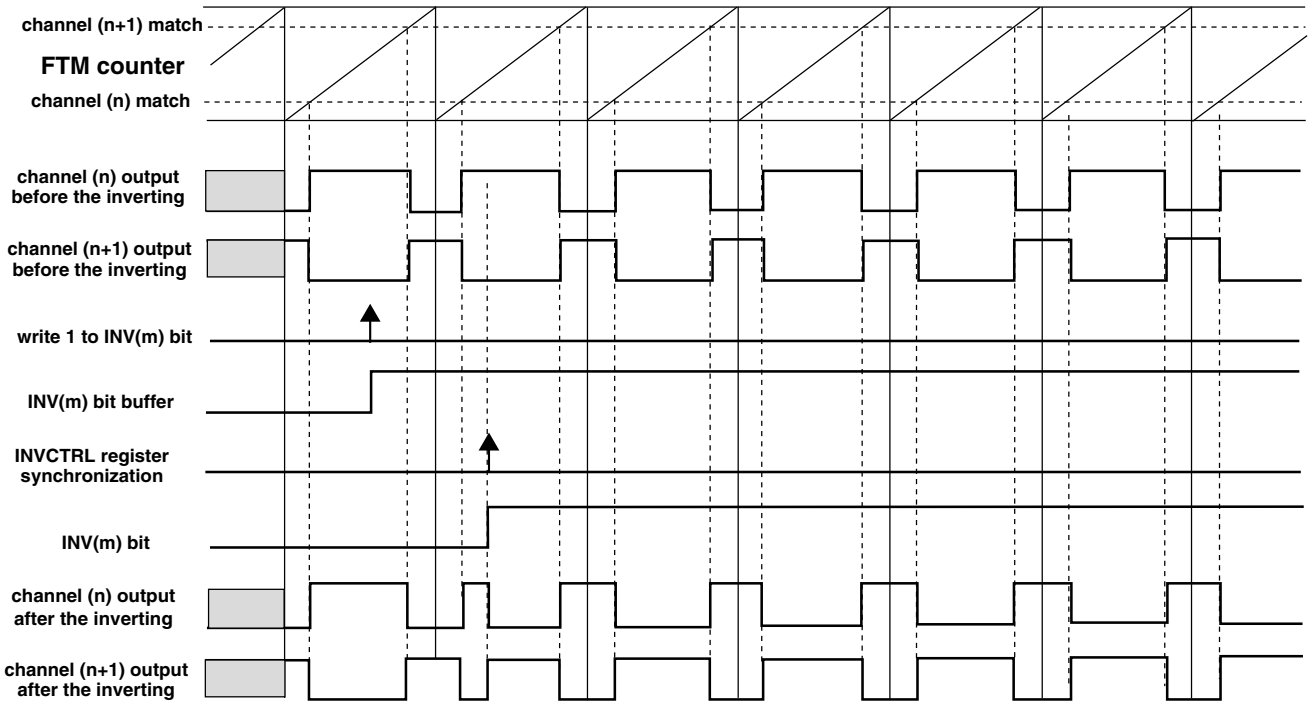
### 40.4.12 Inverting

The invert functionality swaps the signals between channel (n) and channel (n+1) outputs. The inverting operation is selected when:

- FTMEN = 1
- QUADEN = 0
- DECAPEN = 0
- COMBINE = 1
- COMP = 1
- CPWMS = 0, and
- INVm = 1 (where m represents a channel pair)

The INVm bit in INVCTRL register is updated with its buffer value according to INVCTRL register synchronization

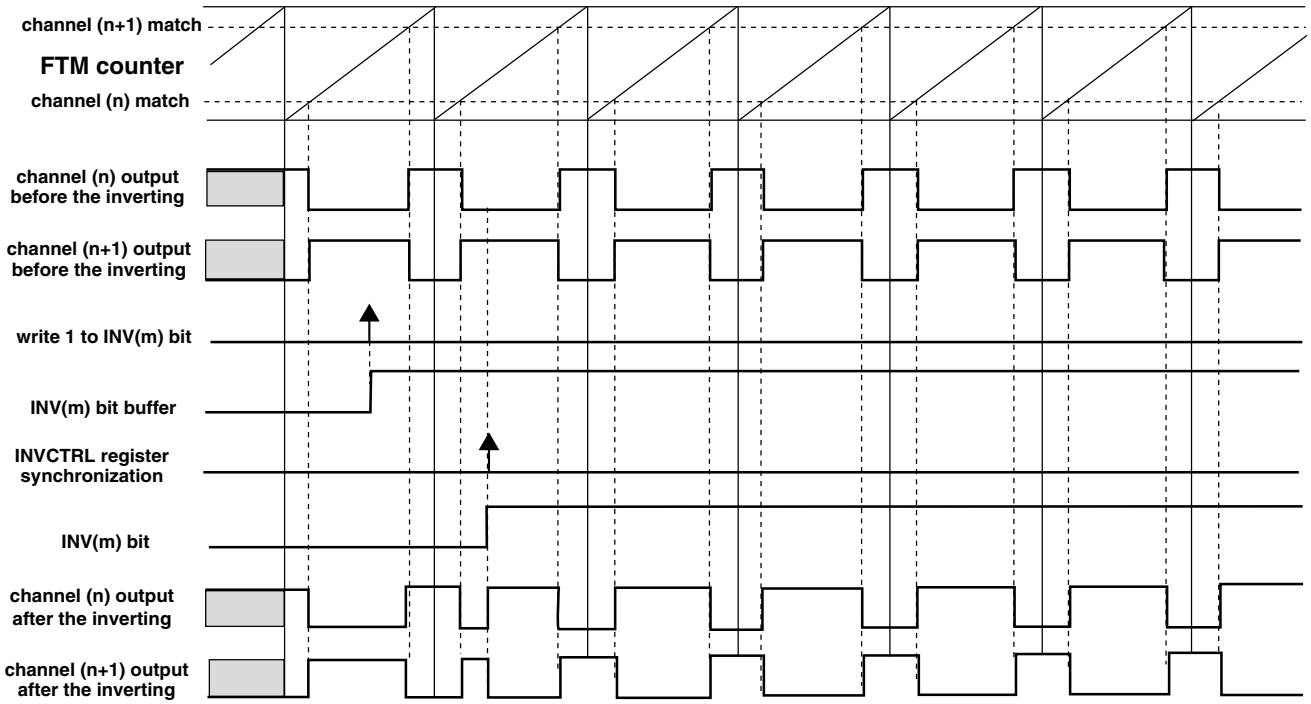
In High-True (ELSnB:ELSnA = 1:0) Combine mode, the channel (n) output is forced low at the beginning of the period (FTM counter = CNTIN), forced high at the channel (n) match and forced low at the channel (n+1) match. If the inverting is selected, the channel (n) output behavior is changed to force high at the beginning of the PWM period, force low at the channel (n) match and force high at the channel (n+1) match. See the following figure.




NOTE
INV(m) bit selects the inverting to the pair channels (n) and (n+1).

**Figure 40-266. Channels (n) and (n+1) outputs after the inverting in High-True (ELSnB:ELSnA = 1:0) Combine mode**

Note that the ELSnB:ELSnA bits value should be considered because they define the active state of the channels outputs. In Low-True (ELSnB:ELSnA = X:1) Combine mode, the channel (n) output is forced high at the beginning of the period, forced low at the channel (n) match and forced high at the channel (n+1) match. When inverting is selected, the channels (n) and (n+1) present waveforms as shown in the following figure.




NOTE
INV(m) bit selects the inverting to the pair channels (n) and (n+1).

**Figure 40-267. Channels (n) and (n+1) outputs after the inverting in Low-True (ELSnB:ELSnA = X:1) Combine mode**

### Note

The inverting feature must be used only in Combine mode.

## 40.4.13 Software output control

The software output control forces the channel output according to software defined values at a specific time in the PWM generation.

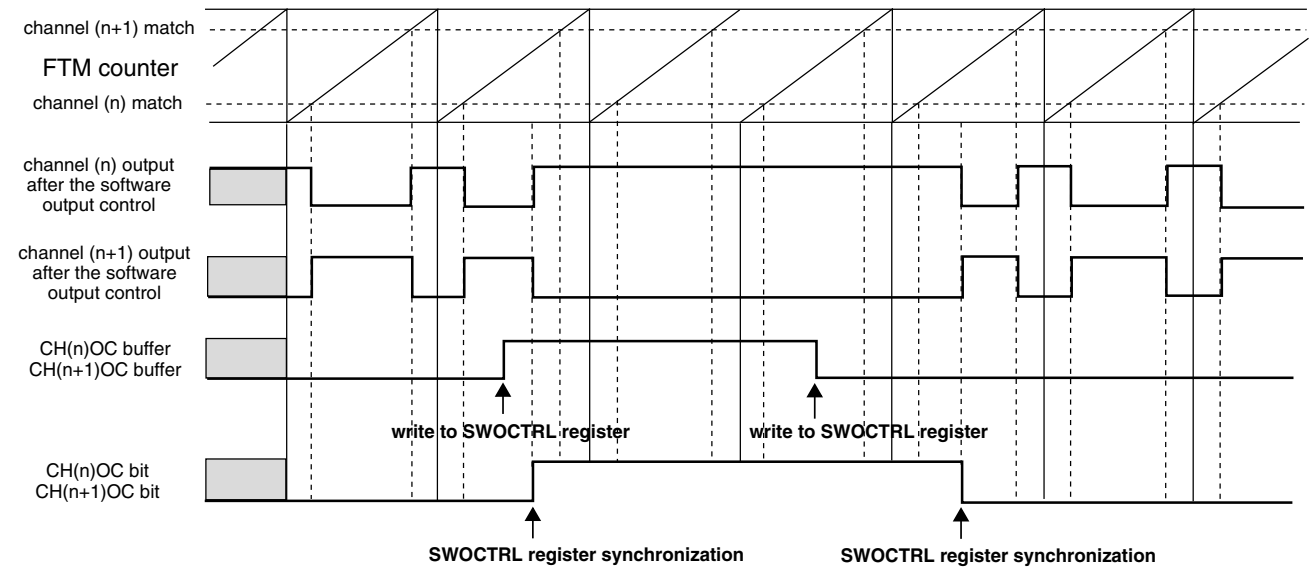
The software output control is selected when:

- FTMEN = 1
- QUADEN = 0
- DECAPEN = 0
- COMBINE = 1
- CPWMS = 0, and
- CHnOC = 1

The CHnOC bit enables the software output control for a specific channel output and the CHnOCV selects the value that is forced to this channel output.

Both CHnOC and CHnOCV bits in SWOCTRL register are buffered and updated with their buffer value according to SWOCTRL register synchronization.

The following figure shows the channels (n) and (n+1) outputs signals when the software output control is used. In this case the channels (n) and (n+1) are set to Combine and Complementary mode.




NOTE
**CH(n)OCV = 1 and CH(n+1)OCV = 0.**

**Figure 40-268. Example of software output control in Combine and Complementary mode**

Software output control forces the following values on channels (n) and (n+1) when the COMP bit is zero.

**Table 40-306. Software ouput control behavior when (COMP = 0)**

| CH(n)OC | CH(n+1)OC | CH(n)OCV | CH(n+1)OCV | Channel (n) Output | Channel (n+1) Output |
|---|---|---|---|---|---|
| 0 | 0 | X | X | is not modified by SWOC | is not modified by SWOC |
| 1 | 1 | 0 | 0 | is forced to zero | is forced to zero |
| 1 | 1 | 0 | 1 | is forced to zero | is forced to one |
| 1 | 1 | 1 | 0 | is forced to one | is forced to zero |
| 1 | 1 | 1 | 1 | is forced to one | is forced to one |

Software output control forces the following values on channels (n) and (n+1) when the COMP bit is one.

**Table 40-307. Software ouput control behavior when (COMP = 1)**

| CH(n)OC | CH(n+1)OC | CH(n)OCV | CH(n+1)OCV | Channel (n) Output | Channel (n+1) Output |
|---|---|---|---|---|---|
| 0 | 0 | X | X | is not modified by SWOC | is not modified by SWOC |
| 1 | 1 | 0 | 0 | is forced to zero | is forced to zero |
| 1 | 1 | 0 | 1 | is forced to zero | is forced to one |
| 1 | 1 | 1 | 0 | is forced to one | is forced to zero |
| 1 | 1 | 1 | 1 | is forced to one | is forced to zero |

**Note**

- The software output control feature must be used only in Combine mode.
- The CH(n)OC and CH(n+1)OC bits should be equal.
- The COMP bit must not be modified when software output control is enabled, that is, CH(n)OC = 1 and/or CH(n +1)OC = 1.
- Software output control has the same behavior with disabled or enabled FTM counter (see the CLKS field description in the Status and Control register).

## 40.4.14 Deadtime insertion

The deadtime insertion is enabled when (DTEN = 1) and (DTVAL[5:0] is non- zero).

DEADTIME register defines the deadtime delay that can be used for all FTM channels. The DTPS[1:0] bits define the prescaler for the system clock and the DTVAL[5:0] bits define the deadtime modulo, that is, the number of the deadtime prescaler clocks.

The deadtime delay insertion ensures that no two complementary signals (channels (n) and (n+1)) drive the active state at the same time.

If POL(n) = 0, POL(n+1) = 0, and the deadtime is enabled, then when the channel (n) match (FTM counter = C(n)V) occurs, the channel (n) output remains at the low value until the end of the deadtime delay when the channel (n) output is set. Similarly, when the channel (n+1) match (FTM counter = C(n+1)V) occurs, the channel (n+1) output remains at the low value until the end of the deadtime delay when the channel (n+1) output is set. See the following figures.

If POL(n) = 1, POL(n+1) = 1, and the deadtime is enabled, then when the channel (n) match (FTM counter = C(n)V) occurs, the channel (n) output remains at the high value until the end of the deadtime delay when the channel (n) output is cleared. Similarly, when the channel (n+1) match (FTM counter = C(n+1)V) occurs, the channel (n+1) output remains at the high value until the end of the deadtime delay when the channel (n +1) output is cleared.

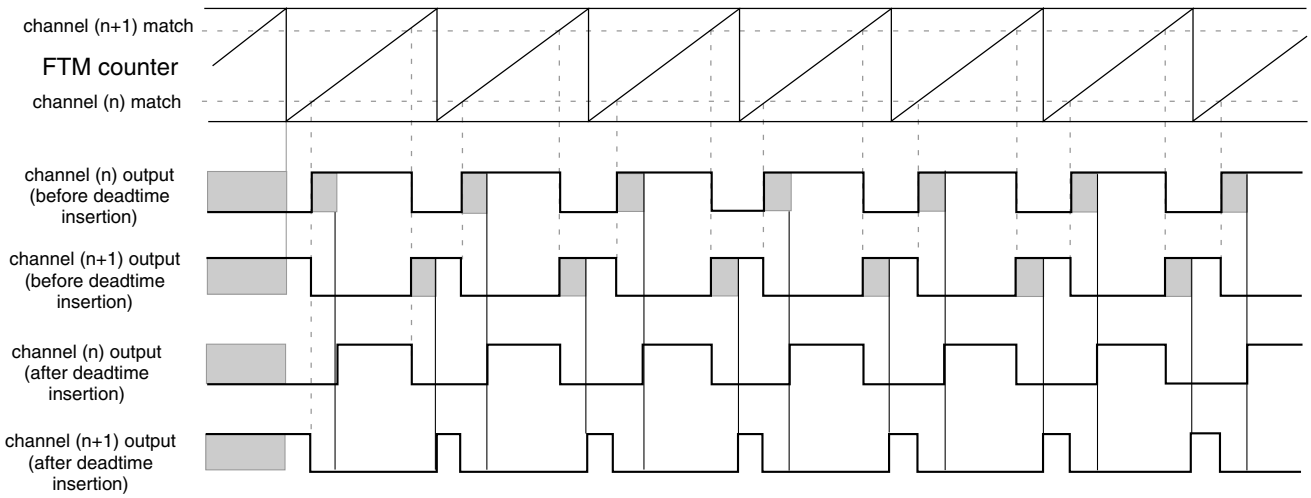



**Figure 40-269. Deadtime insertion with ELSnB:ELSnA = 1:0, POL(n) = 0, and POL(n+1) = 0**

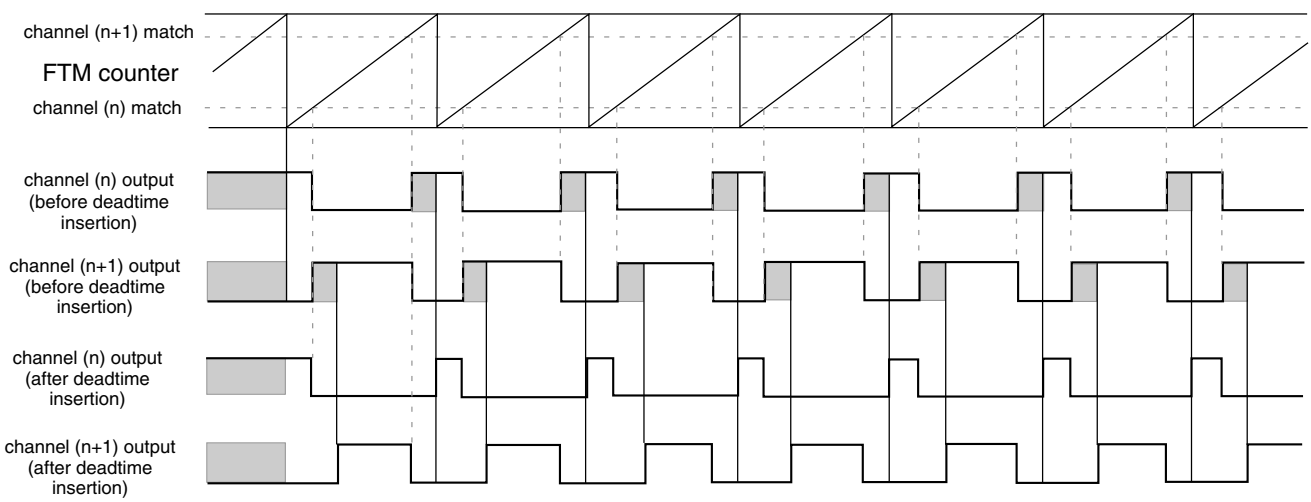



**Figure 40-270. Deadtime insertion with ELSnB:ELSnA = X:1, POL(n) = 0, and POL(n+1) = 0**

## NOTE

The deadtime feature must be used only in Combine and Complementary modes.

### 40.4.14.1 Deadtime insertion corner cases

If (PS[2:0] is cleared), (DTPS[1:0] = 0:0 or DTPS[1:0] = 0:1):

- and the deadtime delay is greater than or equal to the channel (n) duty cycle ((C(n +1)V – C(n)V) × system clock), then the channel (n) output is always the inactive value (POL(n) bit value).
- and the deadtime delay is greater than or equal to the channel (n+1) duty cycle ((MOD – CNTIN + 1 – (C(n+1)V – C(n)V) ) × system clock), then the channel (n+1) output is always the inactive value (POL(n+1) bit value).

Although, in most cases the deadtime delay is not comparable to channels (n) and (n+1) duty cycle, the following figures show examples where the deadtime delay is comparable to the duty cycle.

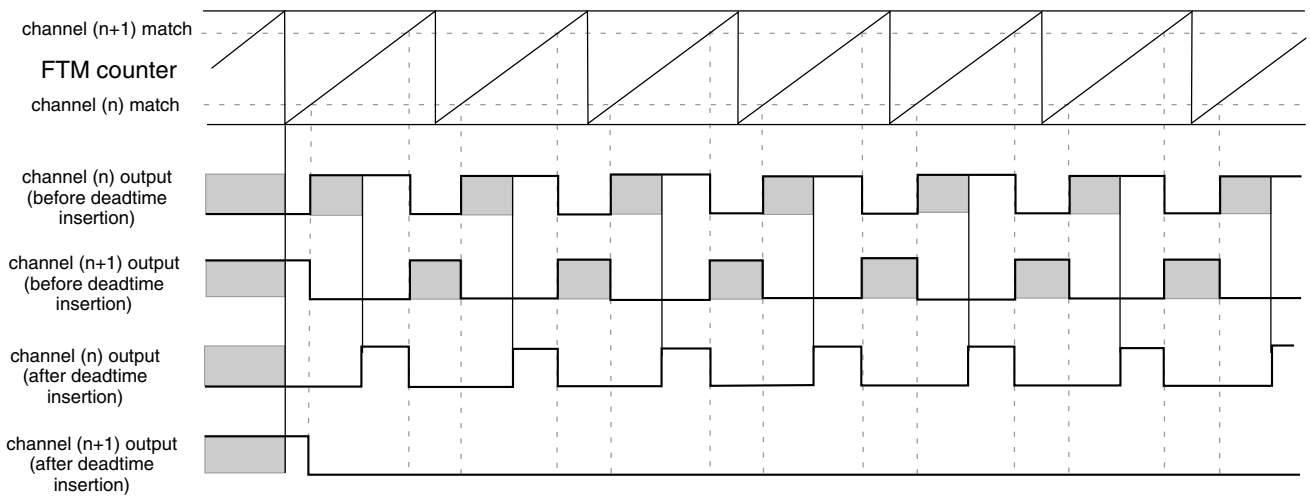



**Figure 40-271. Example of the deadtime insertion (ELSnB:ELSnA = 1:0, POL(n) = 0, and POL(n+1) = 0) when the deadtime delay is comparable to channel (n+1) duty cycle**

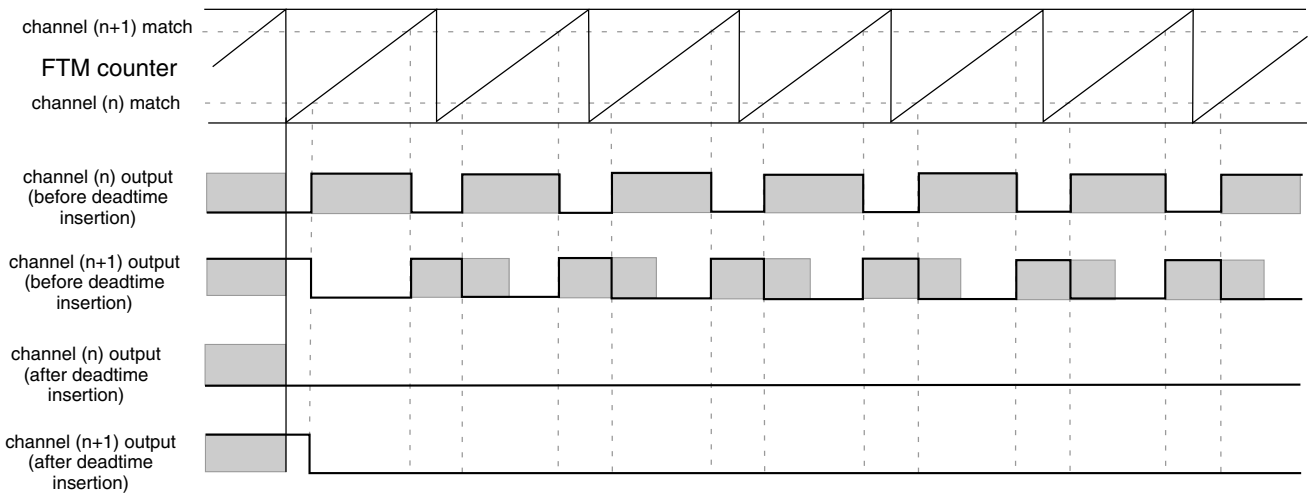



**Figure 40-272. Example of the deadtime insertion (ELSnB:ELSnA = 1:0, POL(n) = 0, and POL(n+1) = 0) when the deadtime delay is comparable to channels (n) and (n+1) duty cycle**

## 40.4.15 Output mask

The output mask can be used to force channels output to their inactive state through software. For example: to control a BLDC motor.

Any write to the OUTMASK register updates its write buffer. The OUTMASK register is updated with its buffer value by PWM synchronization; see OUTMASK register synchronization.

If CHnOM = 1, then the channel (n) output is forced to its inactive state (POLn bit value). If CHnOM = 0, then the channel (n) output is unaffected by the output mask. See the following figure.

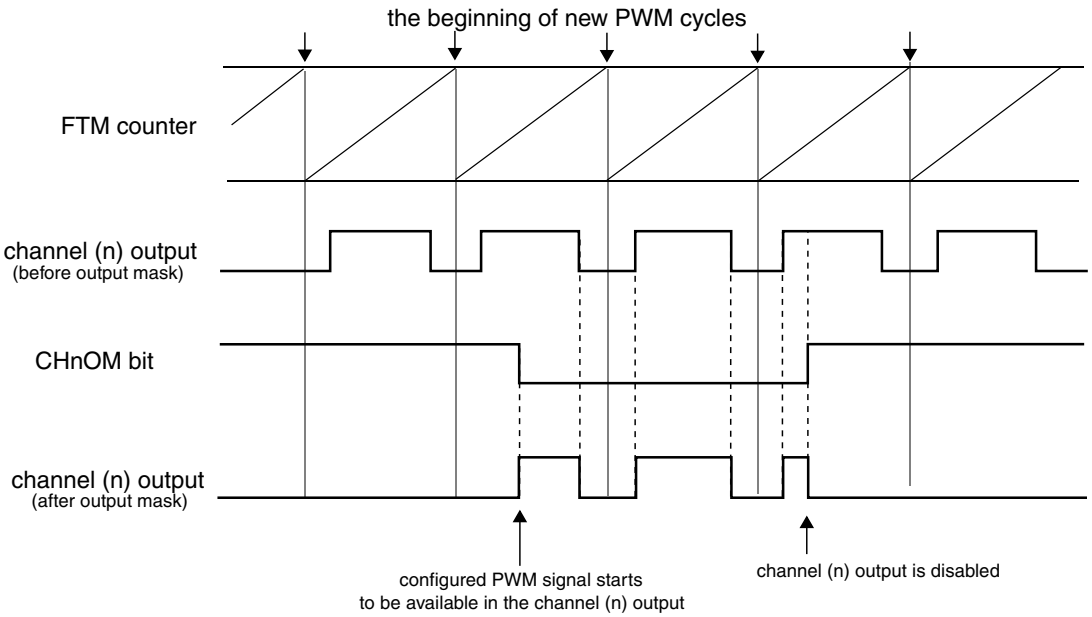



**Figure 40-273. Output mask with POLn = 0**

The following table shows the output mask result before the polarity control.

**Table 40-308. Output mask result for channel (n) before the polarity control**

| CHnOM | Output Mask Input | Output Mask Result |
|---|---|---|
| 0 | inactive state | inactive state |
| | active state | active state |
| 1 | inactive state | inactive state |
| | active state | |

### Note

The output mask feature must be used only in Combine mode.

## 40.4.16 Fault control

The fault control is enabled if (FTMEN = 1) and (FAULTM[1:0] ≠ 0:0).

FTM can have up to four fault inputs. FAULTnEN bit (where n = 0, 1, 2, 3) enables the fault input n and FFLTRnEN bit enables the fault input n filter. FFVAL[3:0] bits select the value of the enabled filter in each enabled fault input.

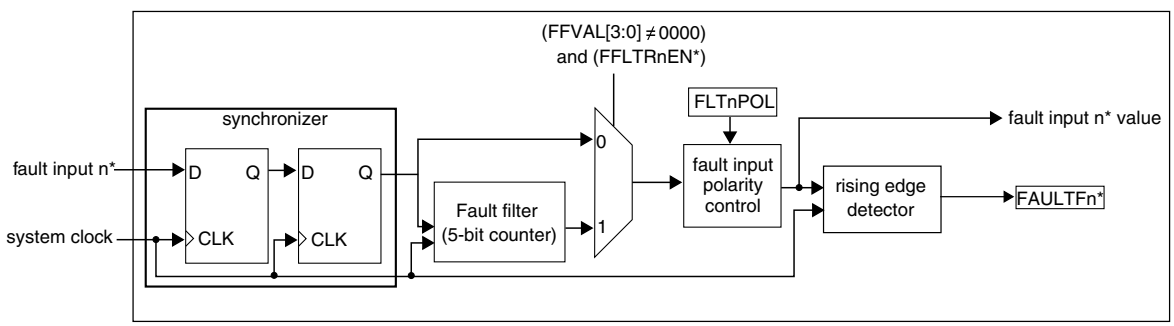
First, each fault input signal is synchronized by the system clock; see the synchronizer block in the following figure. Following synchronization, the fault input n signal enters the filter block. When there is a state change in the fault input n signal, the 5-bit counter

is reset and starts counting up. As long as the new state is stable on the fault input n, the counter continues to increment. If the 5-bit counter overflows, that is, the counter exceeds the value of the FFVAL[3:0] bits, the new fault input n value is validated. It is then transmitted as a pulse edge to the edge detector.

If the opposite edge appears on the fault input n signal before validation (counter overflow), the counter is reset. At the next input transition, the counter starts counting again. Any pulse that is shorter than the minimum value selected by FFVAL[3:0] bits (× system clock) is regarded as a glitch and is not passed on to the edge detector.

The fault input n filter is disabled when the FFVAL[3:0] bits are zero or when FAULTnEN = 0. In this case, the fault input n signal is delayed 2 rising edges of the system clock and the FAULTFn bit is set on 3th rising edge of the system clock after a rising edge occurs on the fault input n.

If FFVAL[3:0] ≠ 0000 and FAULTnEN = 1, then the fault input n signal is delayed (3 + FFVAL[3:0]) rising edges of the system clock, that is, the FAULTFn bit is set (4 + FFVAL[3:0]) rising edges of the system clock after a rising edge occurs on the fault input n.




* where n = 3, 2, 1, 0

**Figure 40-274. Fault input n control block diagram**

If the fault control and fault input n are enabled and a rising edge at the fault input n signal is detected, a fault condition has occurred and the FAULTFn bit is set. The FAULTF bit is the logic OR of FAULTFn[3:0] bits. See the following figure.

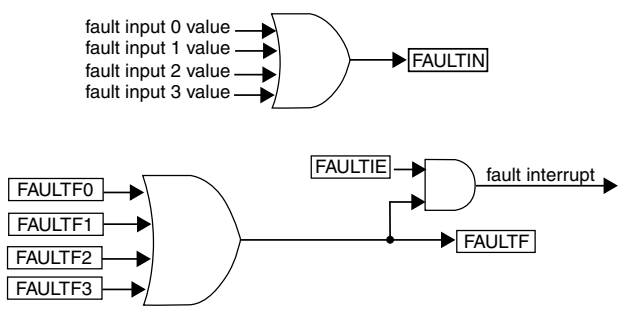



**Figure 40-275. FAULTF and FAULTIN bits and fault interrupt**

If the fault control is enabled (FAULTM[1:0] ≠ 0:0), a fault condition has occurred and (FAULTEN = 1), then outputs are forced to their safe values:

- Channel (n) output takes the value of POL(n)
- Channel (n+1) takes the value of POL(n+1)

The fault interrupt is generated when (FAULTF = 1) and (FAULTIE = 1). This interrupt request remains set until:

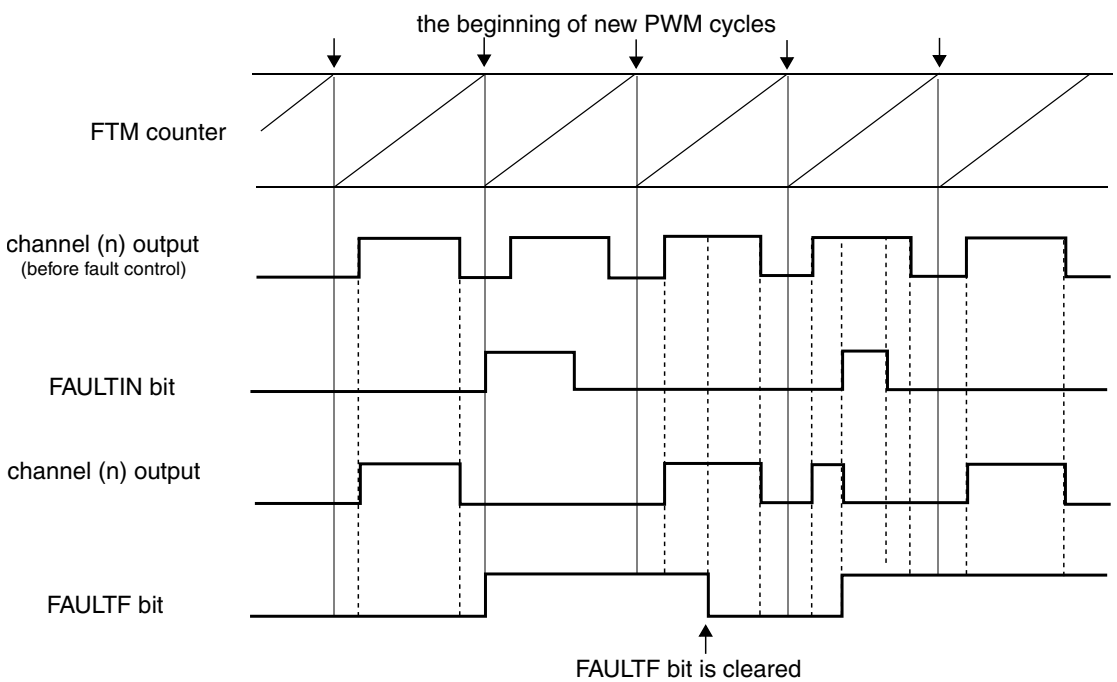
- Software clears the FAULTF bit by reading FAULTF bit as 1 and writing 0 to it
- Software clears the FAULTIE bit
- A reset occurs

**Note**

The fault control must be used only in Combine mode.

### 40.4.16.1 Automatic fault clearing

If the automatic fault clearing is selected (FAULTM[1:0] = 1:1), then the channels output disabled by fault control is again enabled when the fault input signal (FAULTIN) returns to zero and a new PWM cycle begins. See the following figure.

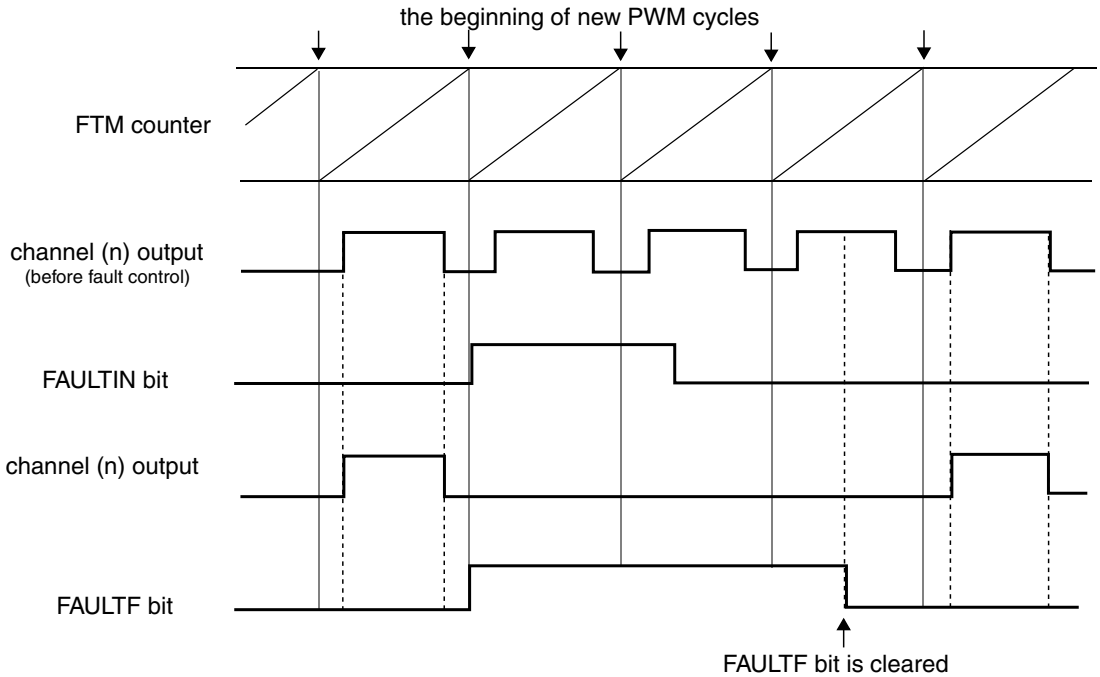



**NOTE**
The channel (n) output is after the fault control with automatic fault clearing and POLn = 0.

**Figure 40-276. Fault control with automatic fault clearing**

### 40.4.16.2 Manual fault clearing

If the manual fault clearing is selected (FAULTM[1:0] = 0:1 or 1:0), then the channels output disabled by fault control is again enabled when the FAULTF bit is cleared and a new PWM cycle begins. See the following figure.




**NOTE**
The channel (n) output is after the fault control with manual fault clearing and POLn = 0.

**Figure 40-277. Fault control with manual fault clearing**

### 40.4.16.3 Fault inputs polarity control

The FLTjPOL bit selects the fault input j polarity, where j = 0, 1, 2, 3:

- If FLTjPOL = 0, the fault j input polarity is high, so the logical one at the fault input j indicates a fault.
- If FLTjPOL = 1, the fault j input polarity is low, so the logical zero at the fault input j indicates a fault.

## 40.4.17 Polarity control

The POLn bit selects the channel (n) output polarity:

- If POLn = 0, the channel (n) output polarity is high, so the logical one is the active state and the logical zero is the inactive state.
- If POLn = 1, the channel (n) output polarity is low, so the logical zero is the active state and the logical one is the inactive state.

**Note**

The polarity control must be used only in Combine mode.

## 40.4.18 Initialization

The initialization forces the CHnOI bit value to the channel (n) output when a one is written to the INIT bit.

The initialization depends on COMP and DTEN bits. The following table shows the values that channels (n) and (n+1) are forced by initialization when the COMP and DTEN bits are zero.

**Table 40-309. Initialization behavior when (COMP = 0 and DTEN = 0)**

| CH(n)OI | CH(n+1)OI | Channel (n) Output | Channel (n+1) Output |
|---|---|---|---|
| 0 | 0 | is forced to zero | is forced to zero |
| 0 | 1 | is forced to zero | is forced to one |
| 1 | 0 | is forced to one | is forced to zero |
| 1 | 1 | is forced to one | is forced to one |

The following table shows the values that channels (n) and (n+1) are forced by initialization when (COMP = 1) or (DTEN = 1).

**Table 40-310. Initialization behavior when (COMP = 1 or DTEN = 1)**

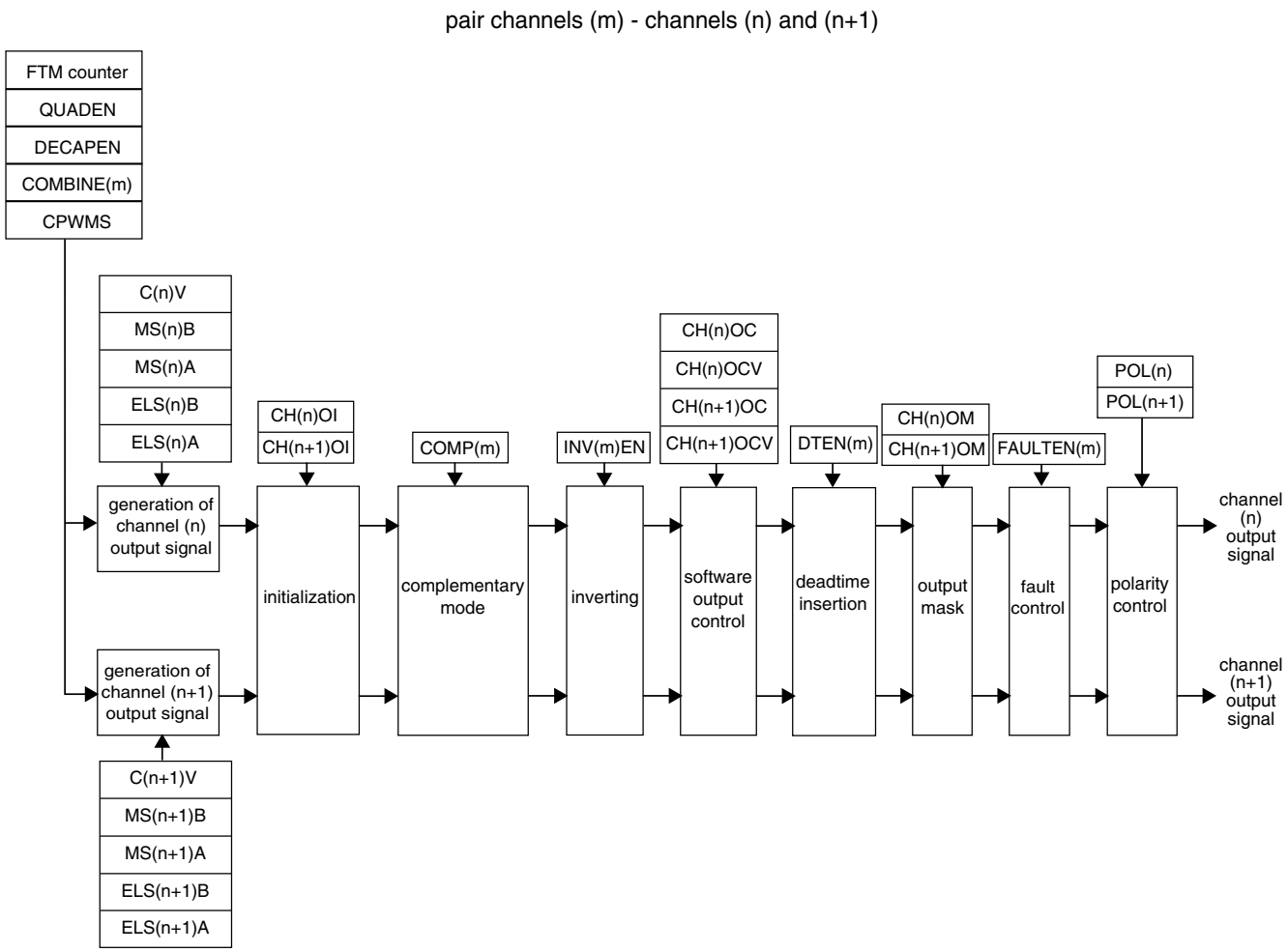
| CH(n)OI | CH(n+1)OI | Channel (n) Output | Channel (n+1) Output |
|---|---|---|---|
| 0 | X | is forced to zero | is forced to one |
| 1 | X | is forced to one | is forced to zero |

**Note**

The initialization feature must be used only in Combine mode and with disabled FTM counter. See the description of the CLKS field in the Status and Control register.

## 40.4.19 Features priority

The following figure shows the priority of the features used at the generation of channels (n) and (n+1) outputs signals.




**NOTE**

The channels (n) and (n+1) are in output compare, EPWM, CPWM or combine modes.

**Figure 40-278. Priority of the features used at the generation of channels (n) and (n+1) outputs signals**

### Note

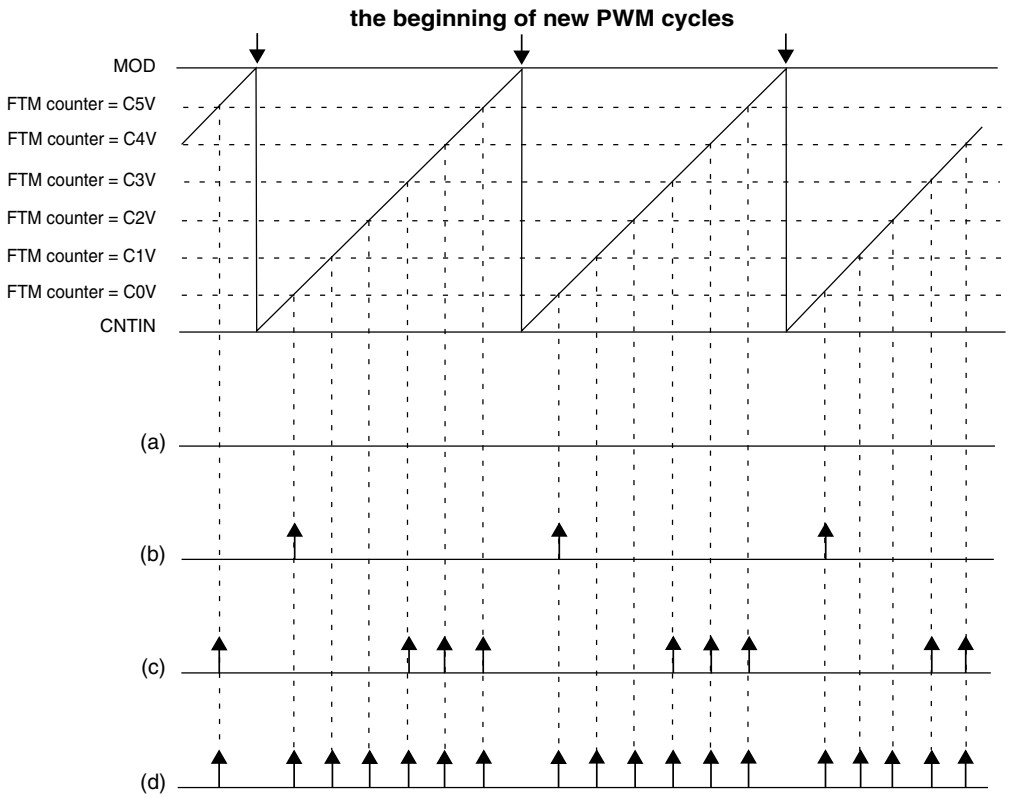
The Initialization feature must not be used with Inverting and Software output control features.

## 40.4.20 Channel trigger output

If CHjTRIG = 1, where j = 0, 1, 2, 3, 4, or 5, then the FTM generates a trigger when the channel (j) match occurs (FTM counter = C(j)V).

The channel trigger output provides a trigger signal that is used for on-chip modules.

The FTM is able to generate multiple triggers in one PWM period. Because each trigger is generated for a specific channel, several channels are required to implement this functionality. This behavior is described in the following figure.




**NOTE**

(a) CH0TRIG = 0, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 0, CH4TRIG = 0, CH5TRIG = 0
(b) CH0TRIG = 1, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 0, CH4TRIG = 0, CH5TRIG = 0
(c) CH0TRIG = 0, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 1, CH4TRIG = 1, CH5TRIG = 1
(d) CH0TRIG = 1, CH1TRIG = 1, CH2TRIG = 1, CH3TRIG = 1, CH4TRIG = 1, CH5TRIG = 1

**Figure 40-279. Channel match trigger**

### Note

The channel match trigger must be used only in Combine mode.

## 40.4.21 Initialization trigger

If INITTRIGEN = 1, then the FTM generates a trigger when the FTM counter is updated with the CNTIN register value in the following cases.

- The FTM counter is automatically updated with the CNTIN register value by the selected counting mode.

- When there is a write to CNT register.
- When there is the FTM counter synchronization.
- If (CNT = CNTIN), (CLKS[1:0] = 0:0), and a value different from zero is written to CLKS[1:0] bits.

The following figures show these cases.

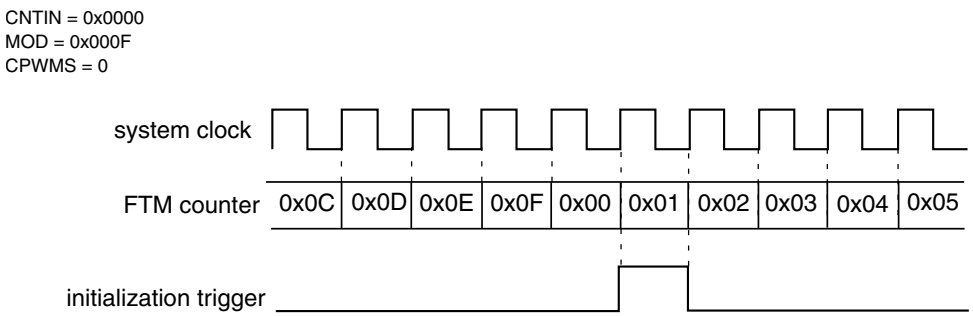



**Figure 40-280. Initialization trigger is generated when the FTM counting achieves the CNTIN register value**

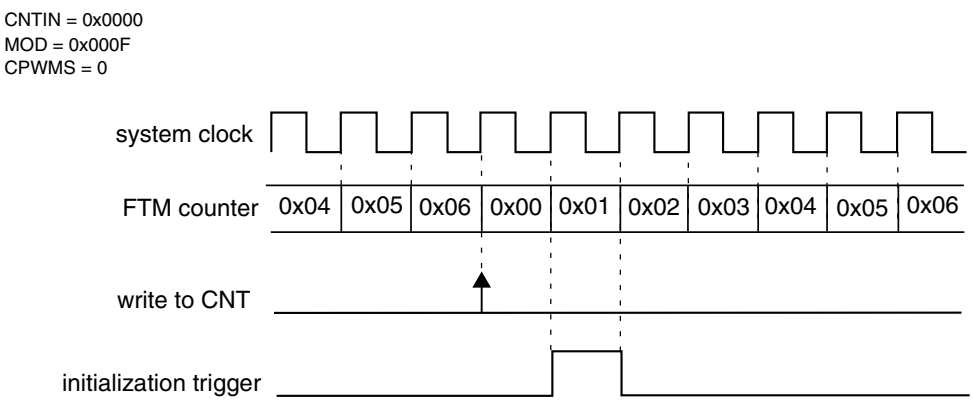



**Figure 40-281. Initialization trigger is generated when there is a write to CNT register**

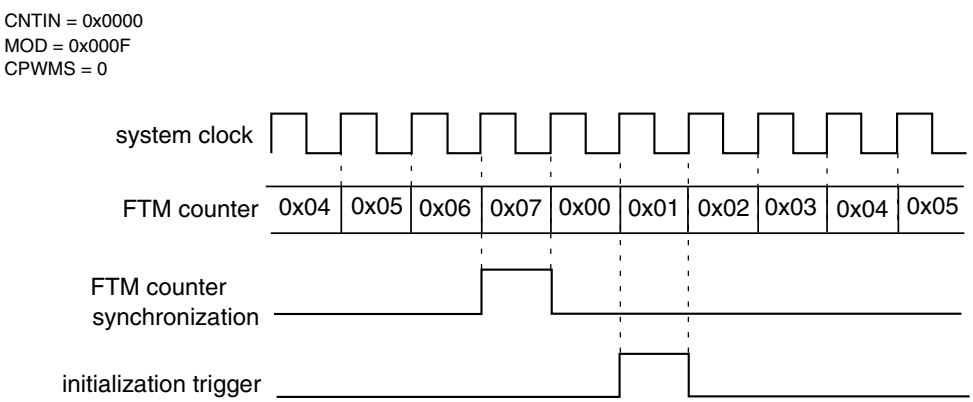



**Figure 40-282. Initialization trigger is generated when there is the FTM counter synchronization**

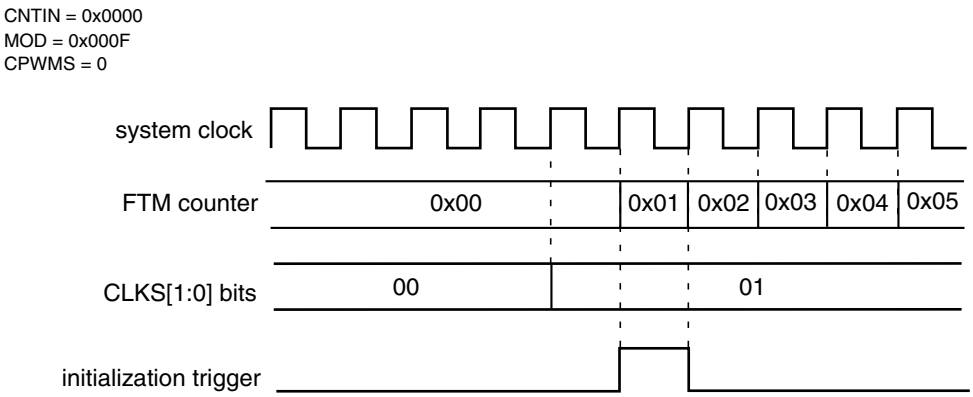



**Figure 40-283. Initialization trigger is generated if (CNT = CNTIN), (CLKS[1:0] = 0:0), and a value different from zero is written to CLKS[1:0] bits**

The initialization trigger output provides a trigger signal that is used for on-chip modules.

**Note**

The initialization trigger must be used only in Combine mode.

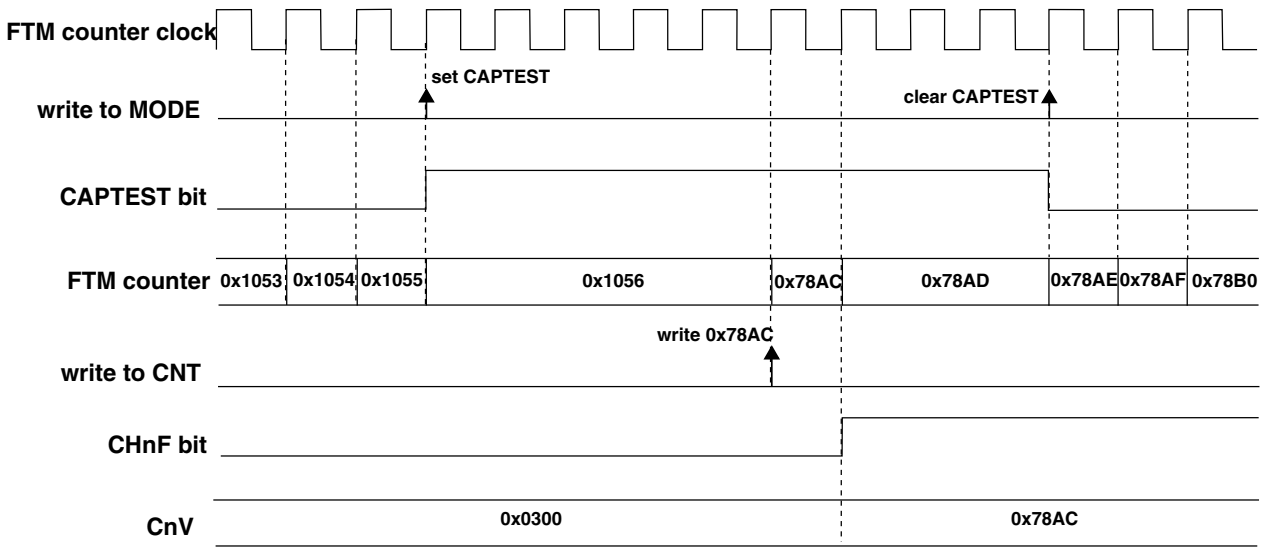
## 40.4.22 Capture Test mode

The Capture Test mode allows to test the CnV registers, the FTM counter and the interconnection logic between the FTM counter and CnV registers.

In this test mode, all channels must be configured for Input Capture mode and FTM counter must be configured to the Up counting.

When the Capture Test mode is enabled (CAPTEST = 1), the FTM counter is frozen and any write to CNT register updates directly the FTM counter; see the following figure. After it was written, all CnV registers are updated with the written value to CNT register and CHnF bits are set. Therefore, the FTM counter is updated with its next value according to its configuration. Its next value depends on CNTIN, MOD, and the written value to FTM counter.

The next reads of CnV registers return the written value to the FTM counter and the next reads of CNT register return FTM counter next value.




NOTE
- FTM counter configuration: (FTMEN = 1), (QUADEN = 0), (CAPTEST = 1), (CPWMS = 0), (CNTIN = 0x0000), and (MOD = 0xFFFF)
- FTM channel n configuration: input capture mode - (DECAPEN = 0), (COMBINE = 0), and (MSnB:MSnA = 0:0)

**Figure 40-284. Capture Test mode**

## 40.4.23 DMA

The channel generates a DMA transfer request according to DMA and CHnIE bits. See the following table.

**Table 40-311. Channel DMA transfer request**

| DMA | CHnIE | Channel DMA Transfer Request | Channel Interrupt |
|---|---|---|---|
| 0 | 0 | The channel DMA transfer request is not generated. | The channel interrupt is not generated. |
| 0 | 1 | The channel DMA transfer request is not generated. | The channel interrupt is generated if (CHnF = 1). |
| 1 | 0 | The channel DMA transfer request is not generated. | The channel interrupt is not generated. |
| 1 | 1 | The channel DMA transfer request is generated if (CHnF = 1). | The channel interrupt is not generated. |

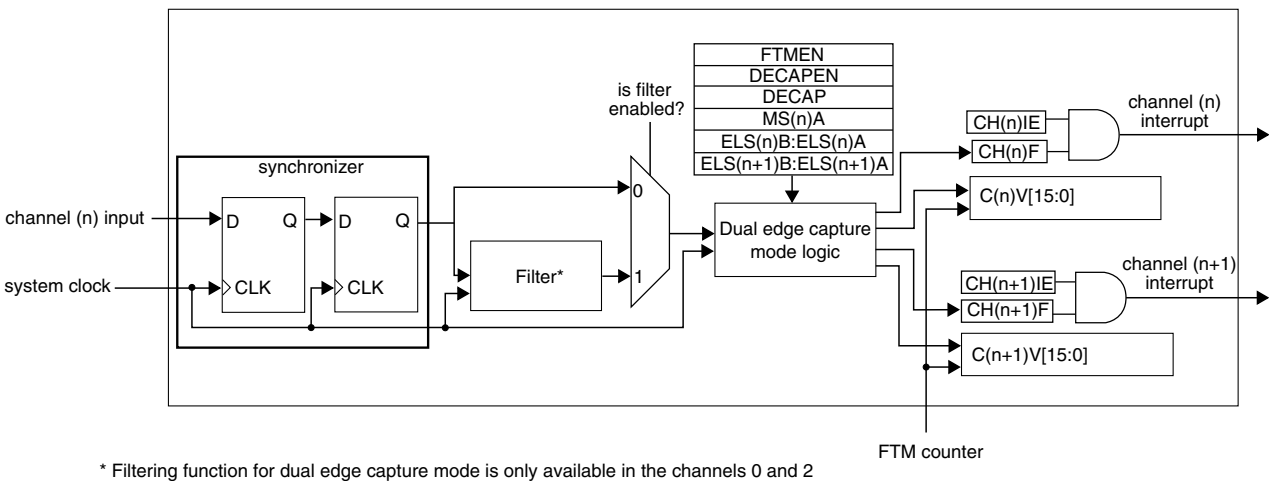
If DMA = 1, the CHnF bit is cleared either by channel DMA transfer done or reading CnSC while CHnF is set and then writing a zero to CHnF bit according to CHnIE bit. See the following table.

**Table 40-312. Clear CHnF bit when DMA = 1**

| CHnIE | How CHnF Bit Can Be Cleared |
|---|---|
| 0 | CHnF bit is cleared either when the channel DMA transfer is done or by reading CnSC while CHnF is set and then writing a 0 to CHnF bit. |
| 1 | CHnF bit is cleared when the channel DMA transfer is done. |

## 40.4.24 Dual Edge Capture mode

The Dual Edge Capture mode is selected if FTMEN = 1 and DECAPEN = 1. This mode allows to measure a pulse width or period of the signal on the input of channel (n) of a channel pair. The channel (n) filter can be active in this mode when n is 0 or 2.




* Filtering function for dual edge capture mode is only available in the channels 0 and 2

**Figure 40-285. Dual Edge Capture mode block diagram**

The MS(n)A bit defines if the Dual Edge Capture mode is one-shot or continuous.

The ELS(n)B:ELS(n)A bits select the edge that is captured by channel (n), and ELS(n +1)B:ELS(n+1)A bits select the edge that is captured by channel (n+1). If both ELS(n)B:ELS(n)A and ELS(n+1)B:ELS(n+1)A bits select the same edge, then it is the period measurement. If these bits select different edges, then it is a pulse width measurement.

In the Dual Edge Capture mode, only channel (n) input is used and channel (n+1) input is ignored.

If the selected edge by channel (n) bits is detected at channel (n) input, then CH(n)F bit is set and the channel (n) interrupt is generated (if CH(n)IE = 1). If the selected edge by channel (n+1) bits is detected at channel (n) input and (CH(n)F = 1), then CH(n+1)F bit is set and the channel (n+1) interrupt is generated (if CH(n+1)IE = 1).

The C(n)V register stores the value of FTM counter when the selected edge by channel (n) is detected at channel (n) input. The C(n+1)V register stores the value of FTM counter when the selected edge by channel (n+1) is detected at channel (n) input.

In this mode, a coherency mechanism ensures coherent data when the C(n)V and C(n +1)V registers are read. The only requirement is that C(n)V must be read before C(n +1)V.

**Note**

- The CH(n)F, CH(n)IE, MS(n)A, ELS(n)B, and ELS(n)A bits are channel (n) bits.
- The CH(n+1)F, CH(n+1)IE, MS(n+1)A, ELS(n+1)B, and ELS(n+1)A bits are channel (n+1) bits.
- The Dual Edge Capture mode must be used with ELS(n)B:ELS(n)A = 0:1 or 1:0, ELS(n+1)B:ELS(n+1)A = 0:1 or 1:0 and the FTM counter in Free running counter.

### 40.4.24.1 One-Shot Capture mode

The One-Shot Capture mode is selected when (FTMEN = 1), (DECAPEN = 1), and (MS(n)A = 0). In this capture mode, only one pair of edges at the channel (n) input is captured. The ELS(n)B:ELS(n)A bits select the first edge to be captured, and ELS(n +1)B:ELS(n+1)A bits select the second edge to be captured.

The edge captures are enabled while DECAP bit is set. For each new measurement in One-Shot Capture mode, first the CH(n)F and CH(n+1) bits must be cleared, and then the DECAP bit must be set.

In this mode, the DECAP bit is automatically cleared by FTM when the edge selected by channel (n+1) is captured. Therefore, while DECAP bit is set, the one-shot capture is in process. When this bit is cleared, both edges were captured and the captured values are ready for reading in the C(n)V and C(n+1)V registers.

Similarly, when the CH(n+1)F bit is set, both edges were captured and the captured values are ready for reading in the C(n)V and C(n+1)V registers.

### 40.4.24.2 Continuous Capture mode

The Continuous Capture mode is selected when (FTMEN = 1), (DECAPEN = 1), and (MS(n)A = 1). In this capture mode, the edges at the channel (n) input are captured continuously. The ELS(n)B:ELS(n)A bits select the initial edge to be captured, and ELS(n+1)B:ELS(n+1)A bits select the final edge to be captured.

The edge captures are enabled while DECAP bit is set. For the initial use, first the CH(n)F and CH(n+1)F bits must be cleared, and then DECAP bit must be set to start the continuous measurements.

When the CH(n+1)F bit is set, both edges were captured and the captured values are ready for reading in the C(n)V and C(n+1)V registers. The latest captured values are always available in these registers even after the DECAP bit is cleared.

In this mode, it is possible to clear only the CH(n+1)F bit. Therefore, when the CH(n+1)F bit is set again, the latest captured values are available in C(n)V and C(n+1)V registers.

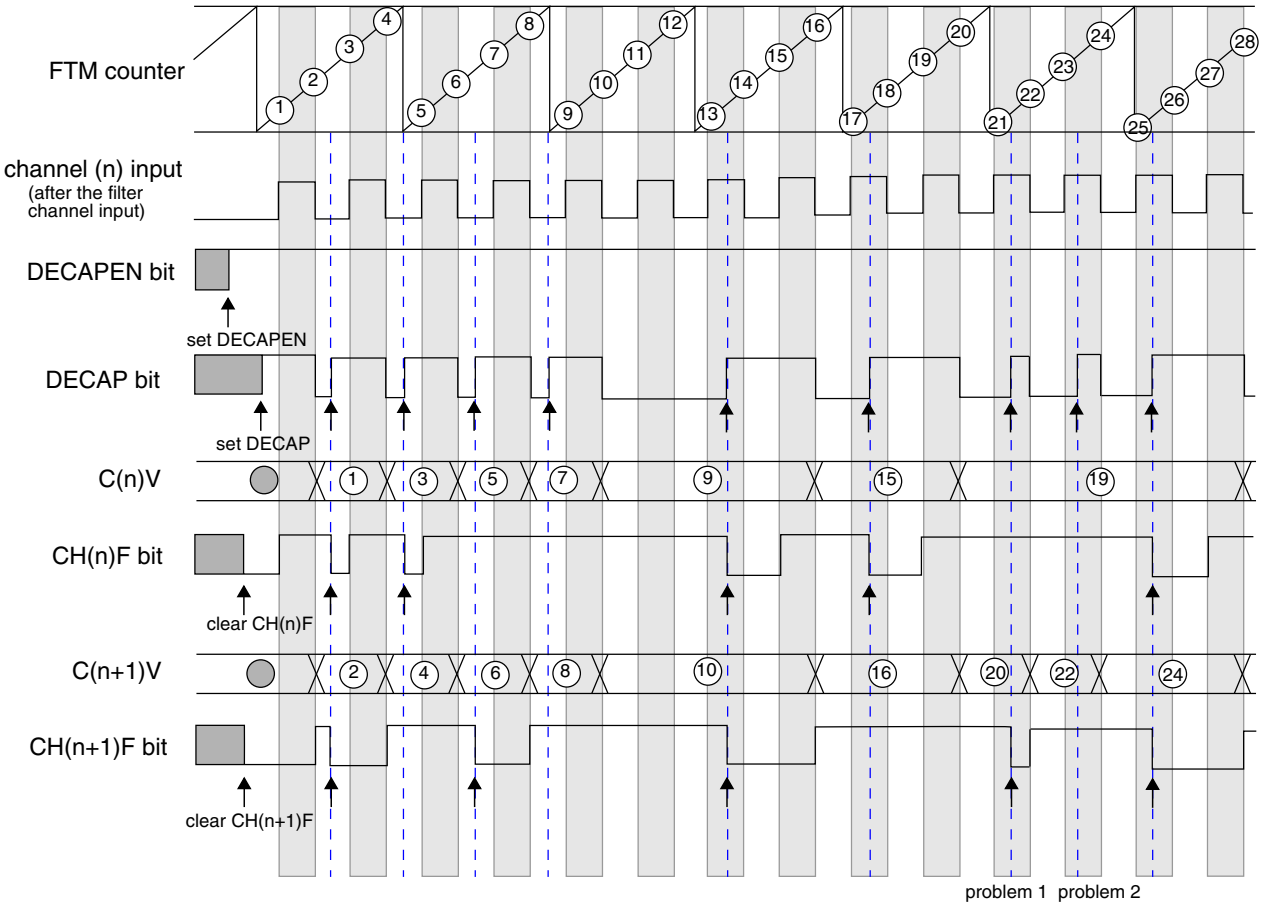
For a new sequence of the measurements in the Dual Edge Capture – Continuous mode, clear the CH(n)F and CH(n+1)F bits to start new measurements.

### 40.4.24.3 Pulse width measurement

If the channel (n) is configured to capture rising edges (ELS(n)B:ELS(n)A = 0:1) and the channel (n+1) to capture falling edges (ELS(n+1)B:ELS(n+1)A = 1:0), then the positive polarity pulse width is measured. If the channel (n) is configured to capture falling edges (ELS(n)B:ELS(n)A = 1:0) and the channel (n+1) to capture rising edges (ELS(n +1)B:ELS(n+1)A = 0:1), then the negative polarity pulse width is measured.

The pulse width measurement can be made in One-Shot Capture mode or Continuous Capture mode.

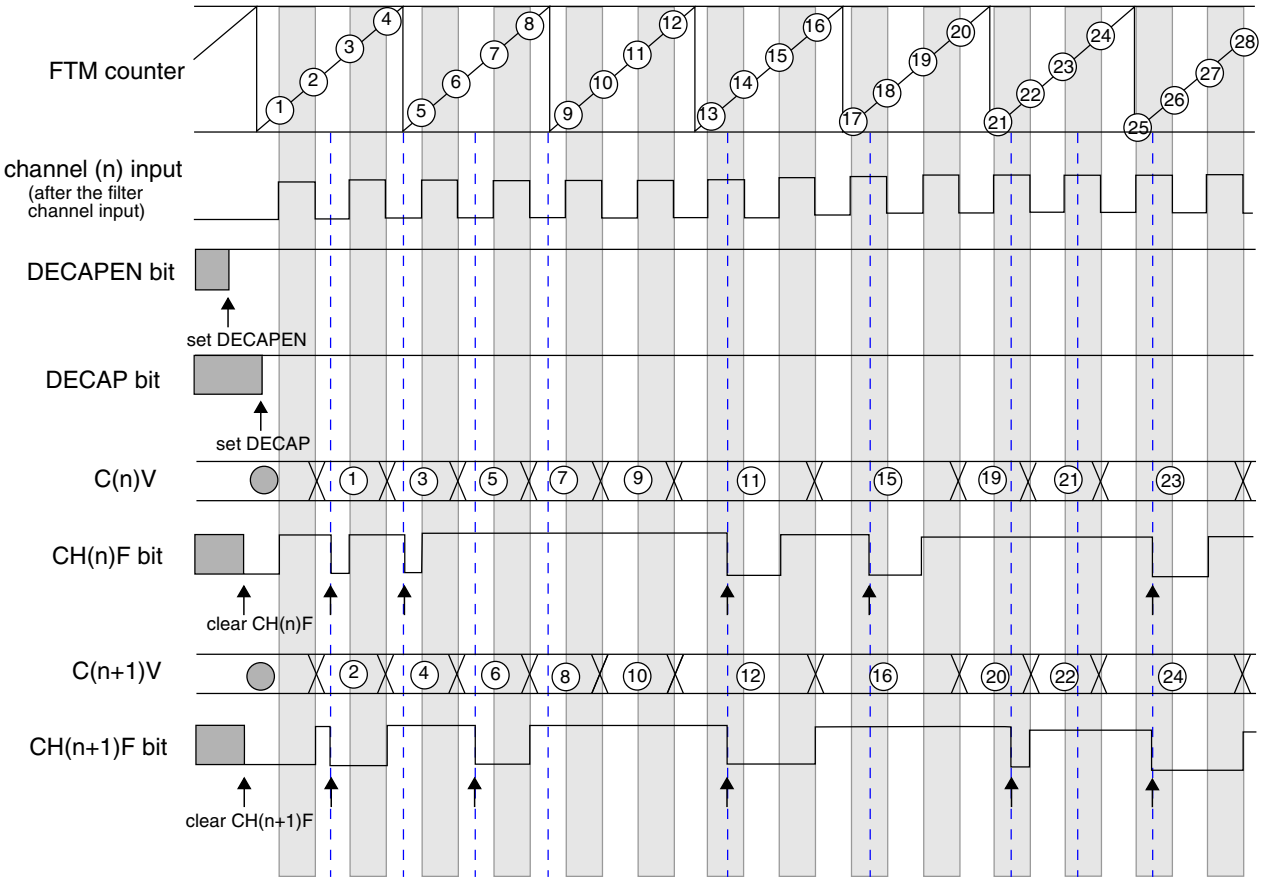
The following figure shows an example of the Dual Edge Capture – One-Shot mode used to measure the positive polarity pulse width. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. The DECAP bit is set to enable the measurement of next positive polarity pulse width. The CH(n)F bit is set when the first edge of this pulse is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set and DECAP bit is cleared when the second edge of this pulse is detected, that is, the edge selected by ELS(n+1)B:ELS(n+1)A bits. Both DECAP and CH(n+1)F bits indicate when two edges of the pulse were captured and the C(n)V and C(n+1)V registers are ready for reading.




Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.
- Problem 1: channel (n) input = 1, set DECAP, not clear CH(n)F, and clear CH(n+1)F.
- Problem 2: channel (n) input = 1, set DECAP, not clear CH(n)F, and not clear CH(n+1)F.

**Figure 40-286. Dual Edge Capture – One-Shot mode for positive polarity pulse width measurement**

The following figure shows an example of the Dual Edge Capture – Continuous mode used to measure the positive polarity pulse width. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. While the DECAP bit is set the configured measurements are made. The CH(n)F bit is set when the first edge of the positive polarity pulse is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set when the second edge of this pulse is detected, that is, the edge selected by ELS(n +1)B:ELS(n+1)A bits. The CH(n+1)F bit indicates when two edges of the pulse were captured and the C(n)V and C(n+1)V registers are ready for reading.




Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.

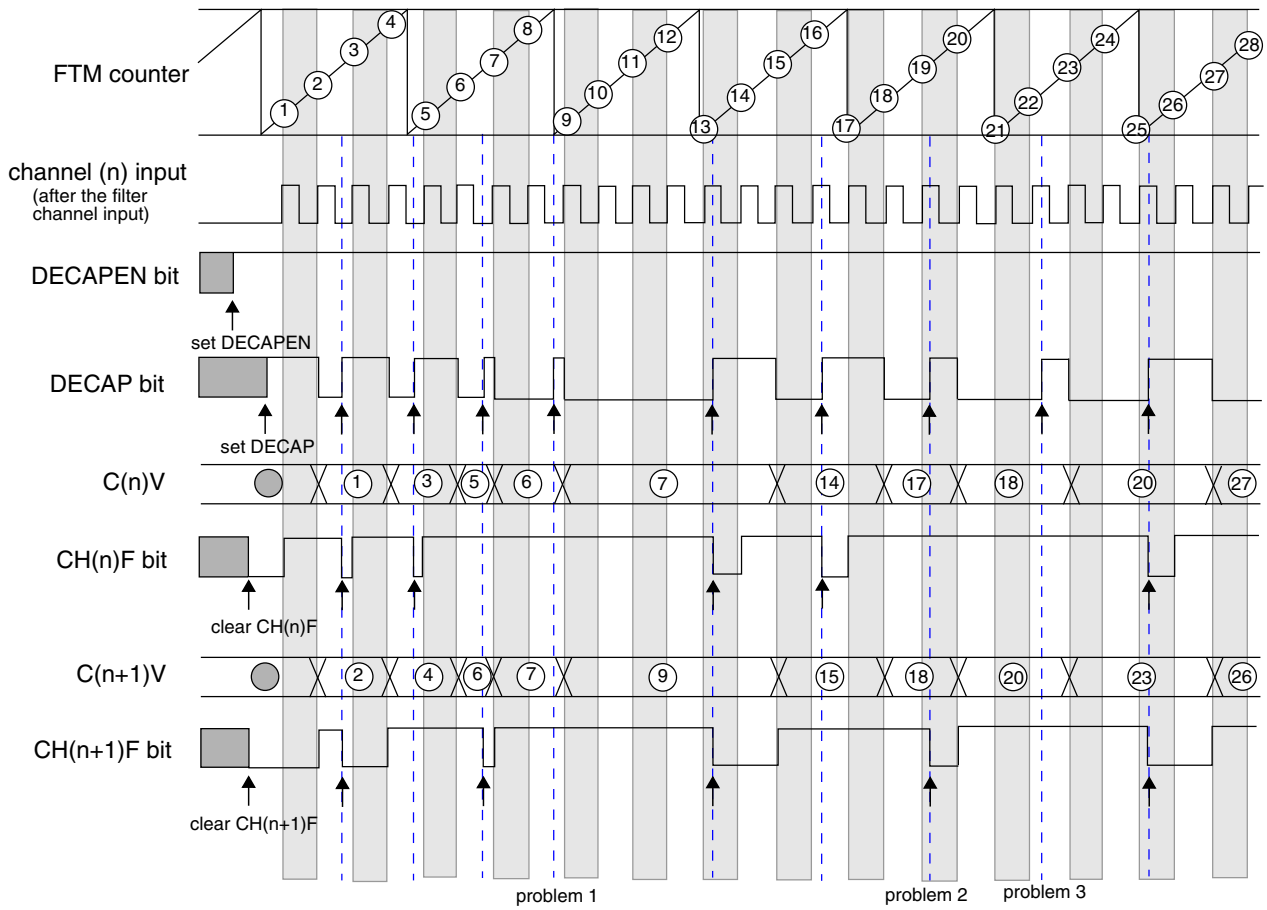
**Figure 40-287. Dual Edge Capture – Continuous mode for positive polarity pulse width measurement**

## 40.4.24.4 Period measurement

If the channels (n) and (n+1) are configured to capture consecutive edges of the same polarity, then the period of the channel (n) input signal is measured. If both channels (n) and (n+1) are configured to capture rising edges (ELS(n)B:ELS(n)A = 0:1 and ELS(n +1)B:ELS(n+1)A = 0:1), then the period between two consecutive rising edges is measured. If both channels (n) and (n+1) are configured to capture falling edges (ELS(n)B:ELS(n)A = 1:0 and ELS(n+1)B:ELS(n+1)A = 1:0), then the period between two consecutive falling edges is measured.

The period measurement can be made in One-Shot Capture mode or Continuous Capture mode.

The following figure shows an example of the Dual Edge Capture – One-Shot mode used to measure the period between two consecutive rising edges. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. The DECAP bit is set to enable the measurement of next period. The CH(n)F bit is set when the first rising edge is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set and DECAP bit is cleared when the second rising edge is detected, that is, the edge selected by ELS(n+1)B:ELS(n+1)A bits. Both DECAP and CH(n+1)F bits indicate when two selected edges were captured and the C(n)V and C(n+1)V registers are ready for reading.

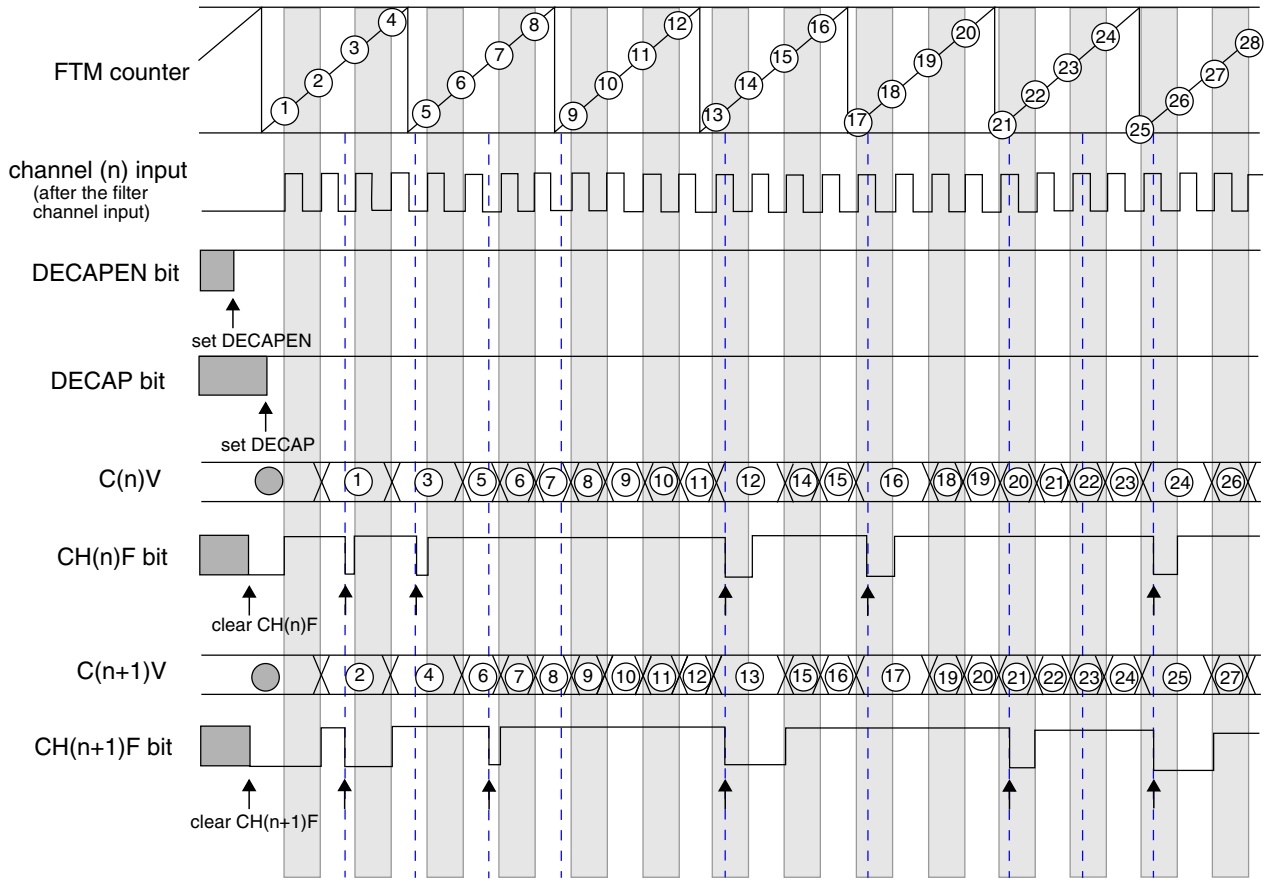



Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.
- Problem 1: channel (n) input = 0, set DECAP, not clear CH(n)F, and not clear CH(n+1)F.
- Problem 2: channel (n) input = 1, set DECAP, not clear CH(n)F, and clear CH(n+1)F.
- Problem 3: channel (n) input = 1, set DECAP, not clear CH(n)F, and not clear CH(n+1)F.

**Figure 40-288. Dual Edge Capture – One-Shot mode to measure of the period between two consecutive rising edges**

The following figure shows an example of the Dual Edge Capture – Continuous mode used to measure the period between two consecutive rising edges. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. While the DECAP bit is set the configured measurements are made. The CH(n)F bit is set when the first rising edge is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set

when the second rising edge is detected, that is, the edge selected by ELS(n+1)B:ELS(n +1)A bits. The CH(n+1)F bit indicates when two edges of the period were captured and the C(n)V and C(n+1)V registers are ready for reading.




Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.

**Figure 40-289. Dual Edge Capture – Continuous mode to measure of the period between two consecutive rising edges**

## 40.4.24.5 Read coherency mechanism

The Dual Edge Capture mode implements a read coherency mechanism between the FTM counter value captured in C(n)V and C(n+1)V registers. The read coherency mechanism is illustrated in the following figure. In this example, the channels (n) and (n +1) are in Dual Edge Capture – Continuous mode for positive polarity pulse width measurement. Thus, the channel (n) is configured to capture the FTM counter value when there is a rising edge at channel (n) input signal, and channel (n+1) to capture the FTM counter value when there is a falling edge at channel (n) input signal.

When a rising edge occurs in the channel (n) input signal, the FTM counter value is captured into channel (n) capture buffer. The channel (n) capture buffer value is transferred to C(n)V register when a falling edge occurs in the channel (n) input signal. C(n)V register has the FTM counter value when the previous rising edge occurred, and the channel (n) capture buffer has the FTM counter value when the last rising edge occurred.

When a falling edge occurs in the channel (n) input signal, the FTM counter value is captured into channel (n+1) capture buffer. The channel (n+1) capture buffer value is transferred to C(n+1)V register when the C(n)V register is read.

In the following figure, the read of C(n)V returns the FTM counter value when the event 1 occurred and the read of C(n+1)V returns the FTM counter value when the event 2 occurred.

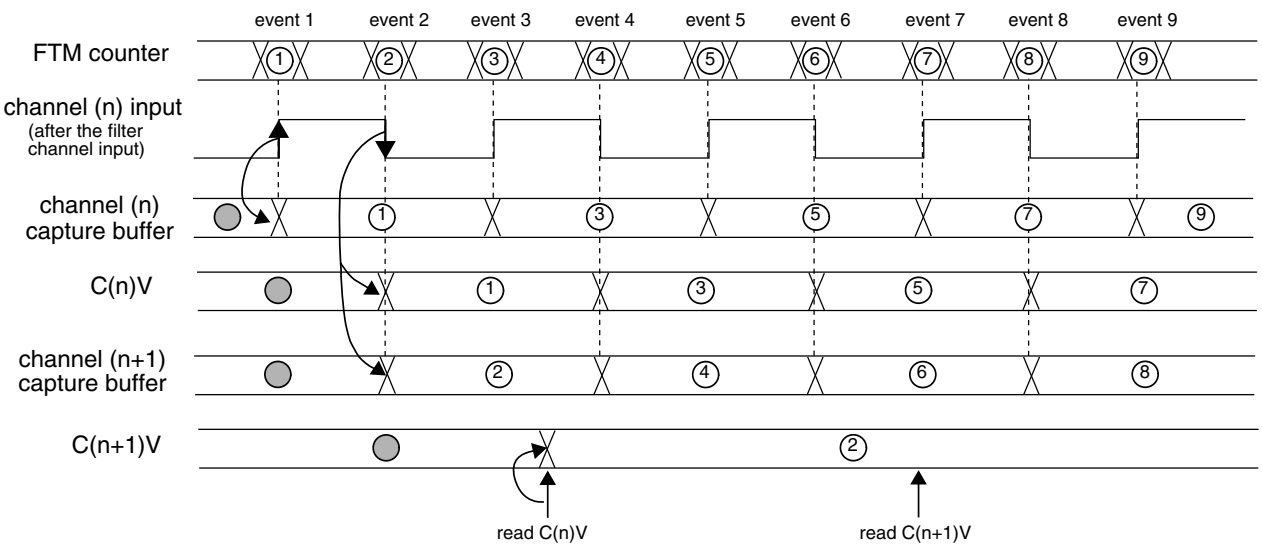



**Figure 40-290. Dual Edge Capture mode read coherency mechanism**

C(n)V register must be read prior to C(n+1)V register in dual edge capture one-shot and continuous modes for the read coherency mechanism works properly.

## 40.4.25 Quadrature Decoder mode

The Quadrature Decoder mode is selected if (FTMEN = 1) and (QUADEN = 1). The Quadrature Decoder mode uses the input signals phase A and B to control the FTM counter increment and decrement. The following figure shows the quadrature decoder block diagram.

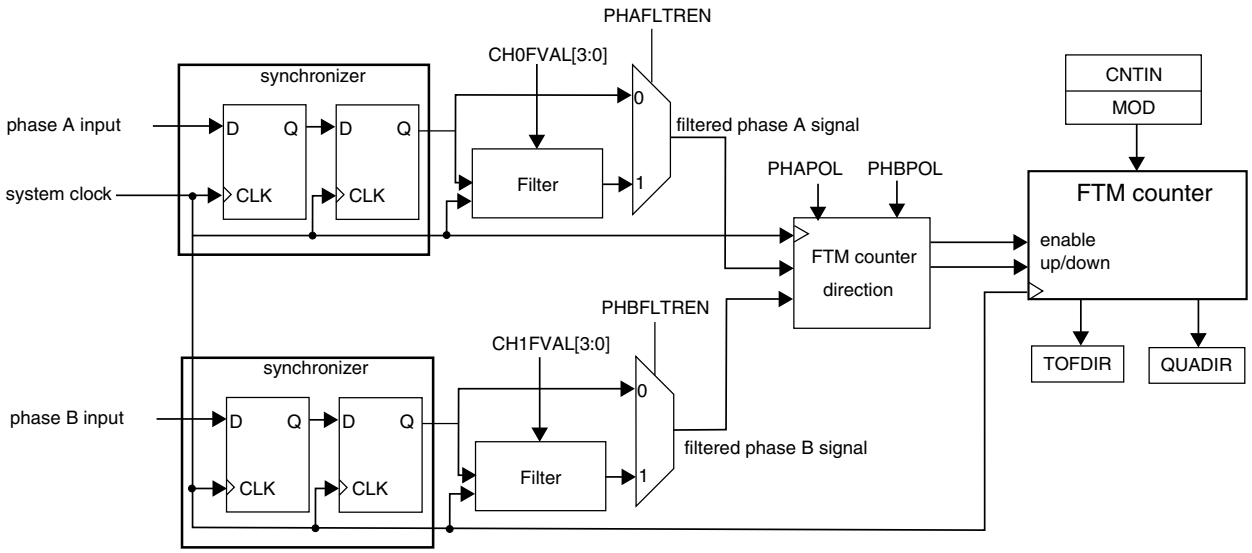



**Figure 40-291. Quadrature Decoder block diagram**

Each one of input signals phase A and B has a filter that is equivalent to the filter used in the channels input; Filter for Input Capture mode. The phase A input filter is enabled by PHAFLTREN bit and this filter's value is defined by CH0FVAL[3:0] bits (CH(n)FVAL[3:0] bits in FILTER0 register). The phase B input filter is enabled by PHBFLTREN bit and this filter's value is defined by CH1FVAL[3:0] bits (CH(n +1)FVAL[3:0] bits in FILTER0 register).

Except for CH0FVAL[3:0] and CH1FVAL[3:0] bits, no channel logic is used in Quadrature Decoder mode.

### Note

Notice that the FTM counter is clocked by the phase A and B input signals when quadrature decoder mode is selected. Therefore it is expected that the Quadrature Decoder be used only with the FTM channels in input capture or output compare modes.

The PHAPOL bit selects the polarity of the phase A input, and the PHBPOL bit selects the polarity of the phase B input.

The QUADMODE selects the encoding mode used in the Quadrature Decoder mode. If QUADMODE = 1, then the count and direction encoding mode is enabled; see the following figure. In this mode, the phase B input value indicates the counting direction, and the phase A input defines the counting rate. The FTM counter is updated when there is a rising edge at phase A input signal.

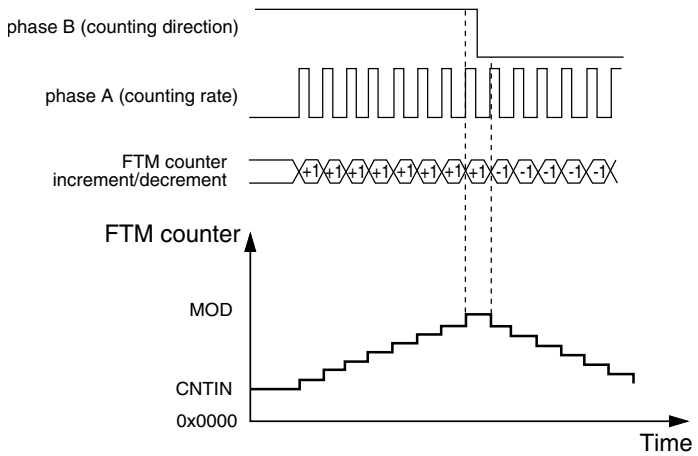



**Figure 40-292. Quadrature Decoder – Count and Direction Encoding mode**

If QUADMODE = 0, then the Phase A and Phase B Encoding mode is enabled; see the following figure. In this mode, the relationship between phase A and B signals indicates the counting direction, and phase A and B signals define the counting rate. The FTM counter is updated when there is an edge either at the phase A or phase B signals.

If PHAPOL = 0 and PHBPOL = 0, then the FTM counter increment happens when:

- there is a rising edge at phase A signal and phase B signal is at logic zero;
- there is a rising edge at phase B signal and phase A signal is at logic one;
- there is a falling edge at phase B signal and phase A signal is at logic zero;
- there is a falling edge at phase A signal and phase B signal is at logic one;

and the FTM counter decrement happens when:

- there is a falling edge at phase A signal and phase B signal is at logic zero;
- there is a falling edge at phase B signal and phase A signal is at logic one;
- there is a rising edge at phase B signal and phase A signal is at logic zero;
- there is a rising edge at phase A signal and phase B signal is at logic one.

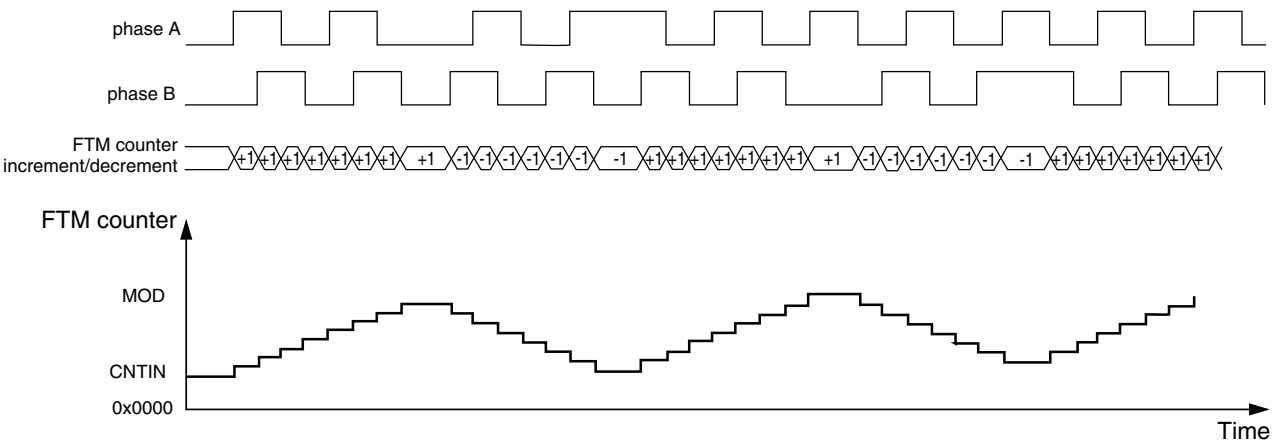



**Figure 40-293. Quadrature Decoder – Phase A and Phase B Encoding mode**

The following figure shows the FTM counter overflow in up counting. In this case, when the FTM counter changes from MOD to CNTIN, TOF and TOFDIR bits are set. TOF bit indicates the FTM counter overflow occurred. TOFDIR indicates the counting was up when the FTM counter overflow occurred.

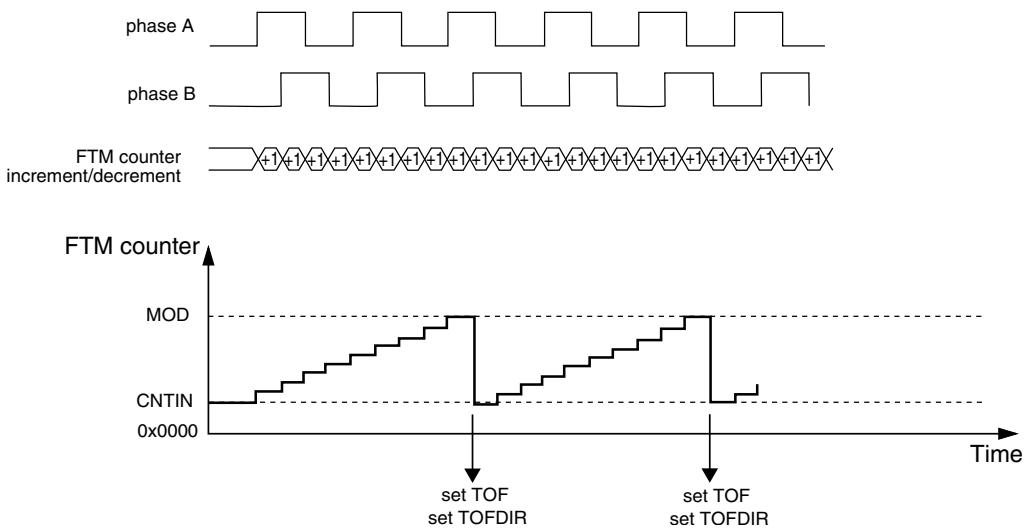



**Figure 40-294. FTM Counter overflow in up counting for Quadrature Decoder mode**

The following figure shows the FTM counter overflow in down counting. In this case, when the FTM counter changes from CNTIN to MOD, TOF bit is set and TOFDIR bit is cleared. TOF bit indicates the FTM counter overflow occurred. TOFDIR indicates the counting was down when the FTM counter overflow occurred.

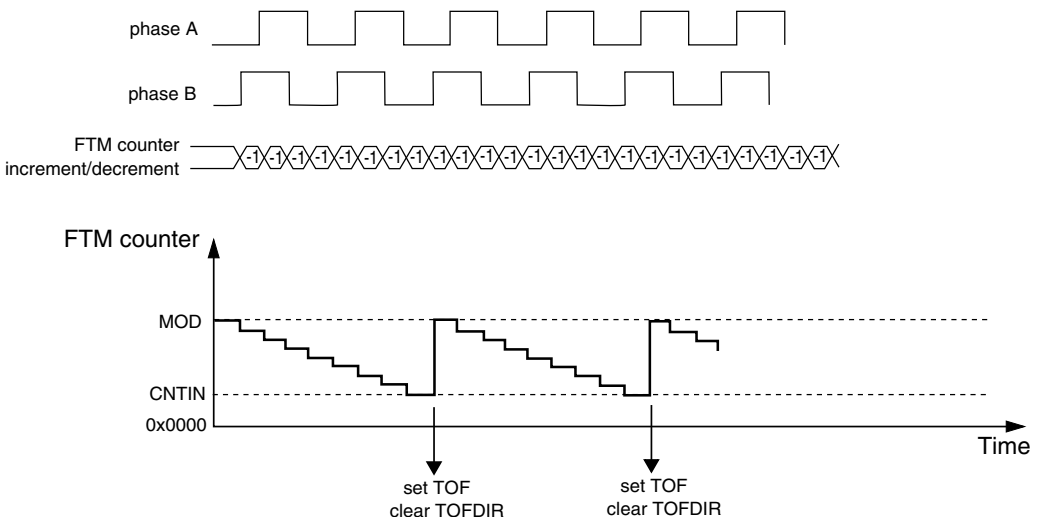



**Figure 40-295. FTM counter overflow in down counting for Quadrature Decoder mode**

### 40.4.25.1 Quadrature Decoder boundary conditions

The following figures show the FTM counter responding to motor jittering typical in motor position control applications.

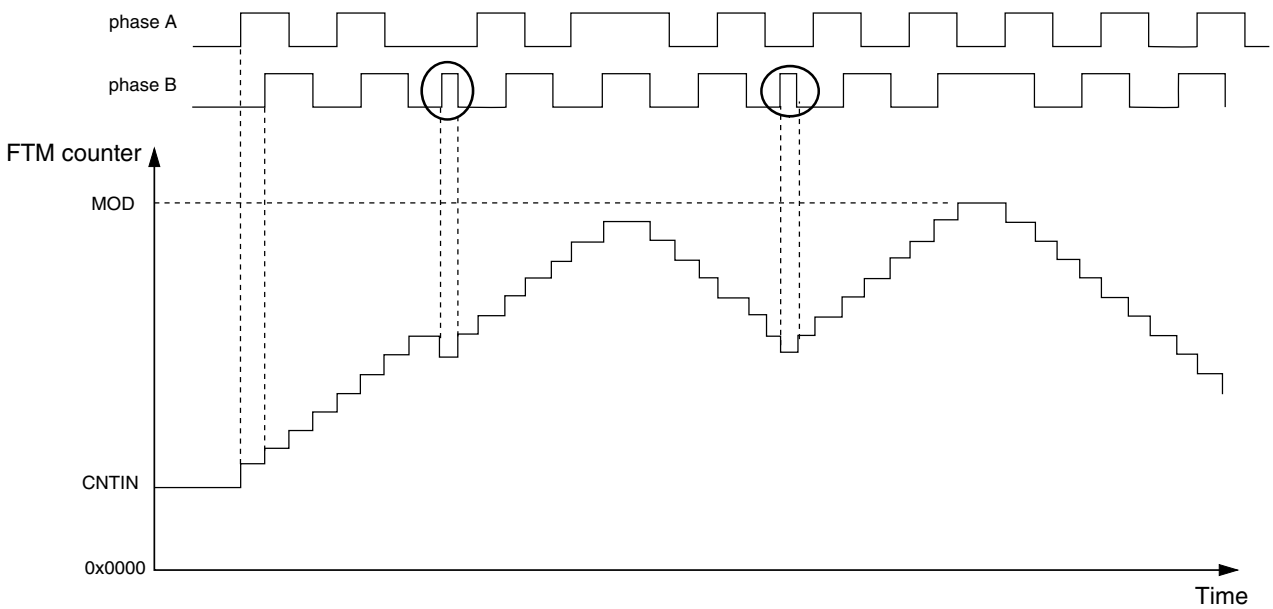



**Figure 40-296. Motor position jittering in a mid count value**

The following figure shows motor jittering produced by the phase B and A pulses respectively:

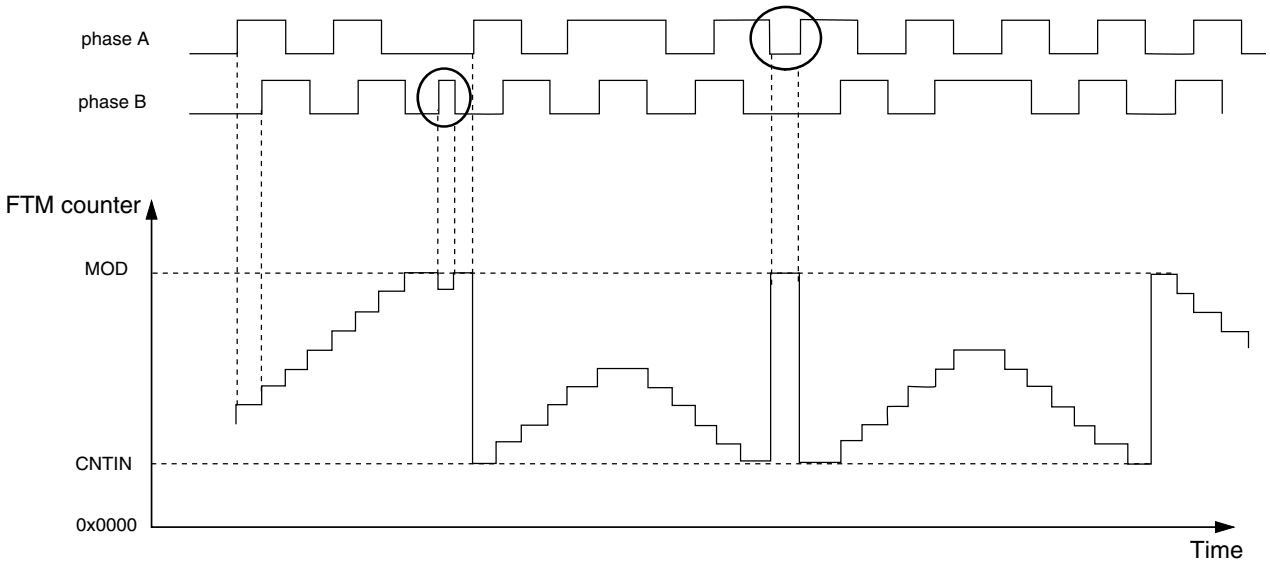



**Figure 40-297. Motor position jittering near maximum and minimum count value**

The first highlighted transition causes a jitter on the FTM counter value near the maximum count value (MOD). The second indicated transition occurs on phase A and causes the FTM counter transition between the maximum and minimum count values which are defined by MOD and CNTIN registers.

The appropriate settings of the phase A and phase B input filters are important to avoid glitches that may cause oscillation on the FTM counter value. The preceding figures show examples of oscillations that can be caused by poor input filter setup. Thus, it is important to guarantee a minimum pulse width to avoid these oscillations.

## 40.4.26 BDM mode

When the chip is in BDM mode, the BDMMODE[1:0] bits select the behavior of the FTM counter, the CH(n)F bit, the channels output, and the writes to the MOD, CNTIN, and C(n)V registers according to the following table.

**Table 40-313. FTM behavior when the chip Is in BDM mode**

| BDMMODE | FTM Counter | CH(n)F Bit | FTM Channels Output | Writes to MOD, CNTIN, and C(n)V Registers |
|---|---|---|---|---|
| 00 | Stopped | can be set | Functional mode | Writes to these registers bypass the registers buffers |
| 01 | Stopped | is not set | The channels outputs are forced to their safe value according to POLn bit | Writes to these registers bypass the registers buffers |
| 10 | Stopped | is not set | The channels outputs are frozen when the chip enters in BDM mode | Writes to these registers bypass the registers buffers |

*Table continues on the next page...*

**Table 40-313. FTM behavior when the chip Is in BDM mode (continued)**

| BDMMODE | FTM Counter | CH(n)F Bit | FTM Channels Output | Writes to MOD, CNTIN, and C(n)V Registers |
|---|---|---|---|---|
| 11 | Functional mode | can be set | Functional mode | Functional mode |

Note that if BDMMODE[1:0] = 2'b00 then the channels outputs remain at the value when the chip enters in BDM mode, because the FTM counter is stopped. However, the following situations modify the channels outputs in this BDM mode.

- Write any value to CNT register; see Counter reset. In this case, the FTM counter is updated with the CNTIN register value and the channels outputs are updated to the initial value – except for those channels set to Output Compare mode.
- FTM counter is reset by PWM Synchronization mode; see FTM counter synchronization. In this case, the FTM counter is updated with the CNTIN register value and the channels outputs are updated to the initial value – except for channels in Output Compare mode.
- In the channels outputs initialization, the channel (n) output is forced to the CH(n)OI bit value when the value 1 is written to INIT bit. See Initialization.

**Note**

The BDMMODE[1:0] = 2'b00 must not be used with the Fault control. Even if the fault control is enabled and a fault condition exists, the channels outputs values are updated as above.

## 40.4.27 Intermediate load

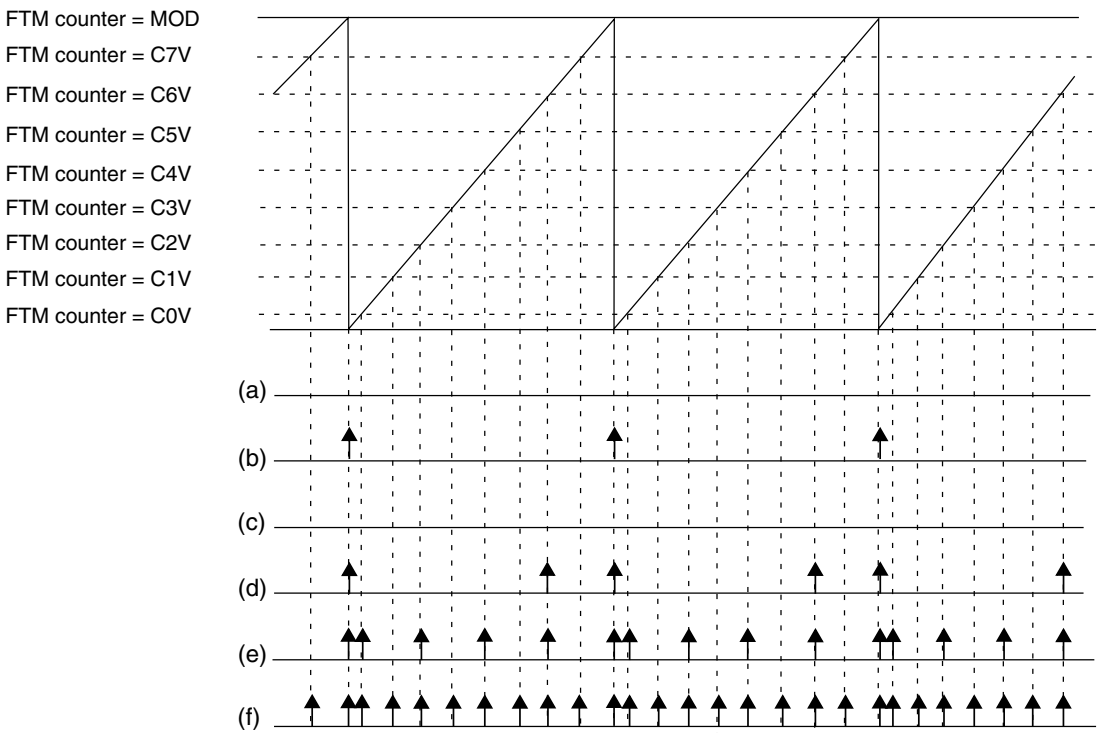
The PWMLOAD register allows to update the MOD, CNTIN, and C(n)V registers with the content of the register buffer at a defined load point. In this case, it is not required to use the PWM synchronization.

There are multiple possible loading points for intermediate load:

**Table 40-314. When possible loading points are enabled**

| Loading point | Enabled |
|---|---|
| When the FTM counter wraps from MOD value to CNTIN value | Always |
| At the channel (j) match (FTM counter = C(j)V) | When CHjSEL = 1 |

The following figure shows some examples of enabled loading points.




**NOTE**

(a) LDOK = 0, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 0, CH4SEL = 0, CH5SEL = 0, CH6SEL = 0, CH7SEL = 0
(b) LDOK = 1, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 0, CH4SEL = 0, CH5SEL = 0, CH6SEL = 0, CH7SEL = 0
(c) LDOK = 0, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 1, CH4SEL = 0, CH5SEL = 0, CH6SEL = 0, CH7SEL = 0
(d) LDOK = 1, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 0, CH4SEL = 0, CH5SEL = 0, CH6SEL = 1, CH7SEL = 0
(e) LDOK = 1, CH0SEL = 1, CH1SEL = 0, CH2SEL = 1, CH3SEL = 0, CH4SEL = 1, CH5SEL = 0, CH6SEL = 1, CH7SEL = 0
(f) LDOK = 1, CH0SEL = 1, CH1SEL = 1, CH2SEL = 1, CH3SEL = 1, CH4SEL = 1, CH5SEL = 1, CH6SEL = 1, CH7SEL = 1

**Figure 40-298. Loading points for intermediate load**

After enabling the loading points, the LDOK bit must be set for the load to occur. In this case, the load occurs at the next enabled loading point according to the following conditions:

**Table 40-315. Conditions for loads occurring at the next enabled loading point**

| When a new value was written | Then |
|---|---|
| To the MOD register | The MOD register is updated with its write buffer value. |
| To the CNTIN register and CNTINC = 1 | The CNTIN register is updated with its write buffer value. |
| To the C(n)V register and SYNCENm = 1 – where m indicates the pair channels (n) and (n+1) | The C(n)V register is updated with its write buffer value. |
| To the C(n+1)V register and SYNCENm = 1 – where m indicates the pair channels (n) and (n+1) | The C(n+1)V register is updated with its write buffer value. |

**NOTE**

- If ELSjB and ELSjA bits are different from zero, then the channel (j) output signal is generated according to the configured output mode. If ELSjB and ELSjA bits are zero, then the generated signal is not available on channel (j) output.
- If CHjIE = 1, then the channel (j) interrupt is generated when the channel (j) match occurs.
- At the intermediate load neither the channels outputs nor the FTM counter are changed. Software must set the intermediate load at a safe point in time.
- The intermediate load feature must be used only in Combine mode.

## 40.4.28 Global time base (GTB)

The global time base (GTB) is a FTM function that allows the synchronization of multiple FTM modules on a chip. The following figure shows an example of the GTB feature used to synchronize two FTM modules. In this case, the FTM A and B channels can behave as if just one FTM module was used, that is, a global time base.

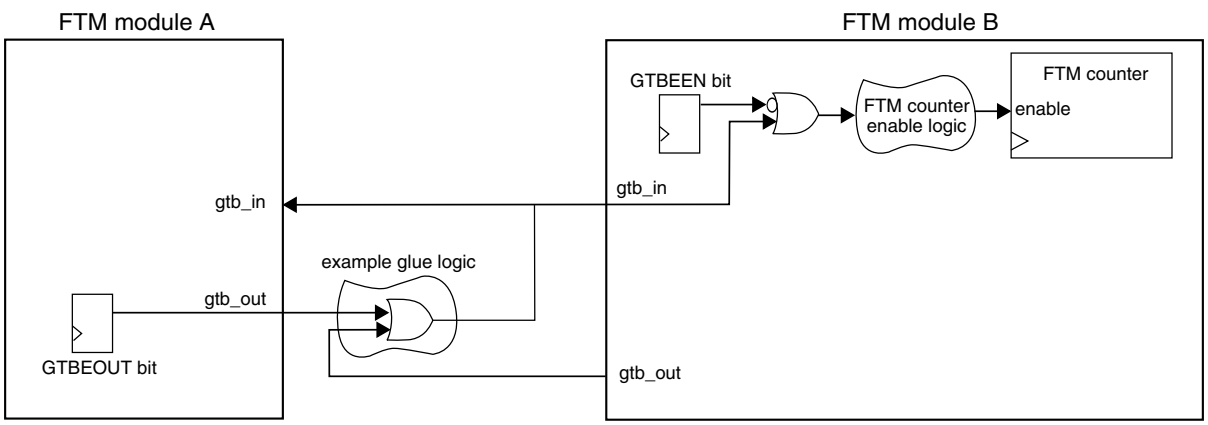



**Figure 40-299. Global time base (GTB) block diagram**

The GTB functionality is implemented by the GTBEEN and GTBEOUT bits in the CONF register, the input signal *gtb_in*, and the output signal *gtb_out*. The GTBEEN bit enables gtb_in to control the FTM counter enable signal:

- If GTBEEN = 0, each one of FTM modules works independently according to their configured mode.
- If GTBEEN = 1, the FTM counter update is enabled only when gtb_in is 1.

In the configuration described in the preceding figure, FTM modules A and B have their FTM counters enabled if at least one of the gtb_out signals from one of the FTM modules is 1. There are several possible configurations for the interconnection of the gtb_in and gtb_out signals, represented by the example glue logic shown in the figure. Note that these configurations are chip-dependent and implemented outside of the FTM modules. See the chip configuration details for the chip's specific implementation.

**NOTE**

- In order to use the GTB signals to synchronize the FTM counter of different FTM modules, the configuration of each FTM module should guarantee that its FTM counter starts counting as soon as the gtb_in signal is 1.
- The GTB feature does not provide continuous synchronization of FTM counters, meaning that the FTM counters may lose synchronization during FTM operation. The GTB feature only allows the FTM counters to *start* their operation synchronously.

### 40.4.28.1 Enabling the global time base (GTB)

To enable the GTB feature, follow these steps for each participating FTM module:

1. Stop the FTM counter: Write 00b to SC[CLKS].
2. Program the FTM to the intended configuration. The FTM counter mode needs to be consistent across all participating modules.
3. Write 1 to CONF[GTBEEN] and write 0 to CONF[GTBEOUT] at the same time.
4. Select the intended FTM counter clock source in SC[CLKS]. The clock source needs to be consistent across all participating modules.
5. Reset the FTM counter: Write any value to the CNT register.

To initiate the GTB feature in the configuration described in the preceding figure, write 1 to CONF[GTBEOUT] in the FTM module used as the time base.

## 40.5 Reset overview

The FTM is reset whenever any chip reset occurs.

When the FTM exits from reset:

- the FTM counter and the prescaler counter are zero and are stopped (CLKS[1:0] = 00b);
- the timer overflow interrupt is zero, see Timer Overflow Interrupt;

- the channels interrupts are zero, see Channel (n) Interrupt;
- the fault interrupt is zero, see Fault Interrupt;
- the channels are in input capture mode, see Input Capture mode;
- the channels outputs are zero;
- the channels pins are not controlled by FTM (ELS(n)B:ELS(n)A = 0:0) (See the table in the description of CnSC register).

The following figure shows the FTM behavior after the reset. At the reset (item 1), the FTM counter is disabled (see the description of the CLKS field in the Status and Control register), its value is updated to zero and the pins are not controlled by FTM (See the table in the description of CnSC register).

After the reset, the FTM should be configurated (item 2). It is necessary to define the FTM counter mode, the FTM counting limits (MOD and CNTIN registers value), the channels mode and CnV registers value according to the channels mode.

Thus, it is recommended to write any value to CNT register (item 3). This write updates the FTM counter with the CNTIN register value and the channels output with its initial value (except for channels in output compare mode) (Counter reset).

The next step is to select the FTM counter clock by the CLKS[1:0] bits (item 4). It is important to highlight that the pins are only controlled by FTM when CLKS[1:0] bits are different from zero (See the table in the description of CnSC register).

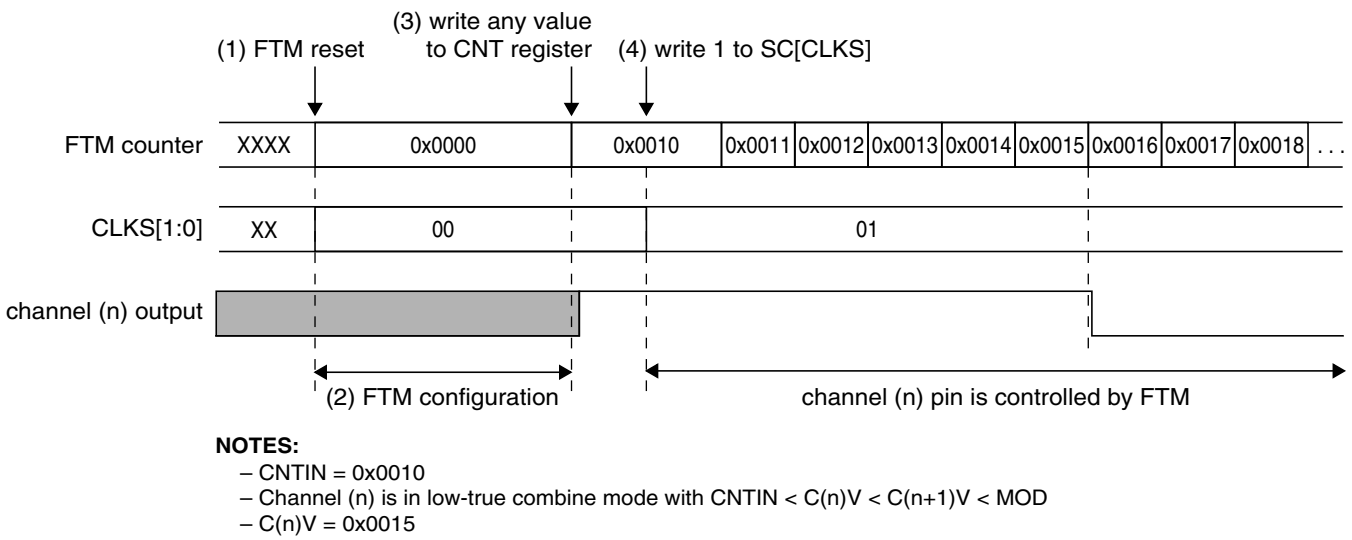



**Figure 40-300. FTM behavior after reset when the channel (n) is in Combine mode**

The following figure shows an example when the channel (n) is in Output Compare mode and the channel (n) output is toggled when there is a match. In the Output Compare mode, the channel output is not updated to its initial value when there is a write to CNT

register (item 3). In this case, use the software output control (Software output control) or the initialization (Initialization) to update the channel output to the selected value (item 4).

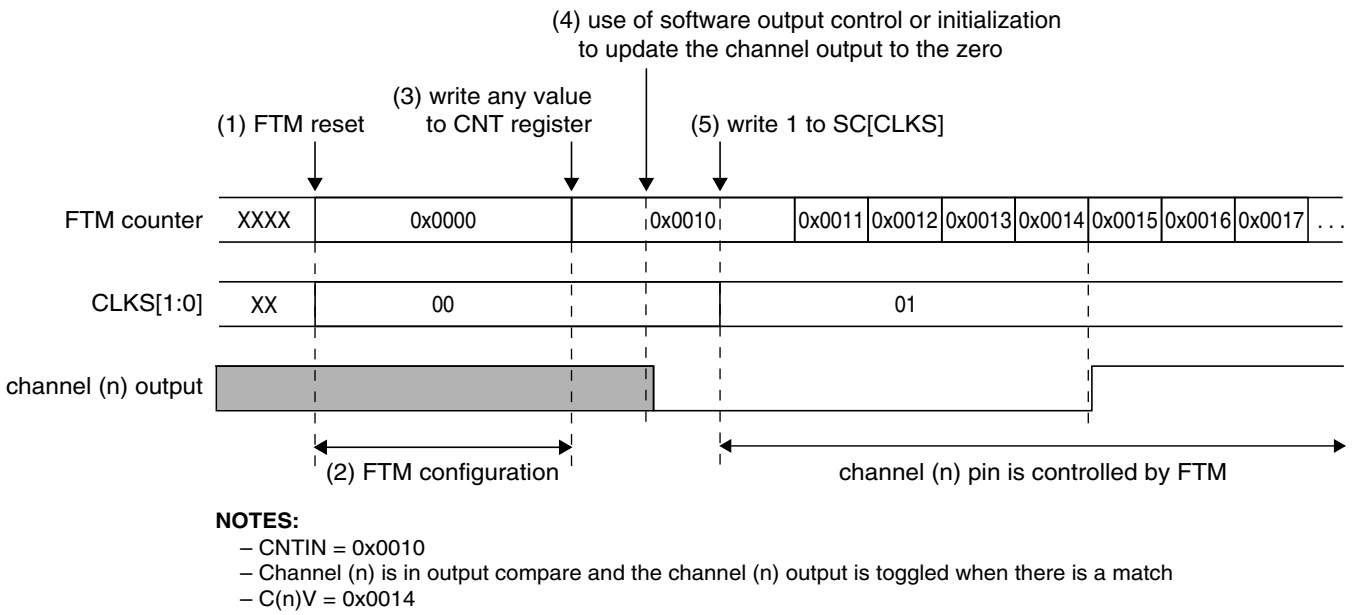



**Figure 40-301. FTM behavior after reset when the channel (n) is in Output Compare mode**

## 40.6 FTM Interrupts

### 40.6.1 Timer Overflow Interrupt

The timer overflow interrupt is generated when (TOIE = 1) and (TOF = 1).

### 40.6.2 Channel (n) Interrupt

The channel (n) interrupt is generated when (CHnIE = 1) and (CHnF = 1).

### 40.6.3 Fault Interrupt

The fault interrupt is generated when (FAULTIE = 1) and (FAULTF = 1).

# Chapter 41
# Periodic Interrupt Timer (PIT)

## 41.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The PIT module is an array of timers that can be used to raise interrupts and triggers.

### 41.1.1 Block diagram

The following figure shows the block diagram of the PIT module.

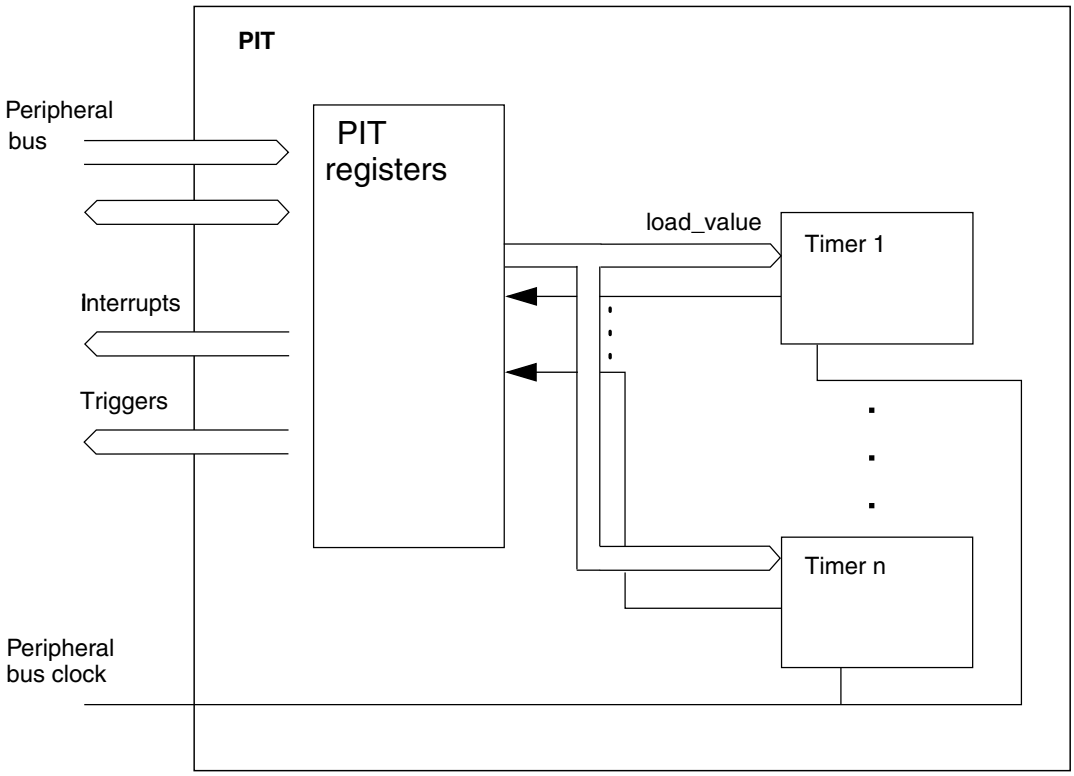



**Figure 41-1. Block diagram of the PIT**

**NOTE**

See the chip configuration details for the number of PIT channels used in this MCU.

## 41.1.2 Features

The main features of this block are:

- Ability of timers to generate trigger pulses
- Ability of timers to generate interrupts
- Maskable interrupts
- Independent timeout periods for each timer

# 41.2 Signal description

The PIT module has no external pins.

## 41.3 Memory map/register description

This section provides a detailed description of all registers accessible in the PIT module.

- Reserved registers will read as 0, writes will have no effect.
- See the chip configuration details for the number of PIT channels used in this MCU.

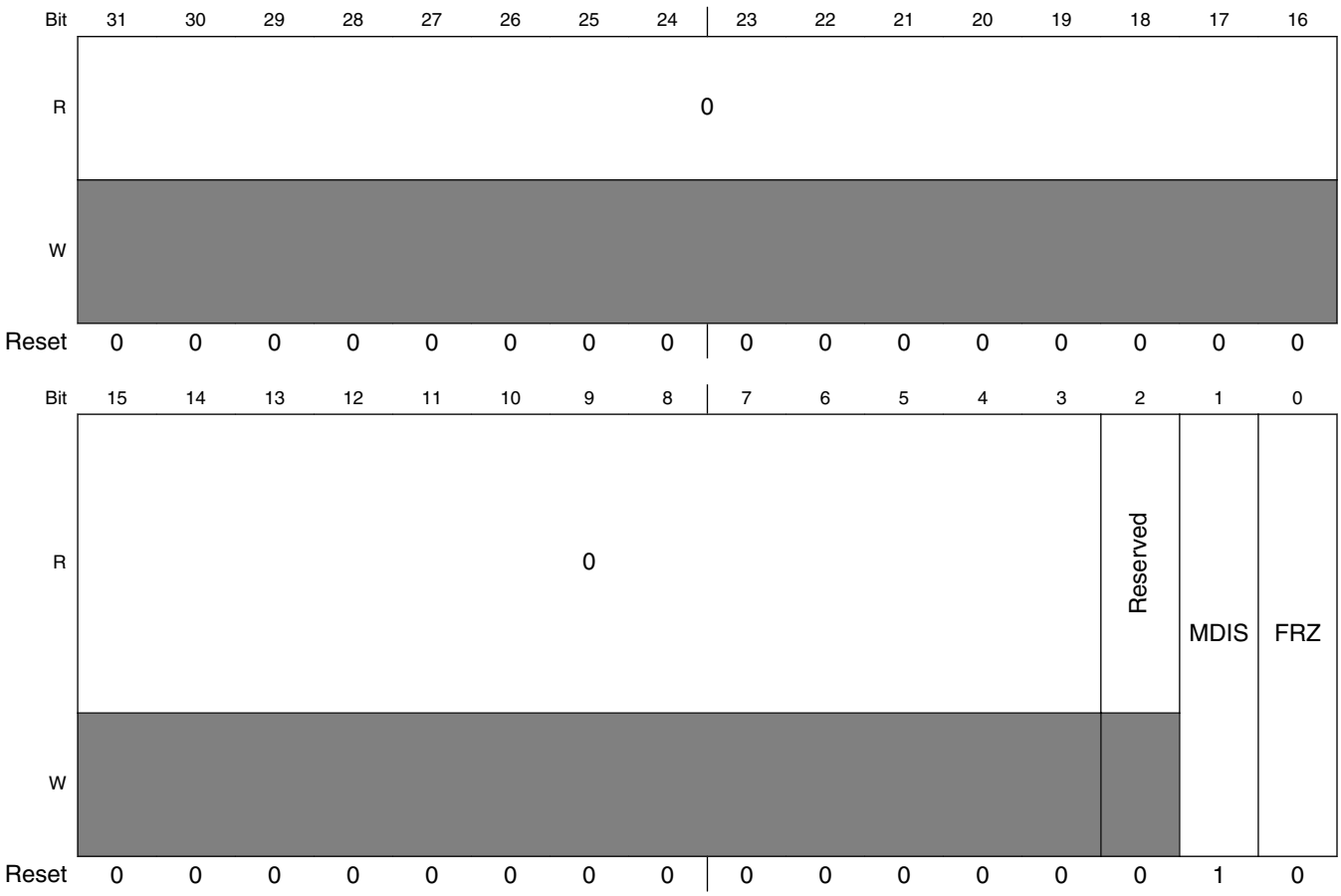
**PIT memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4003_7000 | PIT Module Control Register (PIT_MCR) | 32 | R/W | 0000_0002h | 41.3.1/1049 |
| 4003_7100 | Timer Load Value Register (PIT_LDVAL0) | 32 | R/W | 0000_0000h | 41.3.2/1051 |
| 4003_7104 | Current Timer Value Register (PIT_CVAL0) | 32 | R | 0000_0000h | 41.3.3/1051 |
| 4003_7108 | Timer Control Register (PIT_TCTRL0) | 32 | R/W | 0000_0000h | 41.3.4/1052 |
| 4003_710C | Timer Flag Register (PIT_TFLG0) | 32 | R/W | 0000_0000h | 41.3.5/1053 |
| 4003_7110 | Timer Load Value Register (PIT_LDVAL1) | 32 | R/W | 0000_0000h | 41.3.2/1051 |
| 4003_7114 | Current Timer Value Register (PIT_CVAL1) | 32 | R | 0000_0000h | 41.3.3/1051 |
| 4003_7118 | Timer Control Register (PIT_TCTRL1) | 32 | R/W | 0000_0000h | 41.3.4/1052 |
| 4003_711C | Timer Flag Register (PIT_TFLG1) | 32 | R/W | 0000_0000h | 41.3.5/1053 |
| 4003_7120 | Timer Load Value Register (PIT_LDVAL2) | 32 | R/W | 0000_0000h | 41.3.2/1051 |
| 4003_7124 | Current Timer Value Register (PIT_CVAL2) | 32 | R | 0000_0000h | 41.3.3/1051 |
| 4003_7128 | Timer Control Register (PIT_TCTRL2) | 32 | R/W | 0000_0000h | 41.3.4/1052 |
| 4003_712C | Timer Flag Register (PIT_TFLG2) | 32 | R/W | 0000_0000h | 41.3.5/1053 |
| 4003_7130 | Timer Load Value Register (PIT_LDVAL3) | 32 | R/W | 0000_0000h | 41.3.2/1051 |
| 4003_7134 | Current Timer Value Register (PIT_CVAL3) | 32 | R | 0000_0000h | 41.3.3/1051 |
| 4003_7138 | Timer Control Register (PIT_TCTRL3) | 32 | R/W | 0000_0000h | 41.3.4/1052 |
| 4003_713C | Timer Flag Register (PIT_TFLG3) | 32 | R/W | 0000_0000h | 41.3.5/1053 |

### 41.3.1 PIT Module Control Register (PIT_MCR)

This register enables or disables the PIT timer clocks and controls the timers when the PIT enters the Debug mode.

Address: 4003_7000h base + 0h offset = 4003_7000h




## PIT_MCR field descriptions

| Field | Description |
|---|---|
| 31–3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>Reserved | This field is reserved. |
| 1<br>MDIS | Module Disable - (PIT section)<br><br>Disables the standard timers. This field must be enabled before any other setup is done.<br><br>0 Clock for standard PIT timers is enabled.<br>1 Clock for standard PIT timers is disabled. |
| 0<br>FRZ | Freeze<br><br>Allows the timers to be stopped when the device enters the Debug mode.<br><br>0 Timers continue to run in Debug mode.<br>1 Timers are stopped in Debug mode. |

## 41.3.2 Timer Load Value Register (PIT_LDVAL*n*)

These registers select the timeout period for the timer interrupts.

Address: 4003_7000h base + 100h offset + (16d × i), where i=0d to 3d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSV | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### PIT_LDVAL*n* field descriptions

| Field | Description |
|---|---|
| 31–0<br>TSV | Timer Start Value<br><br>Sets the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer; instead the value will be loaded after the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again. |

## 41.3.3 Current Timer Value Register (PIT_CVAL*n*)

These registers indicate the current timer position.

Address: 4003_7000h base + 104h offset + (16d × i), where i=0d to 3d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TVL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### PIT_CVAL*n* field descriptions

| Field | Description |
|---|---|
| 31–0<br>TVL | Current Timer Value<br><br>Represents the current timer value, if the timer is enabled.<br><br>**NOTE:**<br>• If the timer is disabled, do not use this field as its value is unreliable.<br>• The timer uses a downcounter. The timer values are frozen in Debug mode if MCR[FRZ] is set. |

## 41.3.4 Timer Control Register (PIT_TCTRL*n*)

These register contain the control bits for each timer.

Address: 4003_7000h base + 108h offset + (16d × i), where i=0d to 3d

| Bit | 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | CHN | TIE | TEN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PIT_TCTRL*n* field descriptions**

| Field | Description |
|---|---|
| 31–3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>CHN | Chain Mode<br>When activated, Timer n-1 needs to expire before timer n can decrement by 1.<br>Timer 0 cannot be chained.<br>0 Timer is not chained.<br>1 Timer is chained to previous timer. For example, for Channel 2, if this field is set, Timer 2 is chained to Timer 1. |
| 1<br>TIE | Timer Interrupt Enable<br>When an interrupt is pending, or, TFLGn[TIF] is set, enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TFLGn[TIF] must be cleared first.<br>0 Interrupt requests from Timer n are disabled.<br>1 Interrupt will be requested whenever TIF is set. |
| 0<br>TEN | Timer Enable<br>Enables or disables the timer.<br>0 Timer n is disabled.<br>1 Timer n is enabled. |

### 41.3.5 Timer Flag Register (PIT_TFLG*n*)

These registers hold the PIT interrupt flags.

Address: 4003_7000h base + 10Ch offset + (16d × i), where i=0d to 3d

| Bit | 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | TIF |
| W | | | | | | | | | | | | | | | | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PIT_TFLG*n* field descriptions**

| Field | Description |
|---|---|
| 31–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>TIF | Timer Interrupt Flag<br><br>Sets to 1 at the end of the timer period. Writing 1 to this flag clears it. Writing 0 has no effect. If enabled, or, when TCTRLn[TIE] = 1, TIF causes an interrupt request.<br><br>0 Timeout has not yet occurred.<br>1 Timeout has occurred. |

## 41.4 Functional description

This section provides the functional description of the module.

### 41.4.1 General operation

This section gives detailed information on the internal operation of the module. Each timer can be used to generate trigger pulses and interrupts. Each interrupt is available on a separate interrupt line.

### 41.4.1.1 Timers

The timers generate triggers at periodic intervals, when enabled. The timers load the start values as specified in their LDVAL registers, count down to 0 and then load the respective start value again. Each time a timer reaches 0, it will generate a trigger pulse and set the interrupt flag.

All interrupts can be enabled or masked by setting TCTRLn[TIE]. A new interrupt can be generated only after the previous one is cleared.

If desired, the current counter value of the timer can be read via the CVAL registers.

The counter period can be restarted, by first disabling, and then enabling the timer with TCTRLn[TEN]. See the following figure.

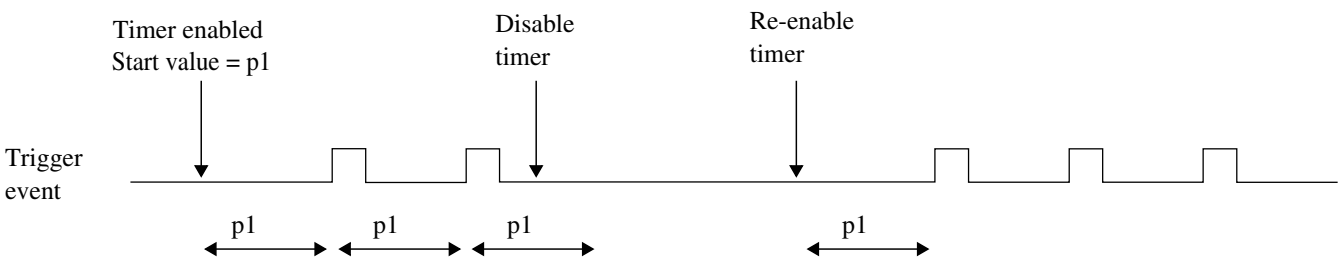



**Figure 41-23. Stopping and starting a timer**

The counter period of a running timer can be modified, by first disabling the timer, setting a new load value, and then enabling the timer again. See the following figure.

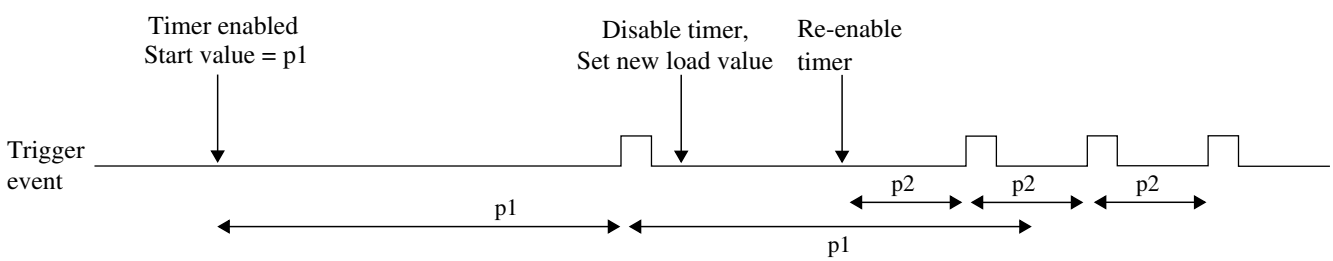



**Figure 41-24. Modifying running timer period**

It is also possible to change the counter period without restarting the timer by writing LDVAL with the new load value. This value will then be loaded after the next trigger event. See the following figure.

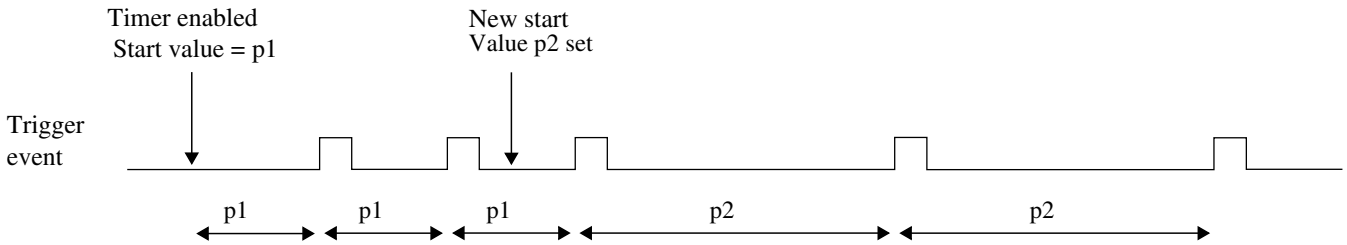



**Figure 41-25. Dynamically setting a new load value**

#### 41.4.1.2 Debug mode

In Debug mode, the timers will be frozen based on MCR[FRZ]. This is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system, for example, the timer values, and then continue the operation.

### 41.4.2 Interrupts

All the timers support interrupt generation. See the MCU specification for related vector addresses and priorities.

Timer interrupts can be enabled by setting TCTRLn[TIE]. TFLGn[TIF] are set to 1 when a timeout occurs on the associated timer, and are cleared to 0 by writing a 1 to the corresponding TFLGn[TIF].

### 41.4.3 Chained timers

When a timer has chain mode enabled, it will only count after the previous timer has expired. So if timer n-1 has counted down to 0, counter n will decrement the value by one. This allows to chain some of the timers together to form a longer timer. The first timer (timer 0) cannot be chained to any other timer.

## 41.5 Initialization and application information

In the example configuration:

- The PIT clock has a frequency of 50 MHz.

- Timer 1 creates an interrupt every 5.12 ms.
- Timer 3 creates a trigger event every 30 ms.

The PIT module must be activated by writing a 0 to MCR[MDIS].

The 50 MHz clock frequency equates to a clock period of 20 ns. Timer 1 needs to trigger every 5.12 ms/20 ns = 256,000 cycles and Timer 3 every 30 ms/20 ns = 1,500,000 cycles. The value for the LDVAL register trigger is calculated as:

LDVAL trigger = (period / clock period) -1

This means LDVAL1 and LDVAL3 must be written with 0x0003E7FF and 0x0016E35F respectively.

The interrupt for Timer 1 is enabled by setting TCTRL1[TIE]. The timer is started by writing 1 to TCTRL1[TEN].

Timer 3 shall be used only for triggering. Therefore, Timer 3 is started by writing a 1 to TCTRL3[TEN]. TCTRL3[TIE] stays at 0.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// Timer 1
PIT_LDVAL1 = 0x0003E7FF; // setup timer 1 for 256000 cycles
PIT_TCTRL1 = TIE; // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN; // start Timer 1


// Timer 3
PIT_LDVAL3 = 0x0016E35F; // setup timer 3 for 1500000 cycles
PIT_TCTRL3 |= TEN; // start Timer 3
```

## 41.6 Example configuration for chained timers

In the example configuration:

- The PIT clock has a frequency of 100 MHz.
- Timers 1 and 2 are available.
- An interrupt shall be raised every 1 hour.

The PIT module needs to be activated by writing a 0 to MCR[MDIS].

The 100 MHz clock frequency equates to a clock period of 10 ns, so the PIT needs to count for 6000 million cycles, which is more than a single timer can do. So, Timer 1 is set up to trigger every 6 s (600 million cycles). Timer 2 is chained to Timer 1 and programmed to trigger 10 times.

The value for the LDVAL register trigger is calculated as number of cycles-1, so LDVAL1 receives the value 0x23C345FF and LDVAL2 receives the value 0x00000009.

The interrupt for Timer 2 is enabled by setting TCTRL2[TIE], the Chain mode is activated by setting TCTRL2[CHN], and the timer is started by writing a 1 to TCTRL2[TEN]. TCTRL1[TEN] needs to be set, and TCTRL1[CHN] and TCTRL1[TIE] are cleared.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// Timer 2
PIT_LDVAL2 = 0x00000009; // setup Timer 2 for 10 counts
PIT_TCTRL2 = TIE; // enable Timer 2 interrupt
PIT_TCTRL2 |= CHN; // chain Timer 2 to Timer 1
PIT_TCTRL2 |= TEN; // start Timer 2


// Timer 1
PIT_LDVAL1 = 0x23C345FF; // setup Timer 1 for 600 000 000 cycles
PIT_TCTRL1 = TEN; // start Timer 1
```

# Chapter 42
# Low-Power Timer (LPTMR)

## 42.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The low-power timer (LPTMR) can be configured to operate as a time counter with optional prescaler, or as a pulse counter with optional glitch filter, across all power modes, including the low-leakage modes. It can also continue operating through most system reset events, allowing it to be used as a time of day counter.

### 42.1.1 Features

The features of the LPTMR module include:

- 16-bit time counter or pulse counter with compare
  - Optional interrupt can generate asynchronous wakeup from any low-power mode
  - Hardware trigger output
  - Counter supports free-running mode or reset on compare
- Configurable clock source for prescaler/glitch filter
- Configurable input source for pulse counter
  - Rising-edge or falling-edge

### 42.1.2 Modes of operation

The following table describes the operation of the LPTMR module in various modes.

**Table 42-1. Modes of operation**

| Modes | Description |
|---|---|
| Run | The LPTMR operates normally. |
| Wait | The LPTMR continues to operate normally and may be configured to exit the low-power mode by generating an interrupt request. |
| Stop | The LPTMR continues to operate normally and may be configured to exit the low-power mode by generating an interrupt request. |
| Low-Leakage | The LPTMR continues to operate normally and may be configured to exit the low-power mode by generating an interrupt request. |
| Debug | The LPTMR operates normally in Pulse Counter mode, but counter does not increment in Time Counter mode. |

## 42.2 LPTMR signal descriptions

**Table 42-2. LPTMR signal descriptions**

| Signal | I/O | Description |
|---|---|---|
| LPTMR_ALT*n* | I | Pulse Counter Input pin |

### 42.2.1 Detailed signal descriptions

**Table 42-3. LPTMR interface—detailed signal descriptions**

| Signal | I/O | Description | |
|---|---|---|---|
| LPTMR_ALT*n* | I | Pulse Counter Input<br>The LPTMR can select one of the input pins to be used in Pulse Counter mode. | |
| | | State meaning | Assertion—If configured for pulse counter mode with active-high input, then assertion causes the CNR to increment.<br>Deassertion—If configured for pulse counter mode with active-low input, then deassertion causes the CNR to increment. |
| | | Timing | Assertion or deassertion may occur at any time; input may assert asynchronously to the bus clock. |

## 42.3 Memory map and register definition

**NOTE**

The LPTMR registers are reset only on a POR or LVD event.
See LPTMR power and reset for more details.

### LPTMR memory map

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4004_0000 | Low Power Timer Control Status Register (LPTMR0_CSR) | 32 | R/W | 0000_0000h | 42.3.1/1061 |
| 4004_0004 | Low Power Timer Prescale Register (LPTMR0_PSR) | 32 | R/W | 0000_0000h | 42.3.2/1063 |
| 4004_0008 | Low Power Timer Compare Register (LPTMR0_CMR) | 32 | R/W | 0000_0000h | 42.3.3/1064 |
| 4004_000C | Low Power Timer Counter Register (LPTMR0_CNR) | 32 | R | 0000_0000h | 42.3.4/1065 |

## 42.3.1 Low Power Timer Control Status Register (LPTMRx_CSR)

Address: 4004_0000h base + 0h offset = 4004_0000h

| Bit | 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | TCF | TIE | TPS | | TPP | TFC | TMS | TEN |
| W | | | | | | | | | w1c | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### LPTMRx_CSR field descriptions

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>TCF | Timer Compare Flag<br><br>TCF is set when the LPTMR is enabled and the CNR equals the CMR and increments. TCF is cleared when the LPTMR is disabled or a logic 1 is written to it.<br><br>0 The value of CNR is not equal to CMR and increments.<br>1 The value of CNR is equal to CMR and increments. |
| 6<br>TIE | Timer Interrupt Enable<br><br>When TIE is set, the LPTMR Interrupt is generated whenever TCF is also set. |

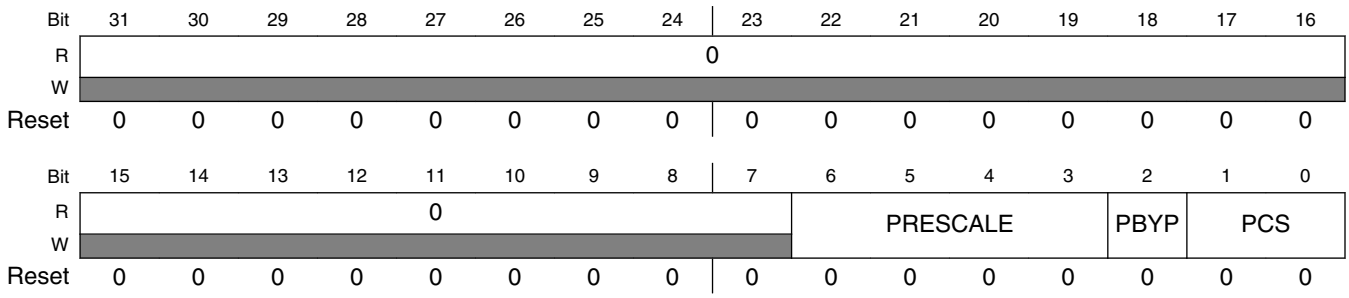
*Table continues on the next page...*

## LPTMRx_CSR field descriptions (continued)

| Field | Description |
|---|---|
| | 0 Timer interrupt disabled.<br>1 Timer interrupt enabled. |
| 5–4<br>TPS | Timer Pin Select<br><br>Configures the input source to be used in Pulse Counter mode. TPS must be altered only when the LPTMR is disabled. The input connections vary by device. See the chip configuration details for information on the connections to these inputs.<br><br>00 Pulse counter input 0 is selected.<br>01 Pulse counter input 1 is selected.<br>10 Pulse counter input 2 is selected.<br>11 Pulse counter input 3 is selected. |
| 3<br>TPP | Timer Pin Polarity<br><br>Configures the polarity of the input source in Pulse Counter mode. TPP must be changed only when the LPTMR is disabled.<br><br>0 Pulse Counter input source is active-high, and the CNR will increment on the rising-edge.<br>1 Pulse Counter input source is active-low, and the CNR will increment on the falling-edge. |
| 2<br>TFC | Timer Free-Running Counter<br><br>When clear, TFC configures the CNR to reset whenever TCF is set. When set, TFC configures the CNR to reset on overflow. TFC must be altered only when the LPTMR is disabled.<br><br>0 CNR is reset whenever TCF is set.<br>1 CNR is reset on overflow. |
| 1<br>TMS | Timer Mode Select<br><br>Configures the mode of the LPTMR. TMS must be altered only when the LPTMR is disabled.<br><br>0 Time Counter mode.<br>1 Pulse Counter mode. |
| 0<br>TEN | Timer Enable<br><br>When TEN is clear, it resets the LPTMR internal logic, including the CNR and TCF. When TEN is set, the LPTMR is enabled. While writing 1 to this field, CSR[5:1] must not be altered.<br><br>0 LPTMR is disabled and internal logic is reset.<br>1 LPTMR is enabled. |

## 42.3.2 Low Power Timer Prescale Register (LPTMRx_PSR)

Address: 4004_0000h base + 4h offset = 4004_0004h




### LPTMRx_PSR field descriptions

| Field | Description |
|---|---|
| 31–7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6–3<br>PRESCALE | Prescale Value<br><br>Configures the size of the Prescaler in Time Counter mode or width of the glitch filter in Pulse Counter mode. PRESCALE must be altered only when the LPTMR is disabled.<br><br>0000 Prescaler divides the prescaler clock by 2; glitch filter does not support this configuration.<br>0001 Prescaler divides the prescaler clock by 4; glitch filter recognizes change on input pin after 2 rising clock edges.<br>0010 Prescaler divides the prescaler clock by 8; glitch filter recognizes change on input pin after 4 rising clock edges.<br>0011 Prescaler divides the prescaler clock by 16; glitch filter recognizes change on input pin after 8 rising clock edges.<br>0100 Prescaler divides the prescaler clock by 32; glitch filter recognizes change on input pin after 16 rising clock edges.<br>0101 Prescaler divides the prescaler clock by 64; glitch filter recognizes change on input pin after 32 rising clock edges.<br>0110 Prescaler divides the prescaler clock by 128; glitch filter recognizes change on input pin after 64 rising clock edges.<br>0111 Prescaler divides the prescaler clock by 256; glitch filter recognizes change on input pin after 128 rising clock edges.<br>1000 Prescaler divides the prescaler clock by 512; glitch filter recognizes change on input pin after 256 rising clock edges.<br>1001 Prescaler divides the prescaler clock by 1024; glitch filter recognizes change on input pin after 512 rising clock edges.<br>1010 Prescaler divides the prescaler clock by 2048; glitch filter recognizes change on input pin after 1024 rising clock edges.<br>1011 Prescaler divides the prescaler clock by 4096; glitch filter recognizes change on input pin after 2048 rising clock edges.<br>1100 Prescaler divides the prescaler clock by 8192; glitch filter recognizes change on input pin after 4096 rising clock edges.<br>1101 Prescaler divides the prescaler clock by 16,384; glitch filter recognizes change on input pin after 8192 rising clock edges. |

*Table continues on the next page...*

**LPTMRx_PSR field descriptions (continued)**

| Field | Description |
|---|---|
| | 1110 Prescaler divides the prescaler clock by 32,768; glitch filter recognizes change on input pin after 16,384 rising clock edges.<br>1111 Prescaler divides the prescaler clock by 65,536; glitch filter recognizes change on input pin after 32,768 rising clock edges. |
| 2<br>PBYP | Prescaler Bypass<br><br>When PBYP is set, the selected prescaler clock in Time Counter mode or selected input source in Pulse Counter mode directly clocks the CNR. When PBYP is clear, the CNR is clocked by the output of the prescaler/glitch filter. PBYP must be altered only when the LPTMR is disabled.<br><br>0 Prescaler/glitch filter is enabled.<br>1 Prescaler/glitch filter is bypassed. |
| 1–0<br>PCS | Prescaler Clock Select<br><br>Selects the clock to be used by the LPTMR prescaler/glitch filter. PCS must be altered only when the LPTMR is disabled. The clock connections vary by device.<br><br>**NOTE:** See the chip configuration details for information on the connections to these inputs.<br><br>00 Prescaler/glitch filter clock 0 selected.<br>01 Prescaler/glitch filter clock 1 selected.<br>10 Prescaler/glitch filter clock 2 selected.<br>11 Prescaler/glitch filter clock 3 selected. |

## 42.3.3 Low Power Timer Compare Register (LPTMRx_CMR)

Address: 4004_0000h base + 8h offset = 4004_0008h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | COMPARE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**LPTMRx_CMR field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–0<br>COMPARE | Compare Value<br><br>When the LPTMR is enabled and the CNR equals the value in the CMR and increments, TCF is set and the hardware trigger asserts until the next time the CNR increments. If the CMR is 0, the hardware trigger will remain asserted until the LPTMR is disabled. If the LPTMR is enabled, the CMR must be altered only when TCF is set. |

### 42.3.4 Low Power Timer Counter Register (LPTMRx_CNR)

Address: 4004_0000h base + Ch offset = 4004_000Ch

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | COUNTER | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**LPTMRx_CNR field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–0<br>COUNTER | Counter Value |

## 42.4 Functional description

### 42.4.1 LPTMR power and reset

The LPTMR remains powered in all power modes, including low-leakage modes. If the LPTMR is not required to remain operating during a low-power mode, then it must be disabled before entering the mode.

The LPTMR is reset only on global Power On Reset (POR) or Low Voltage Detect (LVD). When configuring the LPTMR registers, the CSR must be initially written with the timer disabled, before configuring the PSR and CMR. Then, CSR[TIE] must be set as the last step in the initialization. This ensures the LPTMR is configured correctly and the LPTMR counter is reset to zero following a warm reset.

### 42.4.2 LPTMR clocking

The LPTMR prescaler/glitch filter can be clocked by one of the four clocks. The clock source must be enabled before the LPTMR is enabled.

**NOTE**

The clock source selected may need to be configured to remain enabled in low-power modes, otherwise the LPTMR will not operate during low-power modes.

In Pulse Counter mode with the prescaler/glitch filter bypassed, the selected input source directly clocks the CNR and no other clock source is required. To minimize power in this case, configure the prescaler clock source for a clock that is not toggling.

**NOTE**

The clock source or pulse input source selected for the LPTMR should not exceed the frequency $f_{LPTMR}$ defined in the device datasheet.

## 42.4.3 LPTMR prescaler/glitch filter

The LPTMR prescaler and glitch filter share the same logic which operates as a prescaler in Time Counter mode and as a glitch filter in Pulse Counter mode.

**NOTE**

The prescaler/glitch filter configuration must not be altered when the LPTMR is enabled.

### 42.4.3.1 Prescaler enabled

In Time Counter mode, when the prescaler is enabled, the output of the prescaler directly clocks the CNR. When the LPTMR is enabled, the CNR will increment every $2^2$ to $2^{16}$ prescaler clock cycles. After the LPTMR is enabled, the first increment of the CNR will take an additional one or two prescaler clock cycles due to synchronization logic.

### 42.4.3.2 Prescaler bypassed

In Time Counter mode, when the prescaler is bypassed, the selected prescaler clock increments the CNR on every clock cycle. When the LPTMR is enabled, the first increment will take an additional one or two prescaler clock cycles due to synchronization logic.

#### 42.4.3.3 Glitch filter

In Pulse Counter mode, when the glitch filter is enabled, the output of the glitch filter directly clocks the CNR. When the LPTMR is first enabled, the output of the glitch filter is asserted, that is, logic 1 for active-high and logic 0 for active-low. The following table shows the change in glitch filter output with the selected input source.

| If | Then |
|---|---|
| The selected input source remains deasserted for at least $2^1$ to $2^{15}$ consecutive prescaler clock rising edges | The glitch filter output will also deassert. |
| The selected input source remains asserted for at least $2^1$ to $2^{15}$ consecutive prescaler clock rising-edges | The glitch filter output will also assert. |

**NOTE**

The input is only sampled on the rising clock edge.

The CNR will increment each time the glitch filter output asserts. In Pulse Counter mode, the maximum rate at which the CNR can increment is once every $2^2$ to $2^{16}$ prescaler clock edges. When first enabled, the glitch filter will wait an additional one or two prescaler clock edges due to synchronization logic.

#### 42.4.3.4 Glitch filter bypassed

In Pulse Counter mode, when the glitch filter is bypassed, the selected input source increments the CNR every time it asserts. Before the LPTMR is first enabled, the selected input source is forced to be asserted. This prevents the CNR from incrementing if the selected input source is already asserted when the LPTMR is first enabled.

### 42.4.4 LPTMR compare

When the CNR equals the value of the CMR and increments, the following events occur:

- CSR[TCF] is set.
- LPTMR interrupt is generated if CSR[TIE] is also set.
- LPTMR hardware trigger is generated.
- CNR is reset if CSR[TFC] is clear.

When the LPTMR is enabled, the CMR can be altered only when CSR[TCF] is set. When updating the CMR, the CMR must be written and CSR[TCF] must be cleared before the LPTMR counter has incremented past the new LPTMR compare value.

### 42.4.5 LPTMR counter

The CNR increments by one on every:

- Prescaler clock in Time Counter mode with prescaler bypassed
- Prescaler output in Time Counter mode with prescaler enabled
- Input source assertion in Pulse Counter mode with glitch filter bypassed
- Glitch filter output in Pulse Counter mode with glitch filter enabled

The CNR is reset when the LPTMR is disabled or if the counter register overflows. If CSR[TFC] is cleared, then the CNR is also reset whenever CSR[TCF] is set.

The CNR continues incrementing when the core is halted in Debug mode when configured for Pulse Counter mode, the CNR will stop incrementing when the core is halted in Debug mode when configured for Time Counter mode.

The CNR cannot be initialized, but can be read at any time. On each read of the CNR, software must first write to the CNR with any value. This will synchronize and register the current value of the CNR into a temporary register. The contents of the temporary register are returned on each read of the CNR.

When reading the CNR, the bus clock must be at least two times faster than the rate at which the LPTMR counter is incrementing, otherwise incorrect data may be returned.

### 42.4.6 LPTMR hardware trigger

The LPTMR hardware trigger asserts at the same time the CSR[TCF] is set and can be used to trigger hardware events in other peripherals without software intervention. The hardware trigger is always enabled.

| When | Then |
|---|---|
| The CMR is set to 0 with CSR[TFC] clear | The LPTMR hardware trigger will assert on the first compare and does not deassert. |
| The CMR is set to a nonzero value, or, if CSR[TFC] is set | The LPTMR hardware trigger will assert on each compare and deassert on the following increment of the CNR. |

### 42.4.7 LPTMR interrupt

The LPTMR interrupt is generated whenever CSR[TIE] and CSR[TCF] are set. CSR[TCF] is cleared by disabling the LPTMR or by writing a logic 1 to it.

CSR[TIE] can be altered and CSR[TCF] can be cleared while the LPTMR is enabled.

The LPTMR interrupt is generated asynchronously to the system clock and can be used to generate a wakeup from any low-power mode, including the low-leakage modes, provided the LPTMR is enabled as a wakeup source.

# Chapter 43
# Carrier Modulator Transmitter (CMT)

## 43.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The carrier modulator transmitter (CMT) module provides the means to generate the protocol timing and carrier signals for a wide variety of encoding schemes. The CMT incorporates hardware to off-load the critical and/or lengthy timing requirements associated with signal generation from the CPU, releasing much of its bandwidth to handle other tasks such as:

- Code data generation
- Data decompression, or,
- Keyboard scanning

. The CMT does not include dedicated hardware configurations for specific protocols, but is intended to be sufficiently programmable in its function to handle the timing requirements of most protocols with minimal CPU intervention.

When the modulator is disabled, certain CMT registers can be used to change the state of the infrared output (IRO) signal directly. This feature allows for the generation of future protocol timing signals not readily producible by the current architecture.

## 43.2 Features

The features of this module include:

- Four modes of operation:
  - Time; with independent control of high and low times

    - Baseband
    - Frequency-shift key (FSK)
    - Direct software control of the IRO signal
- Extended space operation in Time, Baseband, and FSK modes
- Selectable input clock divider
- Interrupt on end-of-cycle
    - Ability to disable the IRO signal and use as timer interrupt

## 43.3 Block diagram

The following figure presents the block diagram of the CMT module.

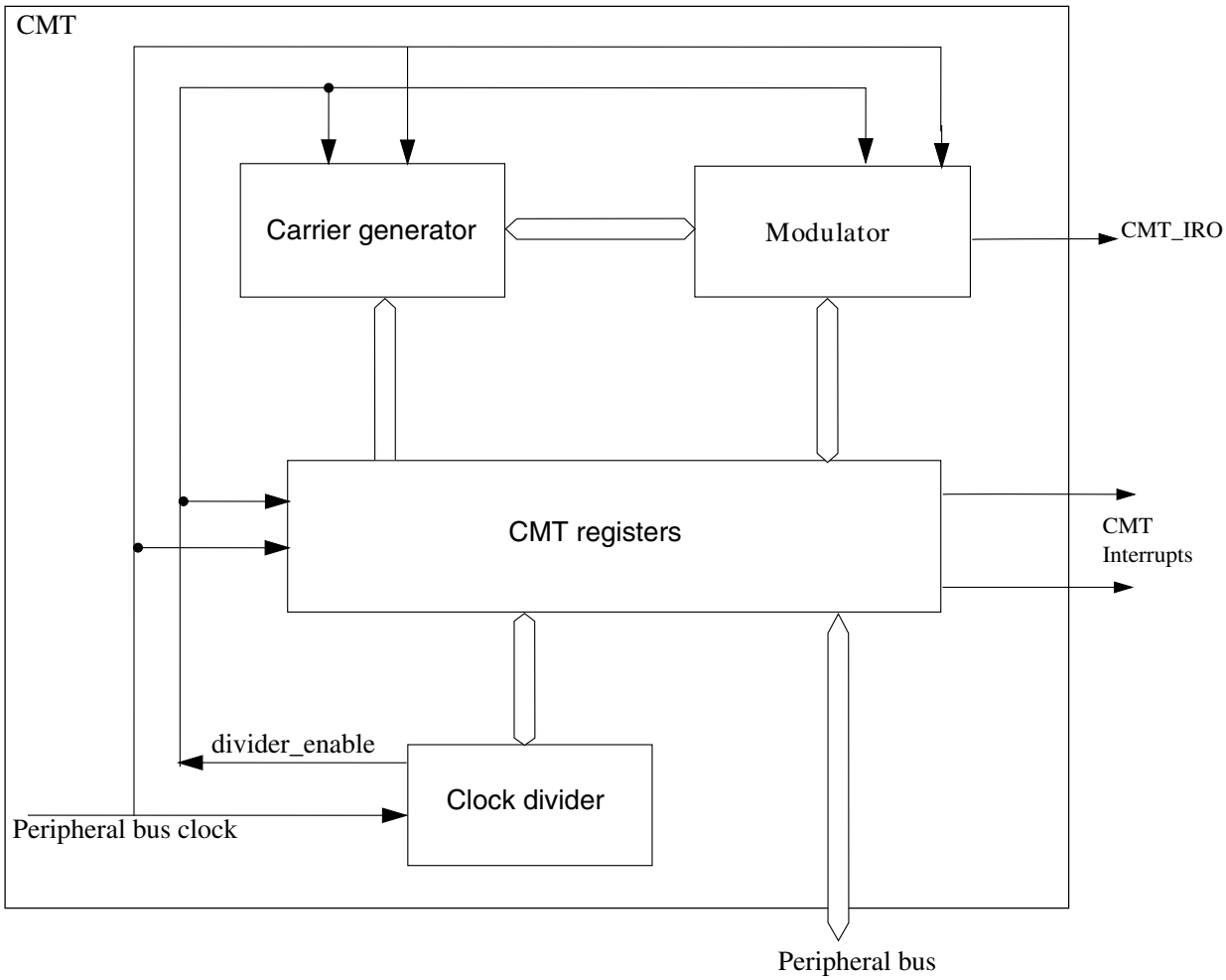



**Figure 43-1. CMT module block diagram**

## 43.4 Modes of operation

The following table describes the operation of the CMT module operates in various modes.

**Table 43-1. Modes of operation**

| Modes | Description |
|---|---|
| Time | In Time mode, the user independently defines the high and low times of the carrier signal to determine both period and duty cycle |
| Baseband | When MSC[BASE] is set, the carrier output ($f_{cg}$) to the modulator is held high continuously to allow for the generation of baseband protocols. |

*Table continues on the next page...*

**Table 43-1. Modes of operation (continued)**

| Modes | Description |
|---|---|
| Frequency-shift key | This mode allows the carrier generator to alternate between two sets of high and low times. When operating in FSK mode, the generator will toggle between the two sets when instructed by the modulator, allowing the user to dynamically switch between two carrier frequencies without CPU intervention. |

The following table summarizes the modes of operation of the CMT module.

**Table 43-2. CMT modes of operation**

| Mode | MSC[MCGEN][1] | MSC[BASE][2] | MSC[FSK][2] | MSC[EXSPC] | Comment |
|---|---|---|---|---|---|
| Time | 1 | 0 | 0 | 0 | $f_{cg}$ controlled by primary high and low registers.<br>$f_{cg}$ transmitted to the IRO signal when modulator gate is open. |
| Baseband | 1 | 1 | X | 0 | $f_{cg}$ is always high. The IRO signal is high when the modulator gate is open. |
| FSK | 1 | 0 | 1 | 0 | $f_{cg}$ control alternates between primary high/low registers and secondary high/low registers.<br>$f_{cg}$ transmitted to the IRO signal when modulator gate is open. |
| Extended Space | 1 | X | X | 1 | Setting MSC[EXSPC] causes subsequent modulator cycles to be spaces (modulator out not asserted) for the duration of the modulator period (mark and space times). |
| IRO Latch | 0 | X | X | X | OC[IROL] controls the state of the IRO signal. |

1. To prevent spurious operation, initialize all data and control registers before beginning a transmission when MSC[MCGEN]=1.
2. This field is not double-buffered and must not be changed during a transmission while MSC[MCGEN]=1.

**NOTE**

The assignment of module modes to core modes is chip-specific. For module-to-core mode assignments, see the chapter that describes how modules are configured.

## 43.4.1 Wait mode operation

During Wait mode, the CMT if enabled, will continue to operate normally. However, there is no change in operating modes of CMT during Wait mode, because the CPU is not operating.

### 43.4.2 Stop mode operation

This section describes the CMT Stop mode operations.

#### 43.4.2.1 Normal Stop mode operation

During Normal Stop mode, clocks to the CMT module are halted. No registers are affected.

The CMT module will resume upon exit from Normal Stop mode because the clocks are halted. Software must ensure that the Normal Stop mode is not entered while the modulator is still in operation so as to prevent the IRO signal from being asserted while in Normal Stop mode. This may require a timeout period from the time that MSC[MCGEN] is cleared to allow the last modulator cycle to complete.

#### 43.4.2.2 Low-Power Stop mode operation

During Low-Power Stop mode, the CMT module is completely powered off internally and the IRO signal state is latched and held at the time when the CMT enters this mode. To prevent the IRO signal from being asserted during Low-Power Stop mode, the software must assure that the signal is not active when entering Low-Power Stop mode. Upon wakeup from Low-Power Stop mode, the CMT module will be in the reset state.

## 43.5 CMT external signal descriptions

The following table shows the description of the external signal.

**Table 43-3. CMT signal description**

| Signal | Description | I/O |
|---|---|---|
| CMT_IRO | Infrared Output | O |

### 43.5.1 CMT_IRO — Infrared Output

This output signal is driven by the modulator output when MSC[MCGEN] and OC[IROPEN] are set. The IRO signal starts a valid transmission with a delay, after MSC[MCGEN] bit be asserted to high, that can be calculated based on two register bits. Table 43-5 shows how to calculate this delay.

The following table describes conditions for the IRO signal to be active.

| If | Then |
|---|---|
| MSC[MCGEN] is cleared and OC[IROPEN] is set | The signal is driven by OC[IROL] . This enables user software to directly control the state of the IRO signal by writing to OC[IROL] . |
| OC[IROPEN] is cleared | The signal is disabled and is not driven by the CMT module. Therefore, CMT can be configured as a modulo timer for generating periodic interrupts without causing signal activity. |

**Table 43-5. CMT_IRO signal delay calculation**

| Condition | Delay (bus clock cycles) |
|---|---|
| MSC[CMTDIV] = 0 | PPS[PPSDIV] + 2 |
| MSC[CMTDIV] > 0 | (PPS[PPSDIV] *2) + 3 |

## 43.6 Memory map/register definition

The following registers control and monitor the CMT operation.

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.

**CMT memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4006_2000 | CMT Carrier Generator High Data Register 1 (CMT_CGH1) | 8 | R/W | Undefined | 43.6.1/1077 |
| 4006_2001 | CMT Carrier Generator Low Data Register 1 (CMT_CGL1) | 8 | R/W | Undefined | 43.6.2/1078 |
| 4006_2002 | CMT Carrier Generator High Data Register 2 (CMT_CGH2) | 8 | R/W | Undefined | 43.6.3/1078 |
| 4006_2003 | CMT Carrier Generator Low Data Register 2 (CMT_CGL2) | 8 | R/W | Undefined | 43.6.4/1079 |
| 4006_2004 | CMT Output Control Register (CMT_OC) | 8 | R/W | 00h | 43.6.5/1079 |
| 4006_2005 | CMT Modulator Status and Control Register (CMT_MSC) | 8 | R/W | 00h | 43.6.6/1080 |
| 4006_2006 | CMT Modulator Data Register Mark High (CMT_CMD1) | 8 | R/W | Undefined | 43.6.7/1082 |

*Table continues on the next page...*

**CMT memory map (continued)**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4006_2007 | CMT Modulator Data Register Mark Low (CMT_CMD2) | 8 | R/W | Undefined | 43.6.8/1083 |
| 4006_2008 | CMT Modulator Data Register Space High (CMT_CMD3) | 8 | R/W | Undefined | 43.6.9/1083 |
| 4006_2009 | CMT Modulator Data Register Space Low (CMT_CMD4) | 8 | R/W | Undefined | 43.6.10/ 1084 |
| 4006_200A | CMT Primary Prescaler Register (CMT_PPS) | 8 | R/W | 00h | 43.6.11/ 1084 |
| 4006_200B | CMT Direct Memory Access Register (CMT_DMA) | 8 | R/W | 00h | 43.6.12/ 1085 |

## 43.6.1 CMT Carrier Generator High Data Register 1 (CMT_CGH1)

This data register contains the primary high value for generating the carrier output.

Address: 4006_2000h base + 0h offset = 4006_2000h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read Write | PH | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**CMT_CGH1 field descriptions**

| Field | Description |
|---|---|
| 7–0 PH | Primary Carrier High Time Data Value<br><br>Contains the number of input clocks required to generate the carrier high time period. When operating in Time mode, this register is always selected. When operating in FSK mode, this register and the secondary register pair are alternately selected under the control of the modulator. The primary carrier high time value is undefined out of reset. This register must be written to nonzero values before the carrier generator is enabled to avoid spurious results. |

## 43.6.2 CMT Carrier Generator Low Data Register 1 (CMT_CGL1)

This data register contains the primary low value for generating the carrier output.

Address: 4006_2000h base + 1h offset = 4006_2001h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | PL | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**CMT_CGL1 field descriptions**

| Field | Description |
|---|---|
| 7–0<br>PL | Primary Carrier Low Time Data Value<br><br>Contains the number of input clocks required to generate the carrier low time period. When operating in Time mode, this register is always selected. When operating in FSK mode, this register and the secondary register pair are alternately selected under the control of the modulator. The primary carrier low time value is undefined out of reset. This register must be written to nonzero values before the carrier generator is enabled to avoid spurious results. |

## 43.6.3 CMT Carrier Generator High Data Register 2 (CMT_CGH2)

This data register contains the secondary high value for generating the carrier output.

Address: 4006_2000h base + 2h offset = 4006_2002h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | SH | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**CMT_CGH2 field descriptions**

| Field | Description |
|---|---|
| 7–0<br>SH | Secondary Carrier High Time Data Value<br><br>Contains the number of input clocks required to generate the carrier high time period. When operating in Time mode, this register is never selected. When operating in FSK mode, this register and the primary register pair are alternately selected under control of the modulator. |

**CMT_CGH2 field descriptions (continued)**

| Field | Description |
|---|---|
| | The secondary carrier high time value is undefined out of reset. This register must be written to nonzero values before the carrier generator is enabled when operating in FSK mode. |

## 43.6.4 CMT Carrier Generator Low Data Register 2 (CMT_CGL2)

This data register contains the secondary low value for generating the carrier output.

Address: 4006_2000h base + 3h offset = 4006_2003h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | SL | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**CMT_CGL2 field descriptions**

| Field | Description |
|---|---|
| 7–0<br>SL | Secondary Carrier Low Time Data Value<br><br>Contains the number of input clocks required to generate the carrier low time period. When operating in Time mode, this register is never selected. When operating in FSK mode, this register and the primary register pair are alternately selected under the control of the modulator. The secondary carrier low time value is undefined out of reset. This register must be written to nonzero values before the carrier generator is enabled when operating in FSK mode. |

## 43.6.5 CMT Output Control Register (CMT_OC)

This register is used to control the IRO signal of the CMT module.

Address: 4006_2000h base + 4h offset = 4006_2004h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | IROL | CMTPOL | IROPEN | 0 | | | | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMT_OC field descriptions**

| Field | Description |
|---|---|
| 7<br>IROL | IRO Latch Control<br><br>Reads the state of the IRO latch. Writing to IROL changes the state of the IRO signal when MSC[MCGEN] is cleared and IROPEN is set. |
| 6<br>CMTPOL | CMT Output Polarity<br><br>Controls the polarity of the IRO signal.<br><br>0 The IRO signal is active-low.<br>1 The IRO signal is active-high. |
| 5<br>IROPEN | IRO Pin Enable<br><br>Enables and disables the IRO signal. When the IRO signal is enabled, it is an output that drives out either the CMT transmitter output or the state of IROL depending on whether MSC[MCGEN] is set or not. Also, the state of output is either inverted or non-inverted, depending on the state of CMTPOL. When the IRO signal is disabled, it is in a high-impedance state and is unable to draw any current. This signal is disabled during reset.<br><br>0 The IRO signal is disabled.<br>1 The IRO signal is enabled as output. |
| 4–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 43.6.6 CMT Modulator Status and Control Register (CMT_MSC)

This register contains the modulator and carrier generator enable (MCGEN), end of cycle interrupt enable (EOCIE), FSK mode select (FSK), baseband enable (BASE), extended space (EXSPC), prescaler (CMTDIV) bits, and the end of cycle (EOCF) status bit.

Address: 4006_2000h base + 5h offset = 4006_2005h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | EOCF | CMTDIV | | EXSPC | BASE | FSK | EOCIE | MCGEN |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMT_MSC field descriptions**

| Field | Description |
|---|---|
| 7<br>EOCF | End Of Cycle Status Flag<br><br>Sets when: |

*Table continues on the next page...*

## CMT_MSC field descriptions (continued)

| Field | Description |
|---|---|
| | • The modulator is not currently active and MCGEN is set to begin the initial CMT transmission.<br>• At the end of each modulation cycle while MCGEN is set. This is recognized when a match occurs between the contents of the space period register and the down counter. At this time, the counter is initialized with, possibly new contents of the mark period buffer, CMD1 and CMD2, and the space period register is loaded with, possibly new contents of the space period buffer, CMD3 and CMD4.<br><br>This flag is cleared by reading MSC followed by an access of CMD2 or CMD4, or by the DMA transfer.<br><br>0 End of modulation cycle has not occured since the flag last cleared.<br>1 End of modulator cycle has occurred. |
| 6–5<br>CMTDIV | CMT Clock Divide Prescaler<br><br>Causes the CMT to be clocked at the IF signal frequency, or the IF frequency divided by 2 ,4, or 8 . This field must not be changed during a transmission because it is not double-buffered.<br><br>00 IF ÷ 1<br>01 IF ÷ 2<br>10 IF ÷ 4<br>11 IF ÷ 8 |
| 4<br>EXSPC | Extended Space Enable<br><br>Enables the extended space operation.<br><br>0 Extended space is disabled.<br>1 Extended space is enabled. |
| 3<br>BASE | Baseband Enable<br><br>When set, BASE disables the carrier generator and forces the carrier output high for generation of baseband protocols. When BASE is cleared, the carrier generator is enabled and the carrier output toggles at the frequency determined by values stored in the carrier data registers. This field is cleared by reset. This field is not double-buffered and must not be written to during a transmission.<br><br>0 Baseband mode is disabled.<br>1 Baseband mode is enabled. |
| 2<br>FSK | FSK Mode Select<br><br>Enables FSK operation.<br><br>0 The CMT operates in Time or Baseband mode.<br>1 The CMT operates in FSK mode. |
| 1<br>EOCIE | End of Cycle Interrupt Enable<br><br>Requests to enable a CPU interrupt when EOCF is set if EOCIE is high. |

*Table continues on the next page...*

**CMT_MSC field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 CPU interrupt is disabled.<br>1 CPU interrupt is enabled. |
| 0<br>MCGEN | Modulator and Carrier Generator Enable<br><br>Setting MCGEN will initialize the carrier generator and modulator and will enable all clocks. When enabled, the carrier generator and modulator will function continuously. When MCGEN is cleared, the current modulator cycle will be allowed to expire before all carrier and modulator clocks are disabled to save power and the modulator output is forced low.<br><br>**NOTE:** To prevent spurious operation, the user should initialize all data and control registers before enabling the system.<br><br>0 Modulator and carrier generator disabled<br>1 Modulator and carrier generator enabled |

## 43.6.7 CMT Modulator Data Register Mark High (CMT_CMD1)

The contents of this register are transferred to the modulator down counter upon the completion of a modulation period.

Address: 4006_2000h base + 6h offset = 4006_2006h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | MB[15:8] | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**CMT_CMD1 field descriptions**

| Field | Description |
|---|---|
| 7–0<br>MB[15:8] | Controls the upper mark periods of the modulator for all modes. |

## 43.6.8 CMT Modulator Data Register Mark Low (CMT_CMD2)

The contents of this register are transferred to the modulator down counter upon the completion of a modulation period.

Address: 4006_2000h base + 7h offset = 4006_2007h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | MB[7:0] | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**CMT_CMD2 field descriptions**

| Field | Description |
|---|---|
| 7–0<br>MB[7:0] | Controls the lower mark periods of the modulator for all modes. |

## 43.6.9 CMT Modulator Data Register Space High (CMT_CMD3)

The contents of this register are transferred to the space period register upon the completion of a modulation period.

Address: 4006_2000h base + 8h offset = 4006_2008h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | SB[15:8] | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**CMT_CMD3 field descriptions**

| Field | Description |
|---|---|
| 7–0<br>SB[15:8] | Controls the upper space periods of the modulator for all modes. |

## 43.6.10 CMT Modulator Data Register Space Low (CMT_CMD4)

The contents of this register are transferred to the space period register upon the completion of a modulation period.

Address: 4006_2000h base + 9h offset = 4006_2009h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | SB[7:0] | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**CMT_CMD4 field descriptions**

| Field | Description |
|---|---|
| 7–0<br>SB[7:0] | Controls the lower space periods of the modulator for all modes. |

## 43.6.11 CMT Primary Prescaler Register (CMT_PPS)

This register is used to set the Primary Prescaler Divider field (PPSDIV).

Address: 4006_2000h base + Ah offset = 4006_200Ah

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | | | | PPSDIV | | | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMT_PPS field descriptions**

| Field | Description |
|---|---|
| 7–4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3–0<br>PPSDIV | Primary Prescaler Divider<br>Divides the CMT clock to generate the Intermediate Frequency clock enable to the secondary prescaler.<br>0000 Bus clock ÷ 1<br>0001 Bus clock ÷ 2<br>0010 Bus clock ÷ 3<br>0011 Bus clock ÷ 4 |

*Table continues on the next page...*

**CMT_PPS field descriptions (continued)**

| Field | Description |
|---|---|
| | 0100 Bus clock ÷ 5<br>0101 Bus clock ÷ 6<br>0110 Bus clock ÷ 7<br>0111 Bus clock ÷ 8<br>1000 Bus clock ÷ 9<br>1001 Bus clock ÷ 10<br>1010 Bus clock ÷ 11<br>1011 Bus clock ÷ 12<br>1100 Bus clock ÷ 13<br>1101 Bus clock ÷ 14<br>1110 Bus clock ÷ 15<br>1111 Bus clock ÷ 16 |

## 43.6.12 CMT Direct Memory Access Register (CMT_DMA)

This register is used to enable/disable direct memory access (DMA).

Address: 4006_2000h base + Bh offset = 4006_200Bh

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | | | | | | | DMA |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMT_DMA field descriptions**

| Field | Description |
|---|---|
| 7–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>DMA | DMA Enable<br>Enables the DMA protocol.<br>0 DMA transfer request and done are disabled.<br>1 DMA transfer request and done are enabled. |

## 43.7 Functional description

The CMT module primarily consists of clock divider, carrier generator, and modulator.

### 43.7.1 Clock divider

The CMT was originally designed to be based on an 8 MHz bus clock that could be divided by 1, 2, 4, or 8 according to the specification. To be compatible with higher bus frequency, the primary prescaler (PPS) was developed to receive a higher frequency and generate a clock enable signal called intermediate frequency (IF). This IF must be approximately equal to 8 MHz and will work as a clock enable to the secondary prescaler. The following figure shows the clock divider block diagram.

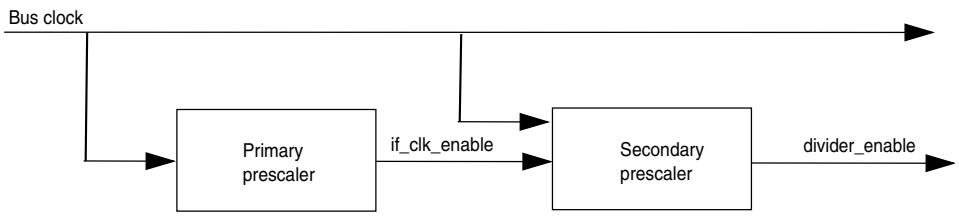



**Figure 43-14. Clock divider block diagram**

For compatibility with previous versions of CMT, when bus clock = 8 MHz, the PPS must be configured to zero. The PPS counter is selected according to the bus clock to generate an intermediate frequency approximately equal to 8 MHz.

### 43.7.2 Carrier generator

The carrier generator resolution is 125 ns when operating with an 8 MHz intermediate frequency signal and the secondary prescaler is set to divide by 1, or, when MSC[CMTDIV] = 00. The carrier generator can generate signals with periods between 250 ns (4 MHz) and 127.5 µs (7.84 kHz) in steps of 125 ns. The following table shows the relationship between the clock divide bits and the carrier generator resolution, minimum carrier generator period, and minimum modulator period.

**Table 43-19. Clock divider**

| Bus clock (MHz) | MSC[CMTDIV] | Carrier generator resolution (µs) | Min. carrier generator period (µs) | Min. modulator period (µs) |
|---|---|---|---|---|
| 8 | 00 | 0.125 | 0.25 | 1.0 |
| 8 | 01 | 0.25 | 0.5 | 2.0 |
| 8 | 10 | 0.5 | 1.0 | 4.0 |
| 8 | 11 | 1.0 | 2.0 | 8.0 |

The possible duty cycle options depend upon the number of counts required to complete the carrier period. For example, 1.6 MHz signal has a period of 625 ns and will therefore require 5 x 125 ns counts to generate. These counts may be split between high and low times, so the duty cycles available will be:

- 20% with one high and four low times
- 40% with two high and three low times
- 60% with three high and two low times, and
- 80% with four high and one low time

.

For low-frequency signals with large periods, high-resolution duty cycles as a percentage of the total period, are possible.

The carrier signal is generated by counting a register-selected number of input clocks (125 ns for an 8 MHz bus) for both the carrier high time and the carrier low time. The period is determined by the total number of clocks counted. The duty cycle is determined by the ratio of high-time clocks to total clocks counted. The high and low time values are user-programmable and are held in two registers.

An alternate set of high/low count values is held in another set of registers to allow the generation of dual-frequency FSK protocols without CPU intervention.

**Note**

Only nonzero data values are allowed. The carrier generator will not work if any of the count values are equal to zero.

MSC[MCGEN] must be set and MSC[BASE] must be cleared to enable carrier generator clocks. When MSC[BASE] is set, the carrier output to the modulator is held high continuously. The following figure represents the block diagram of the clock generator.

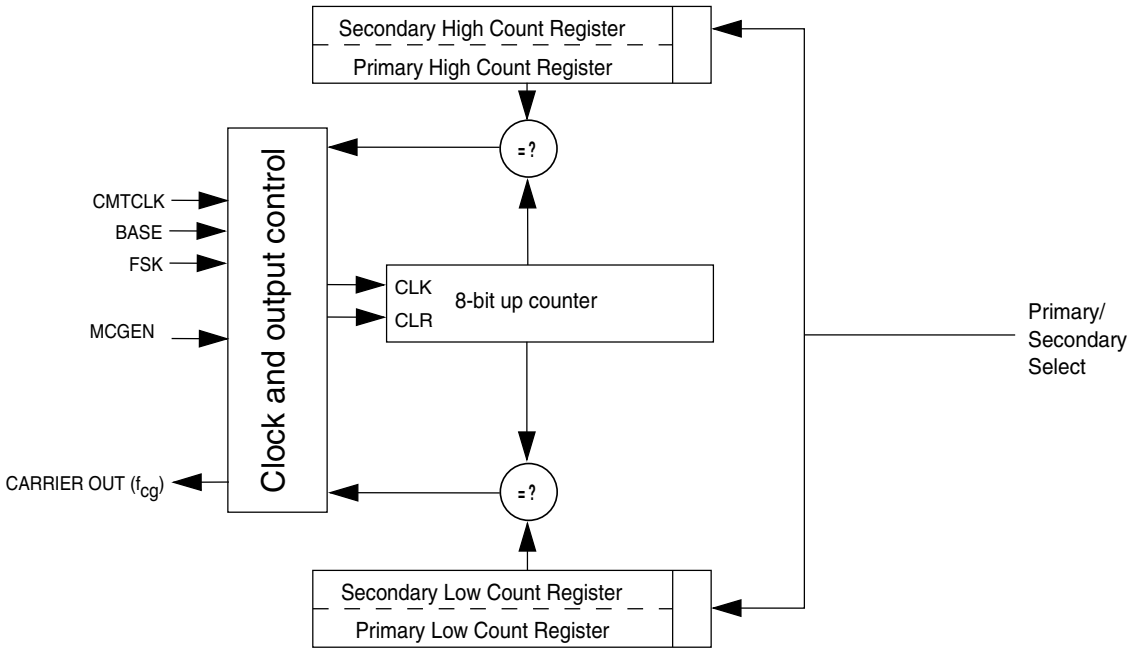



**Figure 43-15. Carrier generator block diagram**

The high/low time counter is an 8-bit up counter. After each increment, the contents of the counter are compared with the appropriate high or low count value register. When the compare value is reached, the counter is reset to a value of 0x01, and the compare is redirected to the other count value register.

Assuming that the high time count compare register is currently active, a valid compare will cause the carrier output to be driven low. The counter will continue to increment starting at the reset value of 0x01. When the value stored in the selected low count value register is reached, the counter will again be reset and the carrier output will be driven high.

The cycle repeats, automatically generating a periodic signal which is directed to the modulator. The lower frequency with maximum period, $f_{max}$, and highest frequency with minimum period, $f_{min}$, which can be generated, are defined as:

$f_{max} = f_{CMTCLK} \div (2 * 1)$ Hz

$f_{min} = f_{CMTCLK} \div (2 * (2^8 - 1))$ Hz

In the general case, the carrier generator output frequency is:

$f_{cg} = f_{CMTCLK} \div$ (High count + Low count) Hz

Where: 0 < High count < 256 and

0 < Low count < 256

The duty cycle of the carrier signal is controlled by varying the ratio of high time to low + high time. As the input clock period is fixed, the duty cycle resolution will be proportional to the number of counts required to generate the desired carrier period.

$$\text{DutyCycle} = \frac{\text{Highcount}}{\text{Highcount+Lowcount}}$$

## 43.7.3 Modulator

The modulator block controls the state of the infrared out signal (IRO). The modulator output is gated on to the IRO signal when the modulator/carrier generator is enabled. . When the modulator/carrier generator is disabled, the IRO signal is controlled by the state of the IRO latch. OC[CMTPOL] enables the IRO signal to be active-high or active-low.

The following table describes the functions of the modulators in different modes:

**Table 43-20. Mode functions**

| Mode | Function |
|---|---|
| Time | The modulator can gate the carrier onto the modulator output. |
| Baseband | The modulator can control the logic level of the modulator output. |
| FSK | The modulator can count carrier periods and instruct the carrier generator to alternate between two carrier frequencies whenever a modulation period consisting of mark and space counts, expires. |

The modulator provides a simple method to control protocol timing. The modulator has a minimum resolution of 1.0 µs with an 8 MHz. It can count bus clocks to provide real-time control, or carrier clocks for self-clocked protocols.

The modulator includes a 17-bit down counter with underflow detection. The counter is loaded from the 16-bit modulation mark period buffer registers, CMD1 and CMD2. The most significant bit is loaded with a logic 0 and serves as a sign bit.

| When | Then |
|---|---|
| The counter holds a positive value | The modulator gate is open and the carrier signal is driven to the transmitter block. |
| The counter underflows | The modulator gate is closed and a 16-bit comparator is enabled which compares the logical complement of the value of the down counter with the contents of the modulation space period register which has been loaded from the registers, CMD3 and CMD4. |

When a match is obtained, the cycle repeats by opening the modulator gate, reloading the counter with the contents of CMD1 and CMD2, and reloading the modulation space period register with the contents of CMD3 and CMD4.

The modulation space period is activated when the carrier signal is low to prohibit cutting off the high pulse of a carrier signal. If the carrier signal is high, the modulator extends the mark period until the carrier signal becomes low. To deassert the space period and assert the mark period, the carrier signal must have gone low to ensure that a space period is not erroneously shortened.

If the contents of the modulation space period register are all zeroes, the match will be immediate and no space period will be generated, for instance, for FSK protocols that require successive bursts of different frequencies).

MSC[MCGEN] must be set to enable the modulator timer.

The following figure presents the block diagram of the modulator.

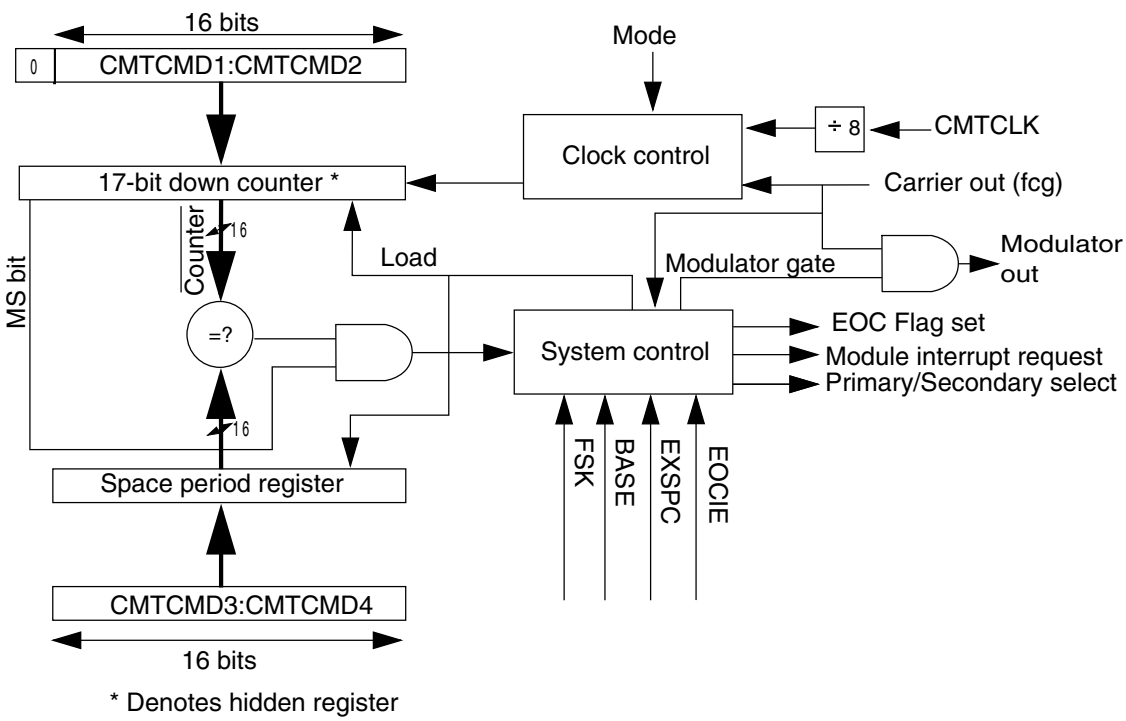



**Figure 43-16. Modulator block diagram**

### 43.7.3.1 Time mode

When the modulator operates in Time mode, or, when MSC[MCGEN] is set, and MSC[BASE] and MSC[FSK] are cleared:

- The modulation mark period consists of an integer number of (CMTCLK ÷ 8) clock periods.
- The modulation space period consists of 0 or an integer number of (CMTCLK ÷ 8) clock periods.

With an 8 MHz IF and MSC[CMTDIV] = 00, the modulator resolution is 1 µs and has a maximum mark and space period of about 65.535 ms each . See Figure 43-17 for an example of the Time and Baseband mode outputs.

The mark and space time equations for Time and Baseband mode are:

$t_{mark}$ = (CMD1:CMD2 + 1) ÷ ($f_{CMTCLK}$ ÷ 8)

$t_{space}$ = CMD3:CMD4 ÷ ($f_{CMTCLK}$ ÷ 8)

where CMD1:CMD2 and CMD3:CMD4 are the decimal values of the concatenated registers.

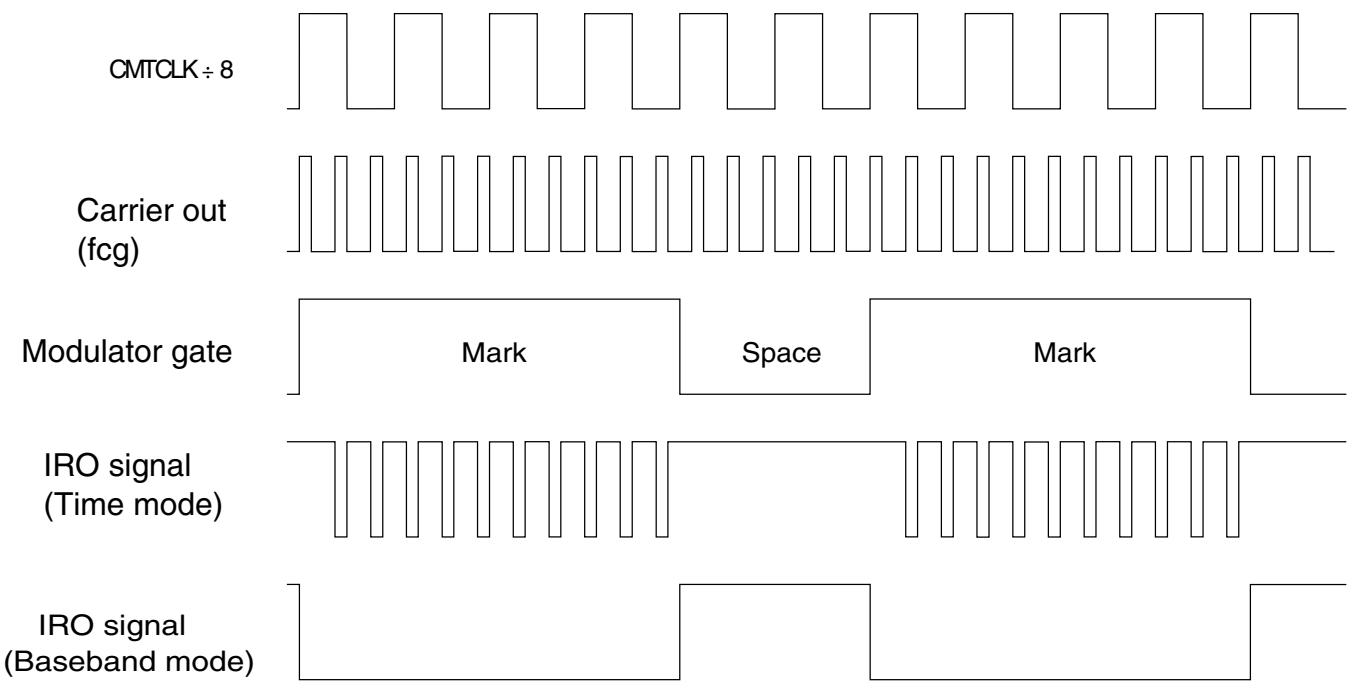



**Figure 43-17. Example: CMT output in Time and Baseband modes with OC[CMTPOL]=0**

### 43.7.3.2 Baseband mode

Baseband mode, that is, when MSC[MCGEN] and MSC[BASE] are set, is a derivative of Time mode, where the mark and space period is based on (CMTCLK ÷ 8) counts. The mark and space calculations are the same as in Time mode.

In this mode, the modulator output will be at a logic 1 for the duration of the mark period and at a logic 0 for the duration of a space period. See Figure 43-17 for an example of the output for both Baseband and Time modes. In the example, the carrier out frequency ($f_{cg}$) is generated with a high count of 0x01 and a low count of 0x02 that results in a divide of 3 of CMTCLK with a 33% duty cycle. The modulator down counter was loaded with the value 0x0003 and the space period register with 0x0002.

**Note**

The waveforms in Figure 43-17 and Figure 43-18 are for the purpose of conceptual illustration and are not meant to represent precise timing relationships between the signals shown.

#### 43.7.3.3 FSK mode

When the modulator operates in FSK mode, that is, when MSC[MCGEN] and MSC[FSK] are set, and MSC[BASE] is cleared:

- The modulation mark and space periods consist of an integer number of carrier clocks (space period can be zero).
- When the mark period expires, the space period is transparently started as in Time mode.
- The carrier generator toggles between primary and secondary data register values whenever the modulator space period expires.

The space period provides an interpulse gap (no carrier). If CMD3:CMD4 = 0x0000, then the modulator and carrier generator will switch between carrier frequencies without a gap or any carrier glitches (zero space).

Using timing data for carrier burst and interpulse gap length calculated by the CPU, FSK mode can automatically generate a phase-coherent, dual-frequency FSK signal with programmable burst and interburst gaps.

The mark and space time equations for FSK mode are:

$t_{mark}$ = (CMD1:CMD2 + 1) ÷ $f_{cg}$

$t_{space}$ = (CMD3:CMD4) ÷ $f_{cg}$

Where $f_{cg}$ is the frequency output from the carrier generator. The example in Figure 43-18 shows what the IRO signal looks like in FSK mode with the following values:

- CMD1:CMD2 = 0x0003
- CMD3:CMD4 = 0x0002
- Primary carrier high count = 0x01
- Primary carrier low count = 0x02

- Secondary carrier high count = 0x03
- Secondary carrier low count = 0x01

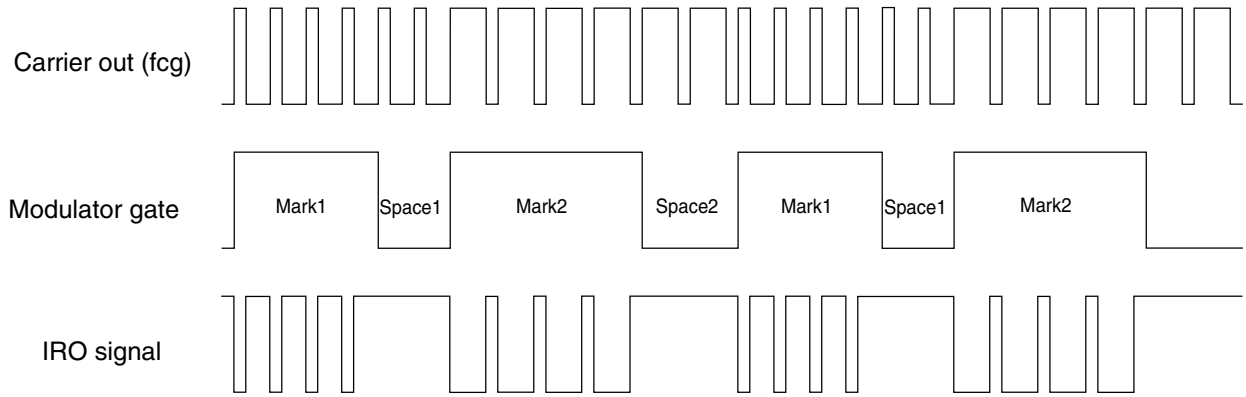



**Figure 43-18. Example: CMT output in FSK mode**

## 43.7.4 Extended space operation

In either Time, Baseband, or FSK mode, the space period can be made longer than the maximum possible value of the space period register. Setting MSC[EXSPC] will force the modulator to treat the next modulation period beginning with the next load of the counter and space period register, as a space period equal in length to the mark and space counts combined. Subsequent modulation periods will consist entirely of these extended space periods with no mark periods. Clearing MSC[EXSPC] will return the modulator to standard operation at the beginning of the next modulation period.

### 43.7.4.1 EXSPC operation in Time mode

To calculate the length of an extended space in Time or Baseband mode, add the mark and space times and multiply by the number of modulation periods when MSC[EXSPC] is set.

$t_{exspace} = (t_{mark} + t_{space})$ * (number of modulation periods)

For an example of extended space operation, see Figure 43-19.

**Note**

The extended space enable feature can be used to emulate a zero mark event.

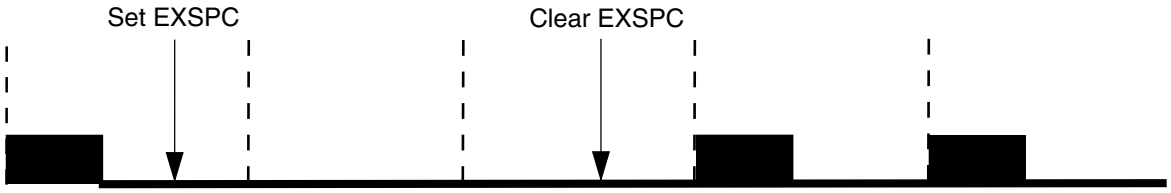



**Figure 43-19. Extended space operation**

### 43.7.4.2 EXSPC operation in FSK mode

In FSK mode, the modulator continues to count carrier out clocks, alternating between the primary and secondary registers at the end of each modulation period.

To calculate the length of an extended space in FSK mode, it is required to know whether MSC[EXSPC] was set on a primary or secondary modulation period, and the total number of both primary and secondary modulation periods completed while MSC[EXSPC] is high. A status bit for the current modulation is not accessible to the CPU. If necessary, software must maintain tracking of the current primary or secondary modulation cycle. The extended space period ends at the completion of the space period time of the modulation period during which MSC[EXSPC]is cleared.

The following table depicts the equations which can be used to calculate the extended space period depending on when MSC[EXSPC] is set.

| If | Then |
|---|---|
| MSC[EXSPC] was set during a primary modulation cycle | Use the equation:<br>$t_{exspace} = (t_{space})_p + (t_{mark} + t_{space})_s + (t_{mark} + t_{space})_p + ...$ |
| MSC[EXSPC] bit was set during a secondary modulation cycle | Use the equation:<br>$t_{exspace} = (t_{space})_s + (t_{mark} + t_{space})_p + (t_{mark} + t_{space})_s + ...$ |

Where the subscripts p and s refer to mark and space times for the primary and secondary modulation cycles.

## 43.8 CMT interrupts and DMA

The CMT generates an interrupt request or a DMA transfer request according to MSC[EOCIE], MSC[EOCF], DMA[DMA] bits.

**Table 43-23. DMA transfer request x CMT interrupt request**

| MSC[EOCF] | DMA[DMA] | MSC[EOCIE] | DMA transfer request | CMT interrupt request |
|---|---|---|---|---|
| 0 | X | X | 0 | 0 |
| 1 | X | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

MSC[EOCF] is set:

- When the modulator is not currently active and MSC[MCGEN] is set to begin the initial CMT transmission.
- At the end of each modulation cycle when the counter is reloaded from CMD1:CMD2, while MSC[MCGEN] is set.

When MSC[MCGEN] is cleared and then set before the end of the modulation cycle, MSC[EOCF] will not be set when MSC[MCGEN] is set, but will become set at the end of the current modulation cycle.

When MSC[MCGEN] becomes disabled, the CMT module does not set MSC[EOCF] at the end of the last modulation cycle.

If MSC[EOCIE] is high when MSC[EOCF] is set, the CMT module will generate an interrupt request or a DMA transfer request.

MSC[EOCF] must be cleared to prevent from being generated by another event like interrupt or DMA request, after exiting the service routine. See the following table.

**Table 43-24. How to clear MSC[EOCF]**

| DMA[DMA] | MSC[EOCIE] | Description |
|---|---|---|
| 0 | X | MSC[EOCF] is cleared by reading MSC followed by an access of CMD2 or CMD4. |
| 1 | X | MSC[EOCF] is cleared by the CMT DMA transfer done. |

The EOC interrupt is coincident with:

- Loading the down-counter with the contents of CMD1:CMD2
- Loading the space period register with the contents of CMD3:CMD4

The EOC interrupt provides a means for the user to reload new mark/space values into the modulator data registers. Modulator data register updates will take effect at the end of the current modulation cycle.

**NOTE**

The down-counter and space period register are updated at the end of every modulation cycle, irrespective of interrupt handling and the state of MSC[EOCF].

# Chapter 44
# Real Time Clock (RTC)

## 44.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

### 44.1.1 Features

The RTC module features include:

- Independent power supply, POR, and 32 kHz crystal oscillator
- 32-bit seconds counter with roll-over protection and 32-bit alarm
- 16-bit prescaler with compensation that can correct errors between 0.12 ppm and 3906 ppm
- Register write protection
  - Lock register requires VBAT POR or software reset to enable write access
  - Access control registers require system reset to enable read and/or write access
- 1 Hz square wave output
- 64-bit monotonic counter with roll-over protection
- Tamper time seconds register that records when the time was invalidated

## 44.1.2 Modes of operation

The RTC remains functional in all low power modes and can generate an interrupt to exit any low power mode. It operates in one of two modes of operation: chip power-up and chip power-down.

During chip power-down, RTC is powered from the backup power supply (VBAT) and is electrically isolated from the rest of the chip but continues to increment the time counter (if enabled) and retain the state of the RTC registers. The RTC registers are not accessible.

During chip power-up, RTC remains powered from the backup power supply (VBAT). All RTC registers are accessible by software and all functions are operational. If enabled, the 32.768 kHz clock can be supplied to the rest of the chip.

## 44.1.3 RTC Signal Descriptions

**Table 44-1. RTC signal descriptions**

| Signal | Description | I/O |
|---|---|---|
| EXTAL32 | 32.768 kHz oscillator input | I |
| XTAL32 | 32.768 kHz oscillator output | O |
| RTC_CLKOUT | 1 Hz square-wave output | O |
| $\overline{\text{RTC\_WAKEUP}}$ | Wakeup for external device | I/O |

### 44.1.3.1 RTC clock output

The clock to the seconds counter is available on the RTC_CLKOUT signal. It is a 1 Hz square wave output.

### 44.1.3.2 RTC wakeup pin

The RTC wakeup pin is an open drain, active low, output that allows the RTC to wakeup the chip via an external component. The wakeup pin asserts when the wakeup pin enable is set and either the RTC interrupt is asserted or the wakeup pin is turned on via a register bit. The wakeup pin does not assert from the RTC seconds interrupt.

The wakeup pin is optional and may not be implemented on all devices.

## 44.2 Register definition

All registers must be accessed using 32-bit writes and all register accesses incur three wait states.

Write accesses to any register and read accesses to tamper and monotonic registers by non-supervisor mode software, when the supervisor access bit in the control register is clear, will terminate with a bus error.

Read accesses to other registers by non-supervisor mode software complete as normal.

Writing to a register protected by the write access register or lock register does not generate a bus error, but the write will not complete.

Reading a register protected by the read access register does not generate a bus error, but the register will read zero.

**RTC memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4003_D000 | RTC Time Seconds Register (RTC_TSR) | 32 | R/W | 0000_0000h | 44.2.1/1100 |
| 4003_D004 | RTC Time Prescaler Register (RTC_TPR) | 32 | R/W | 0000_0000h | 44.2.2/1100 |
| 4003_D008 | RTC Time Alarm Register (RTC_TAR) | 32 | R/W | 0000_0000h | 44.2.3/1101 |
| 4003_D00C | RTC Time Compensation Register (RTC_TCR) | 32 | R/W | 0000_0000h | 44.2.4/1101 |
| 4003_D010 | RTC Control Register (RTC_CR) | 32 | R/W | 0000_0000h | 44.2.5/1102 |
| 4003_D014 | RTC Status Register (RTC_SR) | 32 | R/W | 0000_0001h | 44.2.6/1104 |
| 4003_D018 | RTC Lock Register (RTC_LR) | 32 | R/W | 0000_FFFFh | 44.2.7/1106 |
| 4003_D01C | RTC Interrupt Enable Register (RTC_IER) | 32 | R/W | 0000_0007h | 44.2.8/1108 |
| 4003_D020 | RTC Tamper Time Seconds Register (RTC_TTSR) | 32 | R | Undefined | 44.2.9/1109 |
| 4003_D024 | RTC Monotonic Enable Register (RTC_MER) | 32 | R/W | 0000_0000h | 44.2.10/ 1109 |
| 4003_D028 | RTC Monotonic Counter Low Register (RTC_MCLR) | 32 | R/W | 0000_0000h | 44.2.11/ 1110 |
| 4003_D02C | RTC Monotonic Counter High Register (RTC_MCHR) | 32 | R/W | 0000_0000h | 44.2.12/ 1110 |
| 4003_D800 | RTC Write Access Register (RTC_WAR) | 32 | R/W | 0000_FFFFh | 44.2.13/ 1111 |
| 4003_D804 | RTC Read Access Register (RTC_RAR) | 32 | R/W | 0000_FFFFh | 44.2.14/ 1113 |

## 44.2.1 RTC Time Seconds Register (RTC_TSR)

Address: 4003_D000h base + 0h offset = 4003_D000h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### RTC_TSR field descriptions

| Field | Description |
|---|---|
| 31–0<br>TSR | Time Seconds Register<br><br>When the time counter is enabled, the TSR is read only and increments once a second provided SR[TOF] or SR[TIF] are not set. The time counter will read as zero when SR[TOF] or SR[TIF] are set. When the time counter is disabled, the TSR can be read or written. Writing to the TSR when the time counter is disabled will clear the SR[TOF] and/or the SR[TIF]. Writing to TSR with zero is supported, but not recommended because TSR will read as zero when SR[TIF] or SR[TOF] are set (indicating the time is invalid). |

## 44.2.2 RTC Time Prescaler Register (RTC_TPR)

Address: 4003_D000h base + 4h offset = 4003_D004h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | TPR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### RTC_TPR field descriptions

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–0<br>TPR | Time Prescaler Register<br><br>When the time counter is enabled, the TPR is read only and increments every 32.768 kHz clock cycle. The time counter will read as zero when SR[TOF] or SR[TIF] are set. When the time counter is disabled, the TPR can be read or written. The TSR[TSR] increments when bit 14 of the TPR transitions from a logic one to a logic zero. |

## 44.2.3 RTC Time Alarm Register (RTC_TAR)

Address: 4003_D000h base + 8h offset = 4003_D008h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TAR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### RTC_TAR field descriptions

| Field | Description |
|---|---|
| 31–0<br>TAR | Time Alarm Register<br><br>When the time counter is enabled, the SR[TAF] is set whenever the TAR[TAR] equals the TSR[TSR] and the TSR[TSR] increments. Writing to the TAR clears the SR[TAF]. |

## 44.2.4 RTC Time Compensation Register (RTC_TCR)

Address: 4003_D000h base + Ch offset = 4003_D00Ch

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CIC | | | | | | | | TCV | | | | | | | | CIR | | | | | | | | TCR | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### RTC_TCR field descriptions

| Field | Description |
|---|---|
| 31–24<br>CIC | Compensation Interval Counter<br><br>Current value of the compensation interval counter. If the compensation interval counter equals zero then it is loaded with the contents of the CIR. If the CIC does not equal zero then it is decremented once a second. |
| 23–16<br>TCV | Time Compensation Value<br><br>Current value used by the compensation logic for the present second interval. Updated once a second if the CIC equals 0 with the contents of the TCR field. If the CIC does not equal zero then it is loaded with zero (compensation is not enabled for that second increment). |
| 15–8<br>CIR | Compensation Interval Register<br><br>Configures the compensation interval in seconds from 1 to 256 to control how frequently the TCR should adjust the number of 32.768 kHz cycles in each second. The value written should be one less than the number of seconds. For example, write zero to configure for a compensation interval of one second. This register is double buffered and writes do not take affect until the end of the current compensation interval. |
| 7–0<br>TCR | Time Compensation Register<br><br>Configures the number of 32.768 kHz clock cycles in each second. This register is double buffered and writes do not take affect until the end of the current compensation interval. |

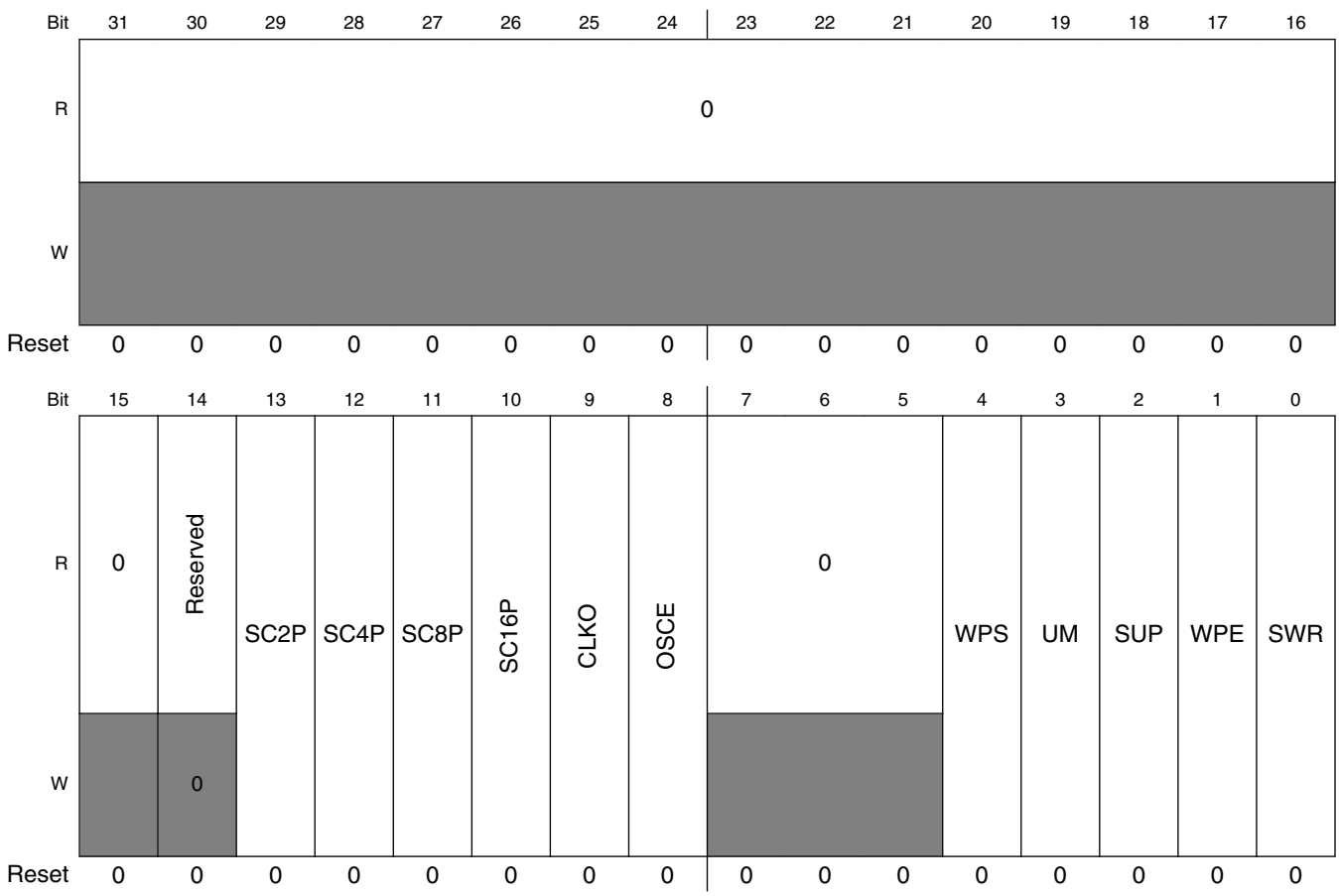
*Table continues on the next page...*

**RTC_TCR field descriptions (continued)**

| Field | Description |
|---|---|
| | 80h Time Prescaler Register overflows every 32896 clock cycles.<br>... ...<br>FFh Time Prescaler Register overflows every 32769 clock cycles.<br>00h Time Prescaler Register overflows every 32768 clock cycles.<br>01h Time Prescaler Register overflows every 32767 clock cycles.<br>... ...<br>7Fh Time Prescaler Register overflows every 32641 clock cycles. |

## 44.2.5 RTC Control Register (RTC_CR)

Address: 4003_D000h base + 10h offset = 4003_D010h




**RTC_CR field descriptions**

| Field | Description |
|---|---|
| 31–15<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

*Table continues on the next page...*

## RTC_CR field descriptions (continued)

| Field | Description |
|---|---|
| 14<br>Reserved | This field is reserved.<br>It must always be written to 0. |
| 13<br>SC2P | Oscillator 2pF Load Configure<br><br>0 Disable the load.<br>1 Enable the additional load. |
| 12<br>SC4P | Oscillator 4pF Load Configure<br><br>0 Disable the load.<br>1 Enable the additional load. |
| 11<br>SC8P | Oscillator 8pF Load Configure<br><br>0 Disable the load.<br>1 Enable the additional load. |
| 10<br>SC16P | Oscillator 16pF Load Configure<br><br>0 Disable the load.<br>1 Enable the additional load. |
| 9<br>CLKO | Clock Output<br><br>0 The 32 kHz clock is output to other peripherals.<br>1 The 32 kHz clock is not output to other peripherals. |
| 8<br>OSCE | Oscillator Enable<br><br>0 32.768 kHz oscillator is disabled.<br>1 32.768 kHz oscillator is enabled. After setting this bit, wait the oscillator startup time before enabling the time counter to allow the 32.768 kHz clock time to stabilize. |
| 7–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>WPS | Wakeup Pin Select<br><br>The wakeup pin is optional and not available on all devices.<br><br>0 Wakeup pin asserts (active low, open drain) if the RTC interrupt asserts or the wakeup pin is turned on.<br>1 Wakeup pin instead outputs the RTC 32kHz clock, provided the wakeup pin is turned on and the 32kHz clock is output to other peripherals. |
| 3<br>UM | Update Mode<br><br>Allows SR[TCE] to be written even when the Status Register is locked. When set, the SR[TCE] can always be written if the SR[TIF] or SR[TOF] are set or if the SR[TCE] is clear.<br><br>Allows the monotonic enable register to be written when it is locked. When set, the monotonic enable register can always be written if the SR[TIF] or SR[MOF] are set or if the montonic counter enable is clear.<br><br>0 Registers cannot be written when locked.<br>1 Registers can be written when locked under limited conditions. |
| 2<br>SUP | Supervisor Access<br><br>Configures non-supervisor mode write access to all RTC registers and non-supervisor mode read access to RTC tamper/monotonic registers |

*Table continues on the next page...*

### RTC_CR field descriptions (continued)

| Field | Description |
|---|---|
| | 0 Non-supervisor mode write accesses are not supported and generate a bus error.<br>1 Non-supervisor mode write accesses are supported. |
| 1<br>WPE | Wakeup Pin Enable<br><br>The wakeup pin is optional and not available on all devices.<br><br>0 Wakeup pin is disabled.<br>1 Wakeup pin is enabled and wakeup pin asserts if the RTC interrupt asserts or the wakeup pin is turned on. |
| 0<br>SWR | Software Reset<br><br>0 No effect.<br>1 Resets all RTC registers except for the SWR bit and the RTC_WAR and RTC_RAR registers . The SWR bit is cleared by VBAT POR and by software explicitly clearing it. |

## 44.2.6 RTC Status Register (RTC_SR)

Address: 4003_D000h base + 14h offset = 4003_D014h

| Bit | 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | TCE | MOF | TAF | TOF | TIF |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### RTC_SR field descriptions

| Field | Description |
|---|---|
| 31–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>TCE | Time Counter Enable<br><br>When time counter is disabled the TSR register and TPR register are writeable, but do not increment. When time counter is enabled the TSR register and TPR register are not writeable, but increment.<br><br>0 Time counter is disabled.<br>1 Time counter is enabled. |
| 3<br>MOF | Monotonic Overflow Flag<br><br>Monotonic overflow flag is set when the monotonic counter is enabled and the monotonic counter high overflows. The monotonic counter does not increment and will read as zero when this bit is set. This bit is cleared by writing the monotonic counter high register when the monotonic counter is disabled. |

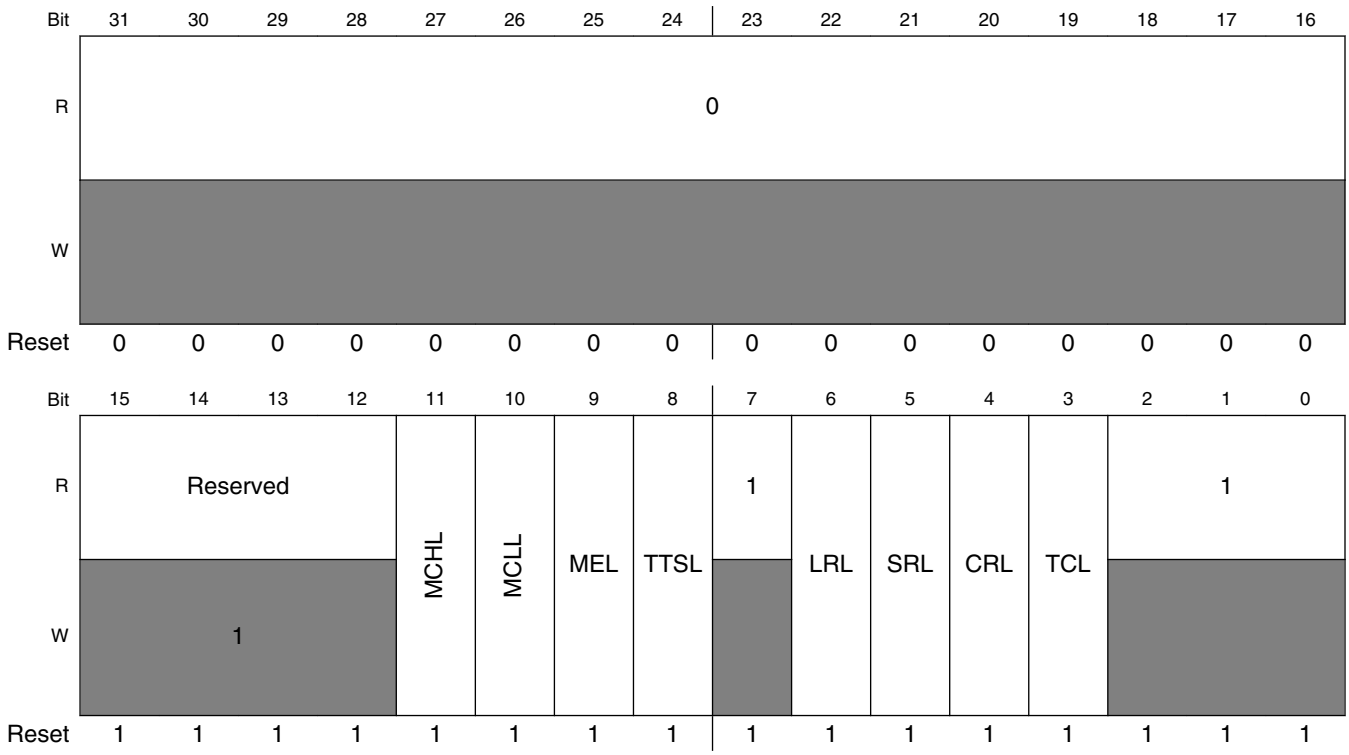
*Table continues on the next page...*

## RTC_SR field descriptions (continued)

| Field | Description |
|---|---|
| | 0 Monotonic counter overflow has not occurred.<br>1 Monotonic counter overflow has occurred and monotonic counter is read as zero. |
| 2<br>TAF | Time Alarm Flag<br><br>Time alarm flag is set when the TAR[TAR] equals the TSR[TSR] and the TSR[TSR] increments. This bit is cleared by writing the TAR register.<br><br>0 Time alarm has not occurred.<br>1 Time alarm has occurred. |
| 1<br>TOF | Time Overflow Flag<br><br>Time overflow flag is set when the time counter is enabled and overflows. The TSR and TPR do not increment and read as zero when this bit is set. This bit is cleared by writing the TSR register when the time counter is disabled.<br><br>0 Time overflow has not occurred.<br>1 Time overflow has occurred and time counter is read as zero. |
| 0<br>TIF | Time Invalid Flag<br><br>The time invalid flag is set on VBAT POR or software reset. The TSR and TPR do not increment and read as zero when this bit is set. This bit is cleared by writing the TSR register when the time counter is disabled.<br><br>The monotonic counter register is held in reset whenever the time invalid flag is set.<br><br>0 Time is valid.<br>1 Time is invalid and time counter is read as zero. |

## 44.2.7 RTC Lock Register (RTC_LR)

Address: 4003_D000h base + 18h offset = 4003_D018h




### RTC_LR field descriptions

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–12<br>Reserved | This field is reserved. |
| 11<br>MCHL | Monotonic Counter High Lock<br><br>After being cleared, this bit can be set only by VBAT POR or software reset.<br><br>0 Monotonic Counter High Register is locked and writes are ignored.<br>1 Monotonic Counter High Register is not locked and writes complete as normal. |
| 10<br>MCLL | Monotonic Counter Low Lock<br><br>After being cleared, this bit can be set only by VBAT POR or software reset.<br><br>0 Monotonic Counter Low Register is locked and writes are ignored.<br>1 Monotonic Counter Low Register is not locked and writes complete as normal. |
| 9<br>MEL | Monotonic Enable Lock<br><br>After being cleared, this bit can be set only by VBAT POR or software reset. |

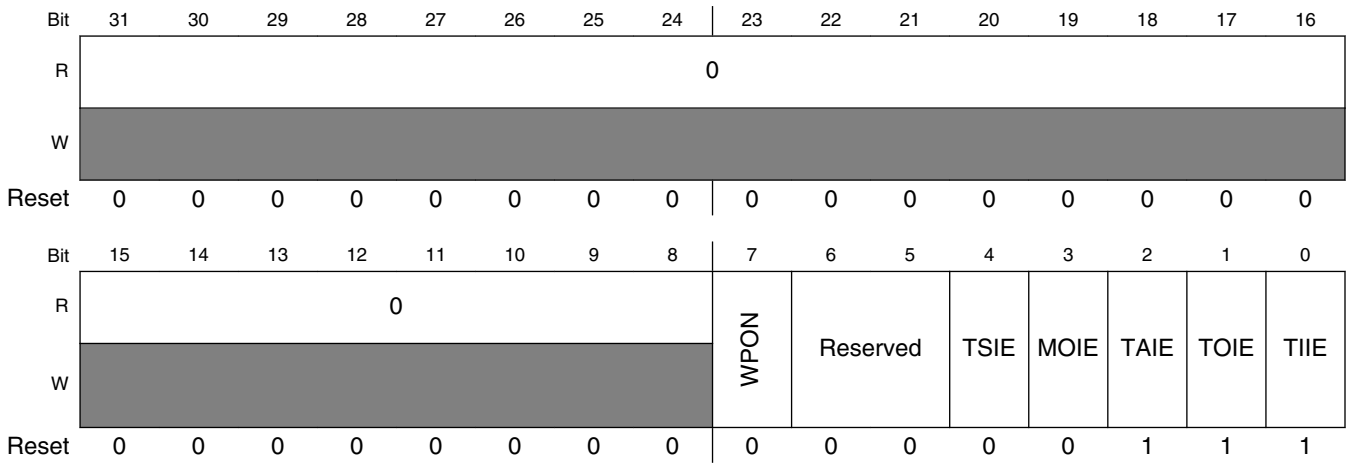
*Table continues on the next page...*

## RTC_LR field descriptions (continued)

| Field | Description |
|---|---|
| | 0 Monotonic Enable Register is locked and writes are ignored.<br>1 Monotonic Enable Register is not locked and writes complete as normal. |
| 8<br>TTSL | Tamper Time Seconds Lock<br><br>After being cleared, this bit can be set only by VBAT POR or software reset.<br><br>0 Tamper Time Seconds Register is locked and writes are ignored.<br>1 Tamper Time Seconds Register is not locked and writes complete as normal. |
| 7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 1. |
| 6<br>LRL | Lock Register Lock<br><br>After being cleared, this bit can be set only by VBAT POR or software reset.<br><br>0 Lock Register is locked and writes are ignored.<br>1 Lock Register is not locked and writes complete as normal. |
| 5<br>SRL | Status Register Lock<br><br>After being cleared, this bit can be set only by VBAT POR or software reset.<br><br>0 Status Register is locked and writes are ignored.<br>1 Status Register is not locked and writes complete as normal. |
| 4<br>CRL | Control Register Lock<br><br>After being cleared, this bit can only be set by VBAT POR.<br><br>0 Control Register is locked and writes are ignored.<br>1 Control Register is not locked and writes complete as normal. |
| 3<br>TCL | Time Compensation Lock<br><br>After being cleared, this bit can be set only by VBAT POR or software reset.<br><br>0 Time Compensation Register is locked and writes are ignored.<br>1 Time Compensation Register is not locked and writes complete as normal. |
| 2–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 1. |

## 44.2.8 RTC Interrupt Enable Register (RTC_IER)

Address: 4003_D000h base + 1Ch offset = 4003_D01Ch




### RTC_IER field descriptions

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>WPON | Wakeup Pin On<br><br>The wakeup pin is optional and not available on all devices. Whenever the wakeup pin is enabled and this bit is set, the wakeup pin will assert.<br><br>0 No effect.<br>1 If the wakeup pin is enabled, then the wakeup pin will assert. |
| 6–5<br>Reserved | This field is reserved. |
| 4<br>TSIE | Time Seconds Interrupt Enable<br><br>The seconds interrupt is an edge-sensitive interrupt with a dedicated interrupt vector. It is generated once a second and requires no software overhead (there is no corresponding status flag to clear).<br><br>0 Seconds interrupt is disabled.<br>1 Seconds interrupt is enabled. |
| 3<br>MOIE | Monotonic Overflow Interrupt Enable<br><br>0 Monotonic overflow flag does not generate an interrupt.<br>1 Monotonic overflow flag does generate an interrupt. |
| 2<br>TAIE | Time Alarm Interrupt Enable<br><br>0 Time alarm flag does not generate an interrupt.<br>1 Time alarm flag does generate an interrupt. |
| 1<br>TOIE | Time Overflow Interrupt Enable |

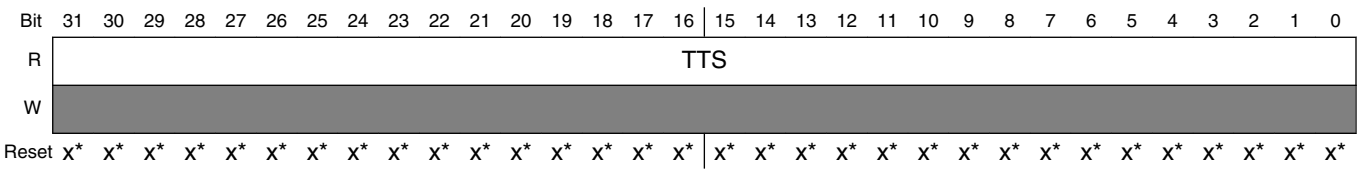
*Table continues on the next page...*

**RTC_IER field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Time overflow flag does not generate an interrupt.<br>1 Time overflow flag does generate an interrupt. |
| 0<br>TIIE | Time Invalid Interrupt Enable<br><br>0 Time invalid flag does not generate an interrupt.<br>1 Time invalid flag does generate an interrupt. |

## 44.2.9 RTC Tamper Time Seconds Register (RTC_TTSR)

Address: 4003_D000h base + 20h offset = 4003_D020h



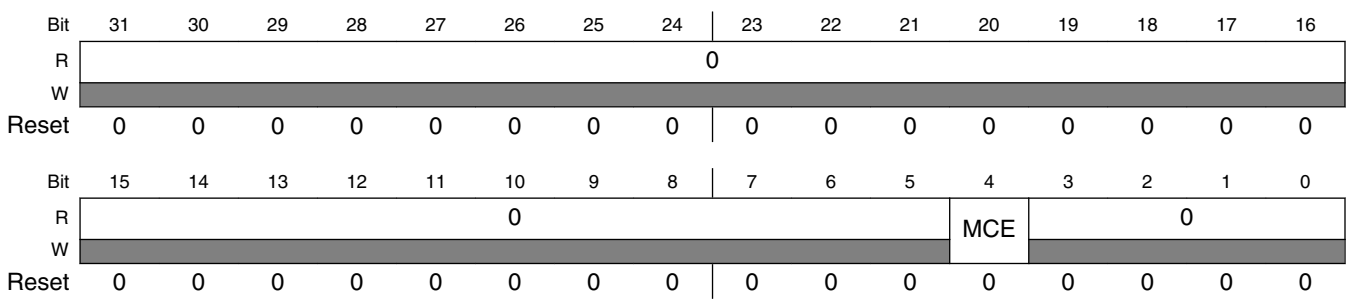

* Notes:
- x = Undefined at reset.

**RTC_TTSR field descriptions**

| Field | Description |
|---|---|
| 31–0<br>TTS | Tamper Time Seconds<br><br>If the time invalid flag is set then reading this register returns the contents of the time seconds register at the point at which the time invalid flag was set. If the time invalid flag is clear then reading this register returns zero. Writing the tamper time seconds register with any value will set the time invalid flag. |

## 44.2.10 RTC Monotonic Enable Register (RTC_MER)

Address: 4003_D000h base + 24h offset = 4003_D024h



### RTC_MER field descriptions

| Field | Description |
|---|---|
| 31–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>MCE | Monotonic Counter Enable<br><br>0 Writes to the monotonic counter load the counter with the value written.<br>1 Writes to the monotonic counter increment the counter. |
| 3–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 44.2.11 RTC Monotonic Counter Low Register (RTC_MCLR)

Address: 4003_D000h base + 28h offset = 4003_D028h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MCL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### RTC_MCLR field descriptions

| Field | Description |
|---|---|
| 31–0<br>MCL | Monotonic Counter Low<br><br>When the time invalid flag is set, the monotonic counter is held in reset. When the monotonic counter enable is clear, a write to this register will load the counter with the value written. When the monotonic counter enable is set, a write to this register will cause it to increment. A write to monotonic counter low that causes it to overflow will also increment monotonic counter high. |

## 44.2.12 RTC Monotonic Counter High Register (RTC_MCHR)

Address: 4003_D000h base + 2Ch offset = 4003_D02Ch

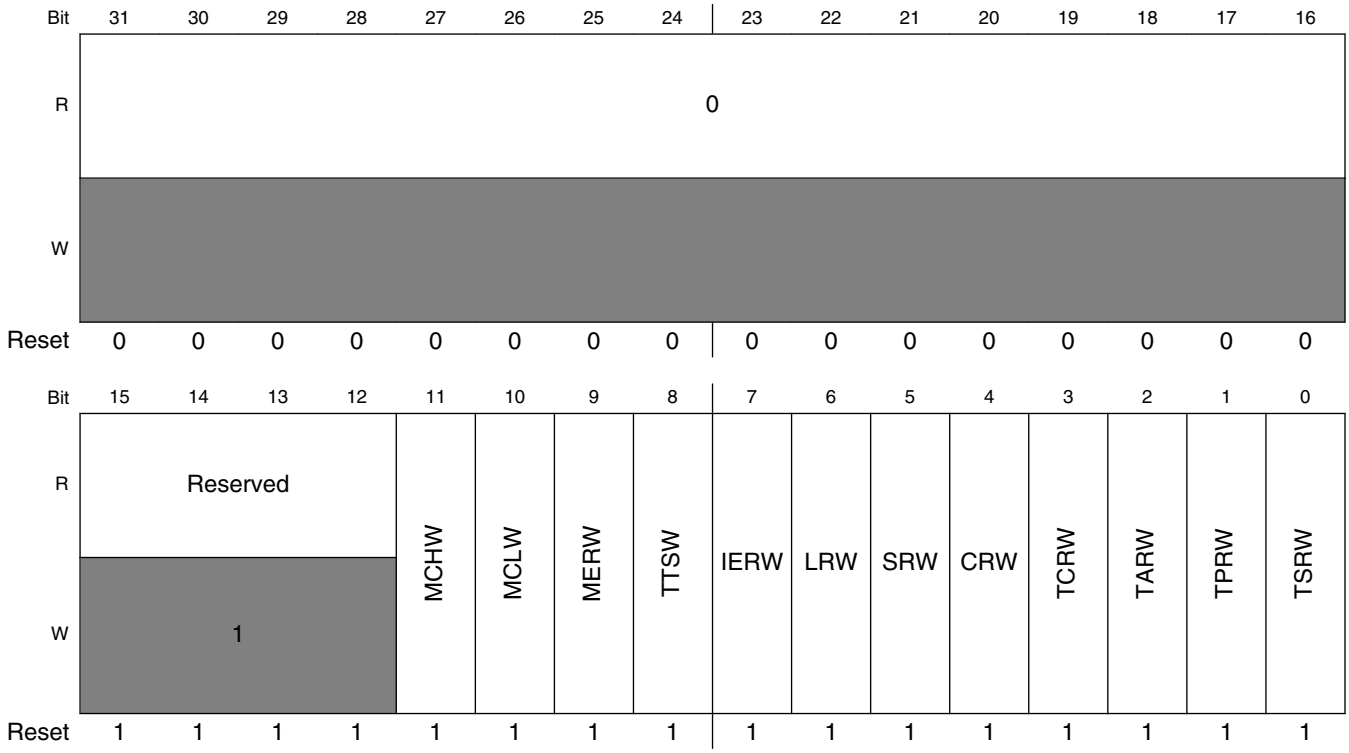
| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MCH | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### RTC_MCHR field descriptions

| Field | Description |
|---|---|
| 31–0<br>MCH | Monotonic Counter High<br><br>When the time invalid flag is set, the monotonic counter is held in reset. When the monotonic counter enable is clear, a write to this register will load the counter with the value written. When the monotonic counter enable is set, a write to this register will cause it to increment. A write to monotonic counter low that causes it to overflow will also increment monotonic counter high. |

## 44.2.13 RTC Write Access Register (RTC_WAR)

Address: 4003_D000h base + 800h offset = 4003_D800h




### RTC_WAR field descriptions

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–12<br>Reserved | This field is reserved. |
| 11<br>MCHW | Monotonic Counter High Write<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Writes to the Monotonic Counter High Register are ignored.<br>1 Writes to the Monotonic Counter High Register complete as normal. |
| 10<br>MCLW | Monotonic Counter Low Write<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Writes to the Monotonic Counter Low Register are ignored.<br>1 Writes to the Monotonic Counter Low Register complete as normal. |
| 9<br>MERW | Monotonic Enable Register Write<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset. |

*Table continues on the next page...*

## RTC_WAR field descriptions (continued)

| Field | Description |
|---|---|
| | 0 Writes to the Monotonic Enable Register are ignored.<br>1 Writes to the Monotonic Enable Register complete as normal. |
| 8<br>TTSW | Tamper Time Seconds Write<br><br>After being cleared, this bit is set onlyby system reset. It is not affected by VBAT POR or software reset.<br><br>0 Writes to the Tamper Time Seconds Register are ignored.<br>1 Writes to the Tamper Time Seconds Register complete as normal. |
| 7<br>IERW | Interrupt Enable Register Write<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Writes to the Interupt Enable Register are ignored.<br>1 Writes to the Interrupt Enable Register complete as normal. |
| 6<br>LRW | Lock Register Write<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Writes to the Lock Register are ignored.<br>1 Writes to the Lock Register complete as normal. |
| 5<br>SRW | Status Register Write<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Writes to the Status Register are ignored.<br>1 Writes to the Status Register complete as normal. |
| 4<br>CRW | Control Register Write<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Writes to the Control Register are ignored.<br>1 Writes to the Control Register complete as normal. |
| 3<br>TCRW | Time Compensation Register Write<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Writes to the Time Compensation Register are ignored.<br>1 Writes to the Time Compensation Register complete as normal. |
| 2<br>TARW | Time Alarm Register Write<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Writes to the Time Alarm Register are ignored.<br>1 Writes to the Time Alarm Register complete as normal. |
| 1<br>TPRW | Time Prescaler Register Write<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Writes to the Time Prescaler Register are ignored.<br>1 Writes to the Time Prescaler Register complete as normal. |

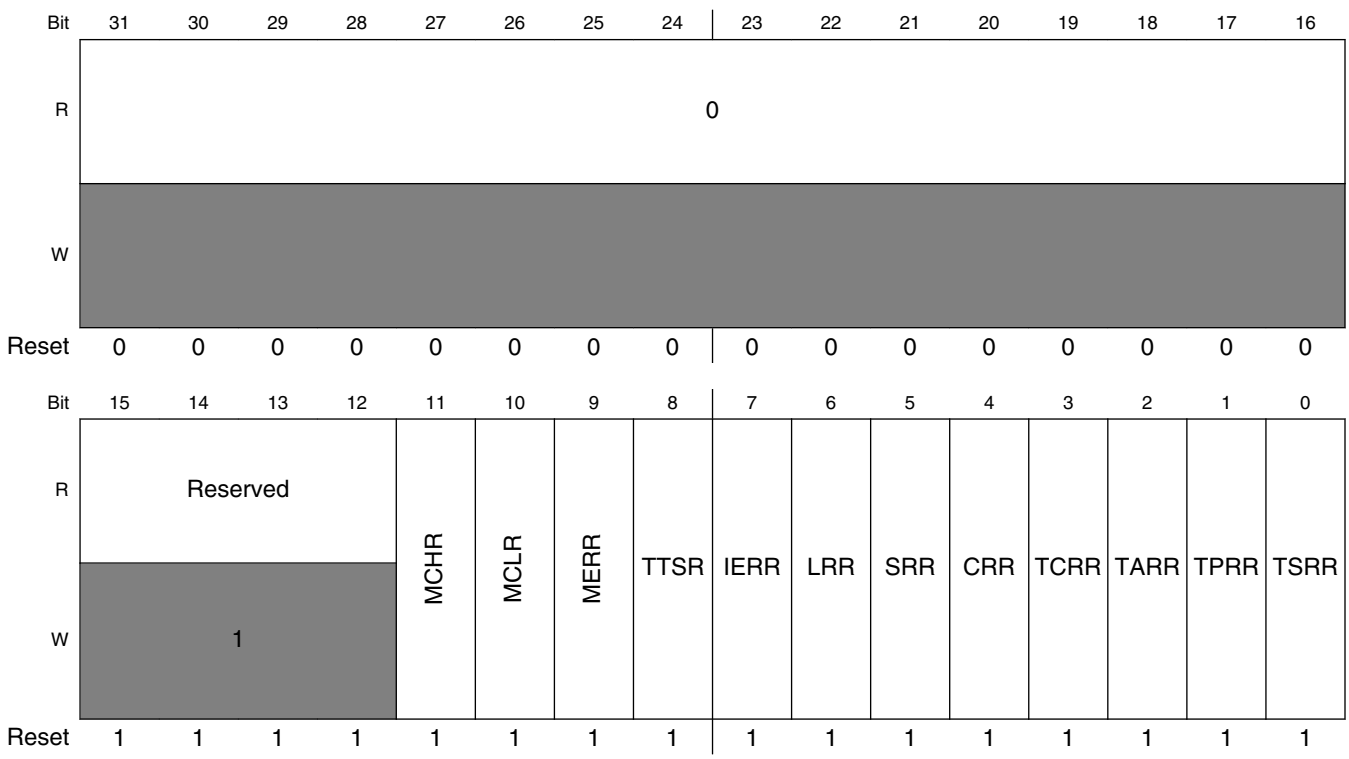
*Table continues on the next page...*

**RTC_WAR field descriptions (continued)**

| Field | Description |
|---|---|
| 0<br>TSRW | Time Seconds Register Write<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Writes to the Time Seconds Register are ignored.<br>1 Writes to the Time Seconds Register complete as normal. |

## 44.2.14 RTC Read Access Register (RTC_RAR)

Address: 4003_D000h base + 804h offset = 4003_D804h




**RTC_RAR field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–12<br>Reserved | This field is reserved. |
| 11<br>MCHR | Monotonic Counter High Read<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Reads to the Monotonic Counter High Register are ignored.<br>1 Reads to the Monotonic Counter High Register complete as normal. |

*Table continues on the next page...*

## RTC_RAR field descriptions (continued)

| Field | Description |
|---|---|
| 10<br>MCLR | Monotonic Counter Low Read<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Reads to the Monotonic Counter Low Register are ignored.<br>1 Reads to the Monotonic Counter Low Register complete as normal. |
| 9<br>MERR | Monotonic Enable Register Read<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Reads to the Monotonic Enable Register are ignored.<br>1 Reads to the Monotonic Enable Register complete as normal. |
| 8<br>TTSR | Tamper Time Seconds Read<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Reads to the Tamper Time Seconds Register are ignored.<br>1 Reads to the Tamper Time Seconds Register complete as normal. |
| 7<br>IERR | Interrupt Enable Register Read<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Reads to the Interrupt Enable Register are ignored.<br>1 Reads to the Interrupt Enable Register complete as normal. |
| 6<br>LRR | Lock Register Read<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Reads to the Lock Register are ignored.<br>1 Reads to the Lock Register complete as normal. |
| 5<br>SRR | Status Register Read<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Reads to the Status Register are ignored.<br>1 Reads to the Status Register complete as normal. |
| 4<br>CRR | Control Register Read<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Reads to the Control Register are ignored.<br>1 Reads to the Control Register complete as normal. |
| 3<br>TCRR | Time Compensation Register Read<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Reads to the Time Compensation Register are ignored.<br>1 Reads to the Time Compensation Register complete as normal. |
| 2<br>TARR | Time Alarm Register Read<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset. |

*Table continues on the next page...*

**RTC_RAR field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Reads to the Time Alarm Register are ignored.<br>1 Reads to the Time Alarm Register complete as normal. |
| 1<br>TPRR | Time Prescaler Register Read<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Reads to the Time Pprescaler Register are ignored.<br>1 Reads to the Time Prescaler Register complete as normal. |
| 0<br>TSRR | Time Seconds Register Read<br><br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br><br>0 Reads to the Time Seconds Register are ignored.<br>1 Reads to the Time Seconds Register complete as normal. |

## 44.3 Functional description

### 44.3.1 Power, clocking, and reset

The RTC is an always powered block that remains active in all low power modes and is powered by the battery power supply (VBAT). The battery power supply ensures that the RTC registers retain their state during chip power-down and that the RTC time counter remains operational.

The time counter within the RTC is clocked by a 32.768 kHz clock and can supply this clock to other peripherals. The 32.768 kHz clock can only be sourced from an external crystal using the oscillator that is part of the RTC module.

The RTC includes its own analog POR block, which generates a VBAT power-on-reset signal whenever the RTC module is powered up and initializes all RTC registers to their default state. A software reset bit can also initialize all RTC registers. The RTC also monitors the chip power supply and electrically isolates itself when the rest of the chip is powered down.

Any attempt to access an RTC register, except the access control registers, when VBAT is powered down, when the RTC is electrically isolated, or when VBAT POR is asserted, will result in a bus error.

#### 44.3.1.1 Oscillator control

The 32.768 kHz crystal oscillator is disabled at VBAT POR and must be enabled by software. After enabling the cystal oscillator, wait the oscillator startup time before setting SR[TCE] or using the oscillator clock external to the RTC.

The crystal oscillator includes tunable capacitors that can be configured by software. Do not change the capacitance unless the oscillator is disabled.

#### 44.3.1.2 Software reset

Writing one to the CR[SWR] forces the equivalent of a VBAT POR to the rest of the RTC module. The CR[SWR] is not affected by the software reset and must be cleared by software. The access control registers are not affected by either VBAT POR or the software reset; they are reset by the chip reset.

#### 44.3.1.3 Supervisor access

When the supervisor access control bit is clear, only supervisor mode software can write to the RTC registers or read the RTC tamper and monotonic registers, non-supervisor mode software will generate a bus error. Both supervisor and non-supervisor mode software can always read the other RTC registers.

### 44.3.2 Time counter

The time counter consists of a 32-bit seconds counter that increments once every second and a 16-bit prescaler register that increments once every 32.768 kHz clock cycle.

The time seconds register and time prescaler register can be written only when SR[TCE] is clear. Always write to the prescaler register before writing to the seconds register, because the seconds register increments on the falling edge of bit 14 of the prescaler register.

The time prescaler register increments provided SR[TCE] is set, SR[TIF] is clear, SR[TOF] is clear, and the 32.768 kHz clock source is present. After enabling the oscillator, wait the oscillator startup time before setting SR[TCE] to allow time for the oscillator clock output to stabilize.

If the time seconds register overflows then the SR[TOF] will set and the time prescaler register will stop incrementing. Clear SR[TOF] by initializing the time seconds register. The time seconds register and time prescaler register read as zero whenever SR[TOF] is set.

SR[TIF] is set on VBAT POR and software reset and is cleared by initializing the time seconds register. The time seconds register and time prescaler register read as zero whenever SR[TIF] is set.

### 44.3.3 Compensation

The compensation logic provides an accurate and wide compensation range and can correct errors as high as 3906 ppm and as low as 0.12 ppm. The compensation factor must be calculated externally to the RTC and supplied by software to the compensation register. The RTC itself does not calculate the amount of compensation that is required, although the 1 Hz clock is output to an external pin in support of external calibration logic.

Crystal compensation can be supported by using firmware and crystal characteristics to determine the compensation amount. Temperature compensation can be supported by firmware that periodically measures the external temperature via ADC and updates the compensation register based on a look-up table that specifies the change in crystal frequency over temperature.

The compensation logic alters the number of 32.768 kHz clock cycles it takes for the prescaler register to overflow and increment the time seconds counter. The time compensation value is used to adjust the number of clock cycles between -127 and +128. Cycles are added or subtracted from the prescaler register when the prescaler register equals 0x3FFF and then increments. The compensation interval is used to adjust the frequency at which the time compensation value is used, that is, from once a second to once every 256 seconds.

Updates to the time compensation register will not take effect until the next time the time seconds register increments and provided the previous compensation interval has expired. When the compensation interval is set to other than once a second then the compensation is applied in the first second interval and the remaining second intervals receive no compensation.

Compensation is disabled by configuring the time compensation register to zero.

### 44.3.4 Time alarm

The time alarm register, SR[TAF], and IER[TAIE] allow the RTC to generate an interrupt at a predefined time. The 32-bit time alarm register is compared with the 32-bit time seconds register each time it increments. The SR[TAF] will set when the time alarm register equals the time seconds register and the time seconds register increments.

The time alarm flag is cleared by writing the time alarm register. This will usually be the next alarm value, although writing a value that is less than the time seconds register, such as zero, will prevent the time alarm flag from setting again. The time alarm flag cannot otherwise be disabled, although the interrupt it generates is enabled or disabled by IER[TAIE].

### 44.3.5 Update mode

The Update Mode bit in the Control register (CR[UM]) configures software write access to the Time Counter Enable (SR[TCE]) bit. When CR[UM] is clear, SR[TCE] can be written only when the LR[SRL] bit is set. When CR[UM] is set, the SR[TCE] can also be written when SR[TCE] is clear or when SR[TIF] or SR[TOF] are set. This allows the time seconds and prescaler registers to be initialized whenever time is invalidated, while preventing the time seconds and prescaler registers from being changed on the fly. When LR[SRL] is set, CR[UM] has no effect on SR[TCE].

CR[UM] also configures software write access to the Monotonic Counter Enable (MER[MCE]) bit. When CR[UM] is clear, MER[MCE] can be written only when LR[MEL] is set. When CR[UM] is set, MER[MCE] can also be written when MER[MCE] is clear or when SR[TIF] or SR[MOF] are set. This allows the monotonic counter register to be initialized whenever the monotonic counter is invalid, while preventing the monotonic counter from being changed on the fly. When LR[MEL] is set, CR[UM] has no effect on MCR[MCE].

### 44.3.6 Monotonic counter

The 64-bit Monotonic Counter is a counter that cannot be exhausted or return to any previous value, once it has been initialized. If the monotonic overflow flag is set, the monotonic counter returns zero and does not increment.

Depending on the value of the monotonic counter enable bit, writing to the monotonic counter either initializes the register with the value written, or increments the register by one (and the value written is ignored).

When the monotonic counter is enabled, the monotonic counter high increments on either a write to the monotonic counter high register or if the monotonic counter low register overflows (due to a write to the monotonic counter low register). The monotonic overflow flag sets when the monotonic counter high register overflows and is cleared by writing the monotonic counter high register when the monotonic counter is disabled.

The monotonic counter is held in reset whenever the time invalid flag is set. Always clear the time invalid flag before initializing the monotonic counter.

### 44.3.7 Register lock

The lock register can be used to block write accesses to certain registers until the next VBAT POR or software reset. Locking the control register will disable the software reset. Locking the lock register will block future updates to the lock register.

Write accesses to a locked register are ignored and do not generate a bus error.

### 44.3.8 Access control

The read access and write access registers are implemented in the chip power domain and reset on the chip reset. They are not affected by the VBAT POR or the software reset. They are used to block read or write accesses to each register until the next chip system reset. When accesses are blocked, the bus access is not seen in the VBAT power supply and does not generate a bus error.

### 44.3.9 Interrupt

The RTC interrupt is asserted whenever a status flag and the corresponding interrupt enable bit are both set. It is always asserted on VBAT POR, and software reset, and when the VBAT power supply is powered down. The RTC interrupt is enabled at the chip level by enabling the chip-specific RTC clock gate control bit. The RTC interrupt can be used to wakeup the chip from any low-power mode. If the RTC wakeup pin is enabled and the chip is powered down, the RTC interrupt will cause the wakeup pin to assert.

The optional RTC seconds interrupt is an edge-sensitive interrupt with a dedicated interrupt vector that is generated once a second and requires no software overhead (there is no corresponding status flag to clear). It is enabled in the RTC by the time seconds interrupt enable bit and enabled at the chip level by setting the chip-specific RTC clock gate control bit. The RTC seconds interrupt does not cause the RTC wakeup pin to assert. This interrupt is optional and may not be implemented on all devices.

# Chapter 45
# Universal Serial Bus OTG Controller (USBOTG)

## 45.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

This section describes the USB. The OTG implementation in this module provides limited host functionality and device solutions for implementing a USB 2.0 full-speed/ low-speed compliant peripheral. The OTG implementation supports the On-The-Go (OTG) addendum to the USB 2.0 Specification. Only one protocol can be active at any time. A negotiation protocol must be used to switch to a USB host functionality from a USB device. This is known as the Master Negotiation Protocol (MNP).

### 45.1.1 USB

The USB is a cable bus that supports data exchange between a host computer and a wide range of simultaneously accessible peripherals. The attached peripherals share USB bandwidth through a host-scheduled, token-based protocol. The bus allows peripherals to be attached, configured, used, and detached while the host and other peripherals are in operation.

USB software provides a uniform view of the system for all application software, hiding implementation details making application software more portable. It manages the dynamic attach and detach of peripherals.

There is only one host in any USB system. The USB interface to the host computer system is referred to as the Host Controller.

There may be multiple USB devices in any system such as joysticks, speakers, printers, etc. USB devices present a standard USB interface in terms of comprehension, response, and standard capability.

The host initiates transactions to specific peripherals, whereas the device responds to control transactions. The device sends and receives data to and from the host using a standard USB data format. USB 2.0 full-speed /low-speed peripherals operate at 12Mbit/s or 1.5 Mbit/s.

For additional information, see the USB 2.0 specification.

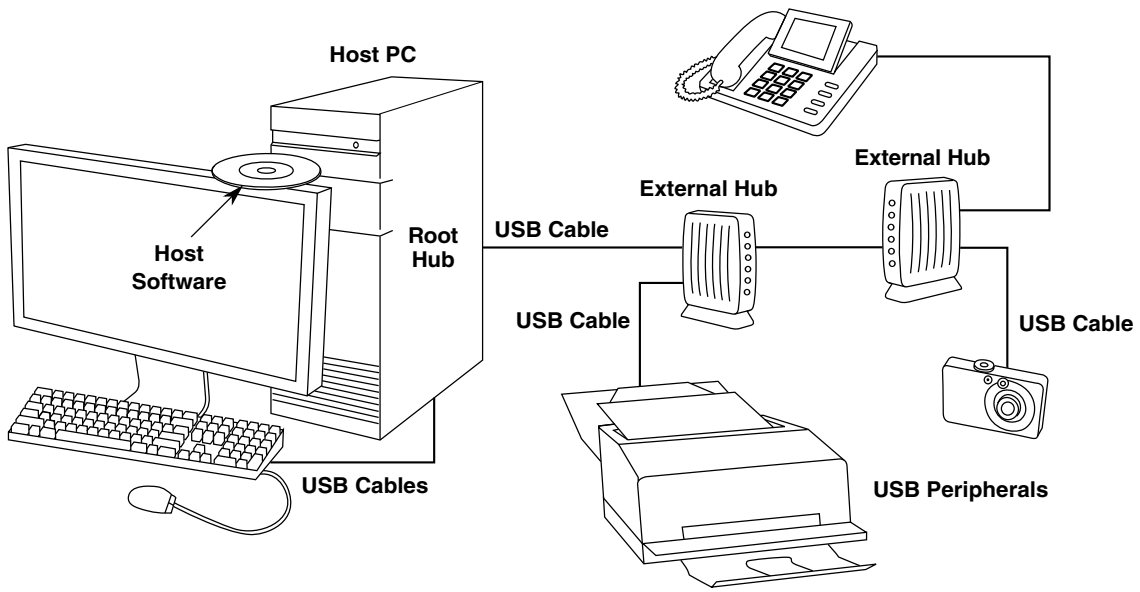



**Figure 45-1. Example USB 2.0 system configuration**

## 45.1.2 USB On-The-Go

USB is a popular standard for connecting peripherals and portable consumer electronic devices such as digital cameras and hand-held computers to host PCs. The On-The-Go (OTG) Supplement to the USB Specification extends USB to peer-to-peer application. Using USB OTG technology consumer electronics, peripherals, and portable devices can connect to each other to exchange data. For example, a digital camera can connect directly to a printer, or a keyboard can connect to a Personal Digital Assistant to exchange data.

With the USB On-The-Go product, you can develop a fully USB-compliant peripheral device that can also assume the role of a USB host. Software determines the role of the device based on hardware signals, and then initializes the device in the appropriate mode of operation (host or peripheral) based on how it is connected. After connecting the devices can negotiate using the OTG protocols to assume the role of host or peripheral based on the task to be accomplished.

For additional information, see the *On-The-Go Supplement to the USB 2.0 Specification*.

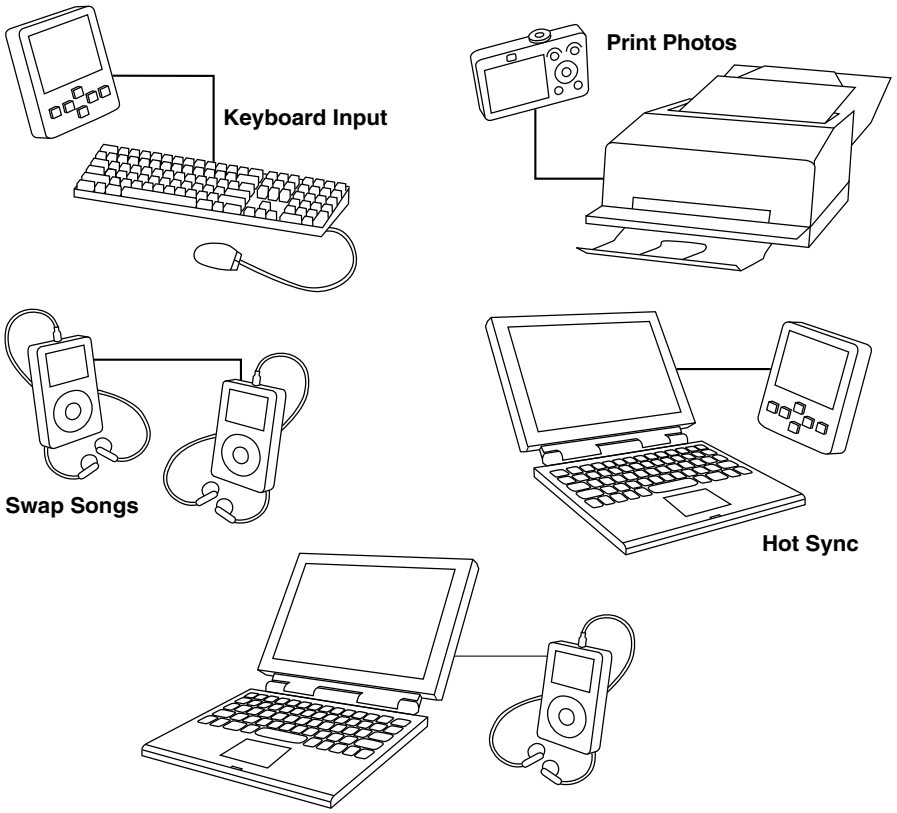



**Figure 45-2. Example USB 2.0 On-The-Go configurations**

### 45.1.3 USB-FS Features

- USB 1.1 and 2.0 compliant full-speed device controller
- 16 bidirectional end points
- DMA or FIFO data stream interfaces
- Low-power consumption
- On-The-Go protocol logic

## 45.2 Functional description

The USB-FS 2.0 full-speed/low-speed module communicates with the processor core through status registers, control registers, and data structures in memory.

### 45.2.1 Data Structures

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus. To efficiently manage USB endpoint communications the USB-FS implements a Buffer Descriptor Table (BDT) in system memory. See Figure 45-3.

## 45.3 Programmers interface

This section discusses the major components of the programming model for the USB module.

### 45.3.1 Buffer Descriptor Table

To efficiently manage USB endpoint communications the USB-FS implements a Buffer Descriptor Table (BDT) in system memory. The BDT resides on a 512-byte boundary in system memory and is pointed to by the BDT Page Registers. Every endpoint direction requires two 8-byte Buffer Descriptor (BD) entries. Therefore, a system with 16 fully bidirectional endpoints would require 512 bytes of system memory to implement the BDT. The two BD entries allows for an EVEN BD and ODD BD entry for each endpoint direction. This allows the microprocessor to process one BD while the USB-FS is processing the other BD. Double buffering BDs in this way allows the USB-FS to transfer data easily at the maximum throughput provided by USB.

The software API intelligently manages buffers for the USB-FS by updating the BDT when needed. This allows the USB-FS to efficiently manage data transmission and reception, while the microprocessor performs communication overhead processing and other function dependent applications. Because the buffers are shared between the microprocessor and the USB-FS, a simple semaphore mechanism is used to distinguish who is allowed to update the BDT and buffers in system memory. A semaphore, the OWN bit, is cleared to 0 when the BD entry is owned by the microprocessor. The microprocessor is allowed read and write access to the BD entry and the buffer in system memory when the OWN bit is 0. When the OWN bit is set to 1, the BD entry and the buffer in system memory are owned by the USB-FS. The USB-FS now has full read and write access and the microprocessor must not modify the BD or its corresponding data buffer. The BD also contains indirect address pointers to where the actual buffer resides in system memory. This indirect address mechanism is shown in the following diagram.

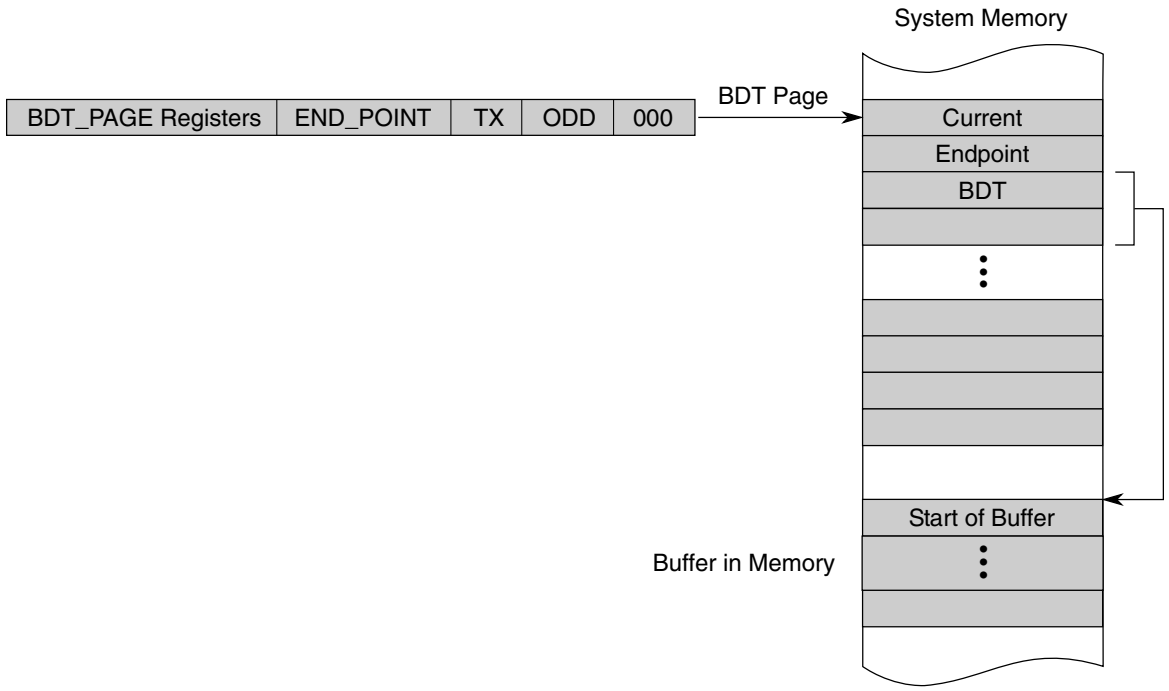



**Figure 45-3. Buffer descriptor table**

## 45.3.2 RX vs. TX as a USB target device or USB host

The USB-FS core uses software control to switch between two modes of operation:

- USB target device
- USB hosts

In either mode, USB host or USB target device, the same data paths and buffer descriptors are used for the transmission and reception of data. For this reason, a USB-FS core centric nomenclature is used to describe the direction of the data transfer between the USB-FS core and the USB:

- RX (or receive) describes transfers that move data from the USB to memory.
- TX (or transmit) describes transfers that move data from memory to the USB.

The following table shows how the data direction corresponds to the USB token type in host and target device applications.

**Table 45-1. Data direction for USB host or USB target**

| | RX | TX |
|---|---|---|
| **Device** | OUT or SETUP | IN |
| **Host** | IN | OUT or SETUP |

### 45.3.3 Addressing BDT entries

An understanding of the addressing mechanism of the Buffer Descriptor Table is useful when accessing endpoint data via the USB-FS or microprocessor. Some points of interest are:

- The BDT occupies up to 512 bytes of system memory.
- 16 bidirectional endpoints can be supported with a full BDT of 512 bytes.
- 16 bytes are needed for each USB endpoint direction.
- Applications with less than 16 endpoints require less RAM to implement the BDT.
- The BDT Page Registers (BDT_PAGE) point to the starting location of the BDT.
- The BDT must be located on a 512-byte boundary in system memory.
- All enabled TX and RX endpoint BD entries are indexed into the BDT to allow easy access via the USB-FS or MCU core.

When a USB token on an enabled endpoint is received, the USB-FS uses its integrated DMA controller to interrogate the BDT. The USB-FS reads the corresponding endpoint BD entry to determine whether it owns the BD and corresponding buffer in system memory.

To compute the entry point in to the BDT, the BDT_PAGE registers is concatenated with the current endpoint and the TX and ODD fields to form a 32-bit address. This address mechanism is shown below:

**Table 45-2. BDT address calculation fields**

| Field | Description |
|---|---|
| BDT_PAGE | BDT_PAGE registers in the Control Register Block |
| END_POINT | END POINT field from the USB TOKEN |
| TX | 1 for transmit transfers and 0 for receive transfers |
| ODD | Maintained within the USB-FS SIE. It corresponds to the buffer currently in use. The buffers are used in a ping-pong fashion. |

### 45.3.4 Buffer Descriptors (BDs)

A buffer descriptor provides endpoint buffer control information for the USB-FS and processor. The Buffer Descriptors have different meaning based on whether it is the USB-FS or processor reading the BD in memory.

The USB-FS Controller uses the data stored in the BDs to determine:

- Who owns the buffer in system memory
- Data0 or Data1 PID
- Whether to release ownership upon packet completion
- No address increment (FIFO mode)
- Whether data toggle synchronization is enabled
- How much data is to be transmitted or received
- Where the buffer resides in system memory

While the processor uses the data stored in the BDs to determine:

- Who owns the buffer in system memory
- Data0 or Data1 PID
- The received TOKEN PID
- How much data was transmitted or received
- Where the buffer resides in system memory

The format for the BD is shown in the following figure.

**Table 45-3. Buffer descriptor format**

| 31:26 | 25:16 | 15:8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| RSVD | BC (10 bits) | RSVD | OWN | DATA0/1 | KEEP/ TOK_PID[3] | NINC/ TOK_PID[2] | DTS/ TOK_PID[1] | BDT_STALL/ TOK_PID[0] | 0 | 0 |
| Buffer Address (32-Bits) | | | | | | | | | | |

**Table 45-4. Buffer descriptor fields**

| Field | Description |
|---|---|
| 31–26<br>RSVD | Reserved |
| 25–16<br>BC | Byte Count<br>Represents the 10-bit byte count. The USB-FS SIE changes this field upon the completion of a RX transfer with the byte count of the data received. |
| 15–8<br>RSVD | Reserved |

*Table continues on the next page...*

## Table 45-4. Buffer descriptor fields (continued)

| Field | Description |
|---|---|
| 7<br>OWN | Determines whether the processor or the USB-FS currently owns the buffer. Except when KEEP=1, the SIE hands ownership back to the processor after completing the token by clearing this bit.<br>This must always be the last byte of the BD that the processor updates when it initializes a BD.<br>0 The processor has exclusive access to the BD. The USB-FS ignores all other fields in the BD.<br>1 USB-FS has exclusive access to the BD. After the BD has been assigned to the USB-FS, the processor should not change it in any way. |
| 6<br>DATA0/1 | Defines whether a DATA0 field (DATA0/1=0) or a DATA1 (DATA0/1=1) field was transmitted or received. It is unchanged by the USB-FS. |
| 5<br>KEEP/<br>TOK_PID[3] | Typically, this bit is 1 with ISO endpoints feeding a FIFO. The microprocessor is not informed that a token has been processed, the data is simply transferred to or from the FIFO. When KEEP is set, normally the NINC bit is also set to prevent address increment.<br>0 Bit 3 of the current token PID is written back to the BD by the USB-FS. Allows the USB-FS to release the BD when a token has been processed.<br>1 This bit is unchanged by the USB-FS. If the OWN bit also is set, the BD remains owned by the USB-FS forever. |
| 4<br>NINC/<br>TOK_PID[2] | No Increment (NINC)<br>Disables the DMA engine address increment. This forces the DMA engine to read or write from the same address. This is useful for endpoints when data needs to be read from or written to a single location such as a FIFO. Typically this bit is set with the KEEP bit for ISO endpoints that are interfacing to a FIFO.<br>0 The USB-FS writes bit 2 of the current token PID to the BD.<br>1 This bit is unchanged by the USB-FS. |
| 3<br>DTS/<br>TOK_PID[1] | Setting this bit enables the USB-FS to perform Data Toggle Synchronization.<br>• If KEEP=0, bit 1 of the current token PID is written back to the BD.<br>• If KEEP=1, this bit is unchanged by the USB-FS.<br>0 Data Toggle Synchronization is disabled.<br>1 Enables the USB-FS to perform Data Toggle Synchronization. |
| 2<br>BDT_STALL<br>TOK_PID[0] | Setting this bit causes the USB-FS to issue a STALL handshake if a token is received by the SIE that would use the BDT in this location. The BDT is not consumed by the SIE (the owns bit remains set and the rest of the BDT is unchanged) when a BDT-STALL bit is set.<br>• If KEEP=0, bit 0 of the current token PID is written back to the BD.<br>• If KEEP=1, this bit is unchanged by the USB-FS.<br>0 No stall issued.<br>1 The BDT is not consumed by the SIE (the OWN bit remains set and the rest of the BDT is unchanged). |

*Table continues on the next page...*

**Table 45-4. Buffer descriptor fields (continued)**

| Field | Description |
|---|---|
| TOK_PID[n] | Bits [5:2] can also represent the current token PID. The current token PID is written back in to the BD by the USB-FS when a transfer completes. The values written back are the token PID values from the USB specification:<br>• 0x1h for an OUT token.<br>• 0x9h for an IN token.<br>• 0xDh for a SETUP token.<br><br>In host mode, this field is used to report the last returned PID or a transfer status indication. The possible values returned are:<br>• 0x3h DATA0<br>• 0xBh DATA1<br>• 0x2h ACK<br>• 0xEh STALL<br>• 0xAh NAK<br>• 0x0h Bus Timeout<br>• 0xFh Data Error |
| 1–0<br><br>Reserved | Reserved, should read as zeroes. |
| ADDR[31:0] | Address<br><br>Represents the 32-bit buffer address in system memory. These bits are unchanged by the USB-FS. |

## 45.3.5 USB transaction

When the USB-FS transmits or receives data, it computes the BDT address using the address generation shown in "Addressing Buffer Descriptor Entries" table.

If OWN =1, the following process occurs:

1. The USB-FS reads the BDT.
2. The SIE transfers the data via the DMA to or from the buffer pointed to by the ADDR field of the BD.
3. When the TOKEN is complete, the USB-FS updates the BDT and, if KEEP=0, changes the OWN bit to 0.
4. The STAT register is updated and the TOK_DNE interrupt is set.
5. When the processor processes the TOK_DNE interrupt, it reads from the status register all the information needed to process the endpoint.
6. At this point, the processor allocates a new BD so that additional USB data can be transmitted or received for that endpoint, and then processes the last BD.

The following figure shows a timeline of how a typical USB token is processed after the BDT is read and OWN=1.

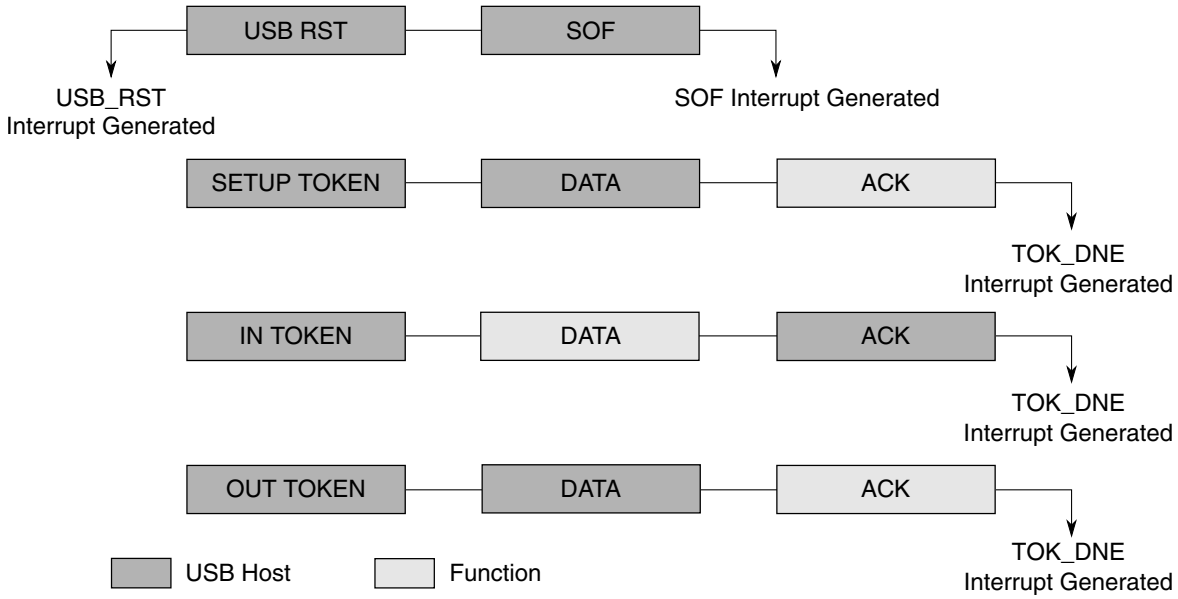



**Figure 45-4. USB token transaction**

The USB has two sources for the DMA overrun error:

Memory Latency

The memory latency may be too high and cause the receive FIFO to overflow. This is predominantly a hardware performance issue, usually caused by transient memory access issues.

Oversized Packets

The packet received may be larger than the negotiated *MaxPacket* size. Typically, this is caused by a software bug. For DMA overrun errors due to oversized data packets, the USB specification is ambiguous. It assumes correct software drivers on both sides. NAKing the packet can result in retransmission of the already oversized packet data. Therefore, in response to oversized packets, the USB core continues ACKing the packet for non-isochronous transfers.

**Table 45-5. USB responses to DMA overrun errors**

| Errors due to Memory Latency | Errors due to Oversized Packets |
|---|---|
| Non-Acknowledgment (NAK) or Bus Timeout (BTO) — See bit 4 in "Error Interrupt Status Register (ERRSTAT)" as appropriate for the class of transaction. | Continues acknowledging (ACKing) the packet for non-isochronous transfers. |
| — | The data written to memory is clipped to the MaxPacket size so as not to corrupt system memory. |
| The DMAERR bit is set in the ERRSTAT register for host and device modes of operation. Depending on the values of the INTENB and ERRENB register, the core may assert an interrupt to notify the processor of the DMA error. | Asserts ERRSTAT[DMAERR] ,which can trigger an interrupt and TOKDNE interrupt fires. Note: The TOK_PID field of the BDT is not 1111 because the DMAERR is not due to latency. |

*Table continues on the next page...*

**Table 45-5. USB responses to DMA overrun errors (continued)**

| Errors due to Memory Latency | Errors due to Oversized Packets |
|---|---|
| • For host mode, the TOKDNE interrupt is generated and the TOK_PID field of the BDT is 1111 to indicate the DMA latency error. Host mode software can decide to retry or move to next scheduled item.<br>• In device mode, the BDT is not written back nor is the TOKDNE interrupt triggered because it is assumed that a second attempt is queued and will succeed in the future. | The packet length field written back to the BDT is the MaxPacket value that represents the length of the clipped data actually written to memory. |
| From here, the software can decide an appropriate course of action for future transactions such as stalling the endpoint, canceling the transfer, disabling the endpoint, etc. | |

## 45.4 Memory map/Register definitions

This section provides the memory map and detailed descriptions of all USB interface registers.

**USB memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4007_2000 | Peripheral ID register (USB0_PERID) | 8 | R | 04h | 45.4.1/1133 |
| 4007_2004 | Peripheral ID Complement register (USB0_IDCOMP) | 8 | R | FBh | 45.4.2/1134 |
| 4007_2008 | Peripheral Revision register (USB0_REV) | 8 | R | 33h | 45.4.3/1134 |
| 4007_200C | Peripheral Additional Info register (USB0_ADDINFO) | 8 | R | 01h | 45.4.4/1135 |
| 4007_2010 | OTG Interrupt Status register (USB0_OTGISTAT) | 8 | R/W | 00h | 45.4.5/1135 |
| 4007_2014 | OTG Interrupt Control Register (USB0_OTGICR) | 8 | R/W | 00h | 45.4.6/1136 |
| 4007_2018 | OTG Status register (USB0_OTGSTAT) | 8 | R/W | 00h | 45.4.7/1137 |
| 4007_201C | OTG Control register (USB0_OTGCTL) | 8 | R/W | 00h | 45.4.8/1138 |
| 4007_2080 | Interrupt Status register (USB0_ISTAT) | 8 | R/W | 00h | 45.4.9/1139 |
| 4007_2084 | Interrupt Enable register (USB0_INTEN) | 8 | R/W | 00h | 45.4.10/ 1140 |
| 4007_2088 | Error Interrupt Status register (USB0_ERRSTAT) | 8 | R/W | 00h | 45.4.11/ 1141 |
| 4007_208C | Error Interrupt Enable register (USB0_ERREN) | 8 | R/W | 00h | 45.4.12/ 1142 |
| 4007_2090 | Status register (USB0_STAT) | 8 | R | 00h | 45.4.13/ 1143 |
| 4007_2094 | Control register (USB0_CTL) | 8 | R/W | 00h | 45.4.14/ 1144 |
| 4007_2098 | Address register (USB0_ADDR) | 8 | R/W | 00h | 45.4.15/ 1145 |

*Table continues on the next page...*

## USB memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4007_209C | BDT Page Register 1 (USB0_BDTPAGE1) | 8 | R/W | 00h | 45.4.16/ 1146 |
| 4007_20A0 | Frame Number Register Low (USB0_FRMNUML) | 8 | R/W | 00h | 45.4.17/ 1146 |
| 4007_20A4 | Frame Number Register High (USB0_FRMNUMH) | 8 | R/W | 00h | 45.4.18/ 1147 |
| 4007_20A8 | Token register (USB0_TOKEN) | 8 | R/W | 00h | 45.4.19/ 1147 |
| 4007_20AC | SOF Threshold Register (USB0_SOFTHLD) | 8 | R/W | 00h | 45.4.20/ 1148 |
| 4007_20B0 | BDT Page Register 2 (USB0_BDTPAGE2) | 8 | R/W | 00h | 45.4.21/ 1149 |
| 4007_20B4 | BDT Page Register 3 (USB0_BDTPAGE3) | 8 | R/W | 00h | 45.4.22/ 1149 |
| 4007_20C0 | Endpoint Control register (USB0_ENDPT0) | 8 | R/W | 00h | 45.4.23/ 1149 |
| 4007_20C4 | Endpoint Control register (USB0_ENDPT1) | 8 | R/W | 00h | 45.4.23/ 1149 |
| 4007_20C8 | Endpoint Control register (USB0_ENDPT2) | 8 | R/W | 00h | 45.4.23/ 1149 |
| 4007_20CC | Endpoint Control register (USB0_ENDPT3) | 8 | R/W | 00h | 45.4.23/ 1149 |
| 4007_20D0 | Endpoint Control register (USB0_ENDPT4) | 8 | R/W | 00h | 45.4.23/ 1149 |
| 4007_20D4 | Endpoint Control register (USB0_ENDPT5) | 8 | R/W | 00h | 45.4.23/ 1149 |
| 4007_20D8 | Endpoint Control register (USB0_ENDPT6) | 8 | R/W | 00h | 45.4.23/ 1149 |
| 4007_20DC | Endpoint Control register (USB0_ENDPT7) | 8 | R/W | 00h | 45.4.23/ 1149 |
| 4007_20E0 | Endpoint Control register (USB0_ENDPT8) | 8 | R/W | 00h | 45.4.23/ 1149 |
| 4007_20E4 | Endpoint Control register (USB0_ENDPT9) | 8 | R/W | 00h | 45.4.23/ 1149 |
| 4007_20E8 | Endpoint Control register (USB0_ENDPT10) | 8 | R/W | 00h | 45.4.23/ 1149 |
| 4007_20EC | Endpoint Control register (USB0_ENDPT11) | 8 | R/W | 00h | 45.4.23/ 1149 |
| 4007_20F0 | Endpoint Control register (USB0_ENDPT12) | 8 | R/W | 00h | 45.4.23/ 1149 |
| 4007_20F4 | Endpoint Control register (USB0_ENDPT13) | 8 | R/W | 00h | 45.4.23/ 1149 |
| 4007_20F8 | Endpoint Control register (USB0_ENDPT14) | 8 | R/W | 00h | 45.4.23/ 1149 |

*Table continues on the next page...*

**USB memory map (continued)**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4007_20FC | Endpoint Control register (USB0_ENDPT15) | 8 | R/W | 00h | 45.4.23/ 1149 |
| 4007_2100 | USB Control register (USB0_USBCTRL) | 8 | R/W | C0h | 45.4.24/ 1150 |
| 4007_2104 | USB OTG Observe register (USB0_OBSERVE) | 8 | R | 50h | 45.4.25/ 1151 |
| 4007_2108 | USB OTG Control register (USB0_CONTROL) | 8 | R/W | 00h | 45.4.26/ 1152 |
| 4007_210C | USB Transceiver Control Register 0 (USB0_USBTRC0) | 8 | R/W | 00h | 45.4.27/ 1152 |
| 4007_2114 | Frame Adjust Register (USB0_USBFRMADJUST) | 8 | R/W | 00h | 45.4.28/ 1153 |

## 45.4.1 Peripheral ID register (USBx_PERID)

Reads back the value of 0x04. This value is defined for the USB peripheral.

Address: 4007_2000h base + 0h offset = 4007_2000h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | | ID | | | | | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

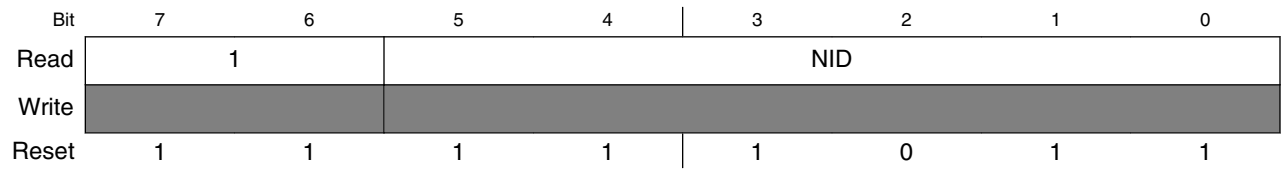
**USBx_PERID field descriptions**

| Field | Description |
|---|---|
| 7–6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5–0<br>ID | Peripheral Identification<br><br>This field always reads 0x4h. |

## 45.4.2 Peripheral ID Complement register (USBx_IDCOMP)

Reads back the complement of the Peripheral ID register. For the USB peripheral, the value is 0xFB.

Address: 4007_2000h base + 4h offset = 4007_2004h



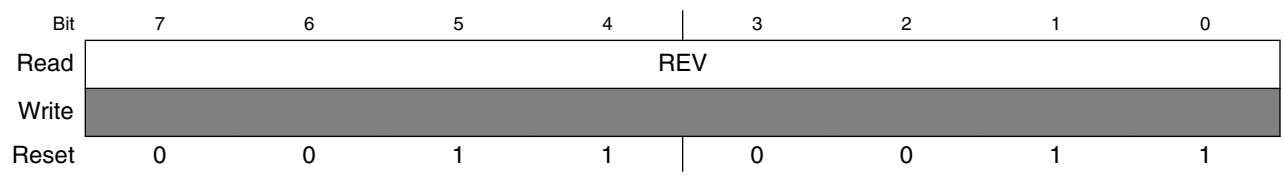

**USBx_IDCOMP field descriptions**

| Field | Description |
|---|---|
| 7–6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 1. |
| 5–0<br>NID | Ones complement of peripheral identification bits. |

## 45.4.3 Peripheral Revision register (USBx_REV)

Contains the revision number of the USB module.

Address: 4007_2000h base + 8h offset = 4007_2008h




**USBx_REV field descriptions**

| Field | Description |
|---|---|
| 7–0<br>REV | Revision<br><br>Indicate the revision number of the USB Core. |

## 45.4.4 Peripheral Additional Info register (USBx_ADDINFO)

Reads back the value of the fixed Interrupt Request Level (IRQNUM) along with the Host Enable bit.

Address: 4007_2000h base + Ch offset = 4007_200Ch

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | IRQNUM | | | | | 0 | | IEHOST |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**USBx_ADDINFO field descriptions**

| Field | Description |
|---|---|
| 7–3<br>IRQNUM | Assigned Interrupt Request Number |
| 2–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>IEHOST | When this bit is set, the USB peripheral is operating in host mode. |

## 45.4.5 OTG Interrupt Status register (USBx_OTGISTAT)

Records changes of the ID sense and VBUS signals. Software can read this register to determine the event that triggers interrupt. Only bits that have changed since the last software read are set. Writing a one to a bit clears the associated interrupt.

Address: 4007_2000h base + 10h offset = 4007_2010h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | IDCHG | ONEMSEC | LINE_<br>STATE_<br>CHG | 0 | SESSVLDC<br>HG | B_SESS_<br>CHG | 0 | AVBUSCHG |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**USBx_OTGISTAT field descriptions**

| Field | Description |
|---|---|
| 7<br>IDCHG | This bit is set when a change in the ID Signal from the USB connector is sensed. |
| 6<br>ONEMSEC | This bit is set when the 1 millisecond timer expires. This bit stays asserted until cleared by software. The interrupt must be serviced every millisecond to avoid losing 1msec counts. |
| 5<br>LINE_STATE_<br>CHG | This bit is set when the USB line state changes. The interrupt associated with this bit can be used to detect Reset, Resume, Connect, and Data Line Pulse signaling |

*Table continues on the next page...*

**USBx_OTGISTAT field descriptions (continued)**

| Field | Description |
|---|---|
| 4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3<br>SESSVLDCHG | This bit is set when a change in VBUS is detected indicating a session valid or a session no longer valid. |
| 2<br>B_SESS_CHG | This bit is set when a change in VBUS is detected on a B device. |
| 1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>AVBUSCHG | This bit is set when a change in VBUS is detected on an A device. |

## 45.4.6 OTG Interrupt Control Register (USBx_OTGICR)

Enables the corresponding interrupt status bits defined in the OTG Interrupt Status Register.

Address: 4007_2000h base + 14h offset = 4007_2014h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | IDEN | ONEMSECEN | LINESTATEEN | 0 | SESSVLDEN | BSESSEN | 0 | AVBUSEN |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**USBx_OTGICR field descriptions**

| Field | Description |
|---|---|
| 7<br>IDEN | ID Interrupt Enable<br><br>0 The ID interrupt is disabled<br>1 The ID interrupt is enabled |
| 6<br>ONEMSECEN | One Millisecond Interrupt Enable<br><br>0 Diables the 1ms timer interrupt.<br>1 Enables the 1ms timer interrupt. |
| 5<br>LINESTATEEN | Line State Change Interrupt Enable<br><br>0 Disables the LINE_STAT_CHG interrupt.<br>1 Enables the LINE_STAT_CHG interrupt. |
| 4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3<br>SESSVLDEN | Session Valid Interrupt Enable<br><br>0 Disables the SESSVLDCHG interrupt.<br>1 Enables the SESSVLDCHG interrupt. |
| 2<br>BSESSEN | B Session END Interrupt Enable |

*Table continues on the next page...*

**USBx_OTGICR field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Disables the B_SESS_CHG interrupt.<br>1 Enables the B_SESS_CHG interrupt. |
| 1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>AVBUSEN | A VBUS Valid Interrupt Enable<br><br>0 Disables the AVBUSCHG interrupt.<br>1 Enables the AVBUSCHG interrupt. |

## 45.4.7 OTG Status register (USBx_OTGSTAT)

Displays the actual value from the external comparator outputs of the ID pin and VBUS.

Address: 4007_2000h base + 18h offset = 4007_2018h

| Bit | 7 | 6 | 5 | 4 |
|---|---|---|---|---|
| Read | ID | ONEMSECEN | LINESTATESTABLE | 0 |
| Write | | | | |
| Reset | 0 | 0 | 0 | 0 |

| Bit | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| Read | SESS_VLD | BSESSEND | 0 | AVBUSVLD |
| Write | | | | |
| Reset | 0 | 0 | 0 | 0 |

**USBx_OTGSTAT field descriptions**

| Field | Description |
|---|---|
| 7<br>ID | Indicates the current state of the ID pin on the USB connector<br><br>0 Indicates a Type A cable is plugged into the USB connector.<br>1 Indicates no cable is attached or a Type B cable is plugged into the USB connector. |
| 6<br>ONEMSECEN | This bit is reserved for the 1ms count, but it is not useful to software. |
| 5<br>LINESTATESTABLE | Indicates that the internal signals that control the LINE_STATE_CHG field of OTGISTAT are stable for at least 1 millisecond. First read LINE_STATE_CHG field and then read this field. If this field reads as 1, then the value of LINE_STATE_CHG can be considered stable.<br><br>0 The LINE_STAT_CHG bit is not yet stable.<br>1 The LINE_STAT_CHG bit has been debounced and is stable. |
| 4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3<br>SESS_VLD | Session Valid<br><br>0 The VBUS voltage is below the B session valid threshold<br>1 The VBUS voltage is above the B session valid threshold. |

*Table continues on the next page...*

**USBx_OTGSTAT field descriptions (continued)**

| Field | Description |
|---|---|
| 2<br>BSESSEND | B Session End<br><br>0 The VBUS voltage is above the B session end threshold.<br>1 The VBUS voltage is below the B session end threshold. |
| 1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>AVBUSVLD | A VBUS Valid<br><br>0 The VBUS voltage is below the A VBUS Valid threshold.<br>1 The VBUS voltage is above the A VBUS Valid threshold. |

## 45.4.8 OTG Control register (USBx_OTGCTL)

Controls the operation of VBUS and Data Line termination resistors.

Address: 4007_2000h base + 1Ch offset = 4007_201Ch

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | DPHIGH | 0 | DPLOW | DMLOW | 0 | OTGEN | 0 | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**USBx_OTGCTL field descriptions**

| Field | Description |
|---|---|
| 7<br>DPHIGH | D+ Data Line pullup resistor enable<br><br>0 D+ pullup resistor is not enabled<br>1 D+ pullup resistor is enabled |
| 6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5<br>DPLOW | D+ Data Line pull-down resistor enable<br><br>This bit should always be enabled together with bit 4 (DMLOW)<br><br>0 D+ pulldown resistor is not enabled.<br>1 D+ pulldown resistor is enabled. |
| 4<br>DMLOW | D– Data Line pull-down resistor enable<br><br>0 D- pulldown resistor is not enabled.<br>1 D- pulldown resistor is enabled. |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>OTGEN | On-The-Go pullup/pulldown resistor enable |

*Table continues on the next page...*

**USBx_OTGCTL field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 If USB_EN is 1 and HOST_MODE is 0 in the Control Register (CTL), then the D+ Data Line pull-up resistors are enabled. If HOST_MODE is 1 the D+ and D– Data Line pull-down resistors are engaged.<br>1 The pull-up and pull-down controls in this register are used. |
| 1–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 45.4.9 Interrupt Status register (USBx_ISTAT)

Contains fields for each of the interrupt sources within the USB Module. Each of these fields are qualified with their respective interrupt enable bits. All fields of this register are logically OR'd together along with the OTG Interrupt Status Register (OTGSTAT) to form a single interrupt source for the processor's interrupt controller. After an interrupt bit has been set it may only be cleared by writing a one to the respective interrupt bit. This register contains the value of 0x00 after a reset.

Address: 4007_2000h base + 80h offset = 4007_2080h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | STALL | ATTACH | RESUME | SLEEP | TOKDNE | SOFTOK | ERROR | USBRST |
| Write | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**USBx_ISTAT field descriptions**

| Field | Description |
|---|---|
| 7<br>STALL | Stall Interrupt<br>In Target mode this bit is asserted when a STALL handshake is sent by the SIE.<br>In Host mode this bit is set when the USB Module detects a STALL acknowledge during the handshake phase of a USB transaction.This interrupt can be used to determine whether the last USB transaction was completed successfully or stalled. |
| 6<br>ATTACH | Attach Interrupt<br>This bit is set when the USB Module detects an attach of a USB device. This signal is only valid if HOSTMODEEN is true. This interrupt signifies that a peripheral is now present and must be configured. |
| 5<br>RESUME | This bit is set depending upon the DP/DM signals, and can be used to signal remote wake-up signaling on the USB bus. When not in suspend mode this interrupt must be disabled. |
| 4<br>SLEEP | This bit is set when the USB Module detects a constant idle on the USB bus for 3 ms. The sleep timer is reset by activity on the USB bus. |
| 3<br>TOKDNE | This bit is set when the current token being processed has completed. The processor must immediately read the STATUS (STAT) register to determine the EndPoint and BD used for this token. Clearing this bit (by writing a one) causes STAT to be cleared or the STAT holding register to be loaded into the STAT register. |
| 2<br>SOFTOK | This bit is set when the USB Module receives a Start Of Frame (SOF) token. |

*Table continues on the next page...*

**USBx_ISTAT field descriptions (continued)**

| Field | Description |
|---|---|
| | In Host mode this field is set when the SOF threshold is reached, so that software can prepare for the next SOF. |
| 1<br>ERROR | This bit is set when any of the error conditions within Error Interrupt Status (ERRSTAT) register occur. The processor must then read the ERRSTAT register to determine the source of the error. |
| 0<br>USBRST | This bit is set when the USB Module has decoded a valid USB reset. This informs the processor that it should write 0x00 into the address register and enable endpoint 0. USBRST is set after a USB reset has been detected for 2.5 microseconds. It is not asserted again until the USB reset condition has been removed and then reasserted. |

## 45.4.10 Interrupt Enable register (USBx_INTEN)

Contains enable fields for each of the interrupt sources within the USB Module. Setting any of these bits enables the respective interrupt source in the ISTAT register. This register contains the value of 0x00 after a reset.

Address: 4007_2000h base + 84h offset = 4007_2084h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | STALLEN | ATTACHEN | RESUMEEN | SLEEPEN | TOKDNEEN | SOFTOKEN | ERROREN | USBRSTEN |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**USBx_INTEN field descriptions**

| Field | Description |
|---|---|
| 7<br>STALLEN | STALL Interrupt Enable<br><br>0 Diasbles the STALL interrupt.<br>1 Enables the STALL interrupt. |
| 6<br>ATTACHEN | ATTACH Interrupt Enable<br><br>0 Disables the ATTACH interrupt.<br>1 Enables the ATTACH interrupt. |
| 5<br>RESUMEEN | RESUME Interrupt Enable<br><br>0 Disables the RESUME interrupt.<br>1 Enables the RESUME interrupt. |
| 4<br>SLEEPEN | SLEEP Interrupt Enable<br><br>0 Disables the SLEEP interrupt.<br>1 Enables the SLEEP interrupt. |
| 3<br>TOKDNEEN | TOKDNE Interrupt Enable<br><br>0 Disables the TOKDNE interrupt.<br>1 Enables the TOKDNE interrupt. |
| 2<br>SOFTOKEN | SOFTOK Interrupt Enable |

*Table continues on the next page...*

**USBx_INTEN field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Disbles the SOFTOK interrupt.<br>1 Enables the SOFTOK interrupt. |
| 1<br>ERROREN | ERROR Interrupt Enable<br><br>0 Disables the ERROR interrupt.<br>1 Enables the ERROR interrupt. |
| 0<br>USBRSTEN | USBRST Interrupt Enable<br><br>0 Disables the USBRST interrupt.<br>1 Enables the USBRST interrupt. |

## 45.4.11 Error Interrupt Status register (USBx_ERRSTAT)

Contains enable bits for each of the error sources within the USB Module. Each of these bits are qualified with their respective error enable bits. All bits of this register are logically OR'd together and the result placed in the ERROR bit of the ISTAT register. After an interrupt bit has been set it may only be cleared by writing a one to the respective interrupt bit. Each bit is set as soon as the error conditions is detected. Therefore, the interrupt does not typically correspond with the end of a token being processed. This register contains the value of 0x00 after a reset.

Address: 4007_2000h base + 88h offset = 4007_2088h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | BTSERR | 0 | DMAERR | BTOERR | DFN8 | CRC16 | CRC5EOF | PIDERR |
| Write | w1c | | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**USBx_ERRSTAT field descriptions**

| Field | Description |
|---|---|
| 7<br>BTSERR | This bit is set when a bit stuff error is detected. If set, the corresponding packet is rejected due to the error. |
| 6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5<br>DMAERR | This bit is set if the USB Module has requested a DMA access to read a new BDT but has not been given the bus before it needs to receive or transmit data. If processing a TX transfer this would cause a transmit data underflow condition. If processing a RX transfer this would cause a receive data overflow condition. This interrupt is useful when developing device arbitration hardware for the microprocessor and the USB module to minimize bus request and bus grant latency. This bit is also set if a data packet to or from the host is larger than the buffer size allocated in the BDT. In this case the data packet is truncated as it is put in buffer memory. |
| 4<br>BTOERR | This bit is set when a bus turnaround timeout error occurs. The USB module contains a bus turnaround timer that keeps track of the amount of time elapsed between the token and data phases of a SETUP or |

*Table continues on the next page...*

**USBx_ERRSTAT field descriptions (continued)**

| Field | Description |
|---|---|
| | OUT TOKEN or the data and handshake phases of a IN TOKEN. If more than 16 bit times are counted from the previous EOP before a transition from IDLE, a bus turnaround timeout error occurs. |
| 3<br>DFN8 | This bit is set if the data field received was not 8 bits in length. USB Specification 1.0 requires that data fields be an integral number of bytes. If the data field was not an integral number of bytes, this bit is set. |
| 2<br>CRC16 | This bit is set when a data packet is rejected due to a CRC16 error. |
| 1<br>CRC5EOF | This error interrupt has two functions. When the USB Module is operating in peripheral mode (HOSTMODEEN=0), this interrupt detects CRC5 errors in the token packets generated by the host. If set the token packet was rejected due to a CRC5 error.<br>When the USB Module is operating in host mode (HOSTMODEEN=1), this interrupt detects End Of Frame (EOF) error conditions. This occurs when the USB Module is transmitting or receiving data and the SOF counter reaches zero. This interrupt is useful when developing USB packet scheduling software to ensure that no USB transactions cross the start of the next frame. |
| 0<br>PIDERR | This bit is set when the PID check field fails. |

## 45.4.12 Error Interrupt Enable register (USBx_ERREN)

Contains enable bits for each of the error interrupt sources within the USB module. Setting any of these bits enables the respective interrupt source in ERRSTAT. Each bit is set as soon as the error conditions is detected. Therefore, the interrupt does not typically correspond with the end of a token being processed. This register contains the value of 0x00 after a reset.

Address: 4007_2000h base + 8Ch offset = 4007_208Ch

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | BTSERREN | 0 | DMAERREN | BTOERREN | DFN8EN | CRC16EN | CRC5EOFEN | PIDERREN |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**USBx_ERREN field descriptions**

| Field | Description |
|---|---|
| 7<br>BTSERREN | BTSERR Interrupt Enable<br>0 Disables the BTSERR interrupt.<br>1 Enables the BTSERR interrupt. |
| 6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5<br>DMAERREN | DMAERR Interrupt Enable<br>0 Disables the DMAERR interrupt.<br>1 Enables the DMAERR interrupt. |

*Table continues on the next page...*

**USBx_ERREN field descriptions (continued)**

| Field | Description |
|---|---|
| 4<br>BTOERREN | BTOERR Interrupt Enable<br><br>0 Disables the BTOERR interrupt.<br>1 Enables the BTOERR interrupt. |
| 3<br>DFN8EN | DFN8 Interrupt Enable<br><br>0 Disables the DFN8 interrupt.<br>1 Enables the DFN8 interrupt. |
| 2<br>CRC16EN | CRC16 Interrupt Enable<br><br>0 Disables the CRC16 interrupt.<br>1 Enables the CRC16 interrupt. |
| 1<br>CRC5EOFEN | CRC5/EOF Interrupt Enable<br><br>0 Disables the CRC5/EOF interrupt.<br>1 Enables the CRC5/EOF interrupt. |
| 0<br>PIDERREN | PIDERR Interrupt Enable<br><br>0 Disables the PIDERR interrupt.<br>1 Enters the PIDERR interrupt. |

## 45.4.13 Status register (USBx_STAT)

Reports the transaction status within the USB module. When the processor's interrupt controller has received a TOKDNE, interrupt the Status Register must be read to determine the status of the previous endpoint communication. The data in the status register is valid when TOKDNE interrupt is asserted. The Status register is actually a read window into a status FIFO maintained by the USB module. When the USB module uses a BD, it updates the Status register. If another USB transaction is performed before the TOKDNE interrupt is serviced, the USB module stores the status of the next transaction in the STAT FIFO. Thus STAT is actually a four byte FIFO that allows the processor core to process one transaction while the SIE is processing the next transaction. Clearing the TOKDNE bit in the ISTAT register causes the SIE to update STAT with the contents of the next STAT value. If the data in the STAT holding register is valid, the SIE immediately reasserts to TOKDNE interrupt.

Address: 4007_2000h base + 90h offset = 4007_2090h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | ENDP | | | | TX | ODD | 0 | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**USBx_STAT field descriptions**

| Field | Description |
|---|---|
| 7–4<br>ENDP | This four-bit field encodes the endpoint address that received or transmitted the previous token. This allows the processor core to determine the BDT entry that was updated by the last USB transaction. |
| 3<br>TX | Transmit Indicator<br>0 The most recent transaction was a receive operation.<br>1 The most recent transaction was a transmit operation. |
| 2<br>ODD | This bit is set if the last buffer descriptor updated was in the odd bank of the BDT. |
| 1–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 45.4.14 Control register (USBx_CTL)

Provides various control and configuration information for the USB module.

Address: 4007_2000h base + 94h offset = 4007_2094h

| Bit | 7 | 6 | 5 | 4 |
|---|---|---|---|---|
| Read<br>Write | JSTATE | SE0 | TXSUSPENDTOKENBUSY | RESET |
| Reset | 0 | 0 | 0 | 0 |

| Bit | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| Read<br>Write | HOSTMODEEN | RESUME | ODDRST | USBENSOFEN |
| Reset | 0 | 0 | 0 | 0 |

**USBx_CTL field descriptions**

| Field | Description |
|---|---|
| 7<br>JSTATE | Live USB differential receiver JSTATE signal<br>The polarity of this signal is affected by the current state of LSEN . |
| 6<br>SE0 | Live USB Single Ended Zero signal |
| 5<br>TXSUSPENDTOKENBUSY | In Host mode, TOKEN_BUSY is set when the USB module is busy executing a USB token. Software must not write more token commands to the Token Register when TOKEN_BUSY is set.. Software should check this field before writing any tokens to the Token Register to ensure that token commands are not lost.<br>In Target mode, TXD_SUSPEND is set when the SIE has disabled packet transmission and reception. Clearing this bit allows the SIE to continue token processing. This bit is set by the SIE when a SETUP Token is received allowing software to dequeue any pending packet transactions in the BDT before resuming token processing. |
| 4<br>RESET | Setting this bit enables the USB Module to generate USB reset signaling. This allows the USB Module to reset USB peripherals. This control signal is only valid in Host mode (HOSTMODEEN=1). Software must set RESET to 1 for the required amount of time and then clear it to 0 to end reset signaling. For more information on reset signaling see Section 7.1.4.3 of the USB specification version 1.0. |

*Table continues on the next page...*

**USBx_CTL field descriptions (continued)**

| Field | Description |
|---|---|
| 3<br>HOSTMODEEN | When set to 1, this bit enables the USB Module to operate in Host mode. In host mode, the USB module performs USB transactions under the programmed control of the host processor. |
| 2<br>RESUME | When set to 1 this bit enables the USB Module to execute resume signaling. This allows the USB Module to perform remote wake-up. Software must set RESUME to 1 for the required amount of time and then clear it to 0. If the HOSTMODEEN bit is set, the USB module appends a Low Speed End of Packet to the Resume signaling when the RESUME bit is cleared. For more information on RESUME signaling see Section 7.1.4.5 of the USB specification version 1.0. |
| 1<br>ODDRST | Setting this bit to 1 resets all the BDT ODD ping/pong fields to 0, which then specifies the EVEN BDT bank. |
| 0<br>USBENSOFEN | USB Enable<br><br>Setting this bit causes the SIE to reset all of its ODD bits to the BDTs. Therefore, setting this bit resets much of the logic in the SIE. When host mode is enabled, clearing this bit causes the SIE to stop sending SOF tokens.<br><br>0 Disables the USB Module.<br>1 Enables the USB Module. |

## 45.4.15 Address register (USBx_ADDR)

Holds the unique USB address that the USB module decodes when in Peripheral mode (HOSTMODEEN=0). When operating in Host mode (HOSTMODEEN=1) the USB module transmits this address with a TOKEN packet. This enables the USB module to uniquely address an USB peripheral. In either mode, USB_EN within the control register must be 1. The Address register is reset to 0x00 after the reset input becomes active or the USB module decodes a USB reset signal. This action initializes the Address register to decode address 0x00 as required by the USB specification.

Address: 4007_2000h base + 98h offset = 4007_2098h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | LSEN | ADDR | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**USBx_ADDR field descriptions**

| Field | Description |
|---|---|
| 7<br>LSEN | Low Speed Enable bit<br><br>Informs the USB module that the next token command written to the token register must be performed at low speed. This enables the USB module to perform the necessary preamble required for low-speed data transmissions. |
| 6–0<br>ADDR | USB Address<br><br>Defines the USB address that the USB module decodes in peripheral mode, or transmits when in host mode. |

## 45.4.16 BDT Page Register 1 (USBx_BDTPAGE1)

Provides address bits 15 through 9 of the base address where the current Buffer Descriptor Table (BDT) resides in system memory. The 32-bit BDT Base Address is always aligned on 512-byte boundaries, so bits 8 through 0 of the base address are always zero.

Address: 4007_2000h base + 9Ch offset = 4007_209Ch

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | BDTBA | | | | | | | 0 |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**USBx_BDTPAGE1 field descriptions**

| Field | Description |
|---|---|
| 7–1<br>BDTBA | Provides address bits 15 through 9 of the BDT base address. |
| 0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 45.4.17 Frame Number Register Low (USBx_FRMNUML)

Contains an 11-bit value used to compute the address where the current Buffer Descriptor Table (BDT) resides in system memory.

Address: 4007_2000h base + A0h offset = 4007_20A0h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | FRM[7:0] | | | | | | | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**USBx_FRMNUML field descriptions**

| Field | Description |
|---|---|
| 7–0<br>FRM[7:0] | This 8-bit field and the 3-bit field in the Frame Number Register High are used to compute the address where the current Buffer Descriptor Table (BDT) resides in system memory. |

## 45.4.18 Frame Number Register High (USBx_FRMNUMH)

Contains an 11-bit value used to compute the address where the current Buffer Descriptor Table (BDT) resides in system memory.

Address: 4007_2000h base + A4h offset = 4007_20A4h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | | | | | FRM[10:8] | | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**USBx_FRMNUMH field descriptions**

| Field | Description |
|---|---|
| 7–3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2–0<br>FRM[10:8] | This 3-bit field and the 8-bit field in the Frame Number Register Low are used to compute the address where the current Buffer Descriptor Table (BDT) resides in system memory. |

## 45.4.19 Token register (USBx_TOKEN)

Used to initiate USB transactions when in host mode (HOSTMODEEN=1). When the software needs to execute a USB transaction to a peripheral, it writes the TOKEN type and endpoint to this register. After this register has been written, the USB module begins the specified USB transaction to the address contained in the address register. The processor core must always check that the TOKEN_BUSY bit in the control register is not 1 before writing to the Token Register. This ensures that the token commands are not overwritten before they can be executed. The address register and endpoint control register 0 are also used when performing a token command and therefore must also be written before the Token Register. The address register is used to select the USB peripheral address transmitted by the token command. The endpoint control register determines the handshake and retry policies used during the transfer.

Address: 4007_2000h base + A8h offset = 4007_20A8h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | TOKENPID | | | | TOKENENDPT | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**USBx_TOKEN field descriptions**

| Field | Description |
|---|---|
| 7–4<br>TOKENPID | Contains the token type executed by the USB module. |

*Table continues on the next page...*

**USBx_TOKEN field descriptions (continued)**

| Field | Description |
|---|---|
| | 0001 OUT Token. USB Module performs an OUT (TX) transaction.<br>1001 IN Token. USB Module performs an In (RX) transaction.<br>1101 SETUP Token. USB Module performs a SETUP (TX) transaction |
| 3–0<br>TOKENENDPT | Holds the Endpoint address for the token command. The four bit value written must be a valid endpoint. |

## 45.4.20 SOF Threshold Register (USBx_SOFTHLD)

The SOF Threshold Register is used only in Host mode (HOSTMODEEN=1). When in Host mode, the 14-bit SOF counter counts the interval between SOF frames. The SOF must be transmitted every 1ms so therefore the SOF counter is loaded with a value of 12000. When the SOF counter reaches zero, a Start Of Frame (SOF) token is transmitted.

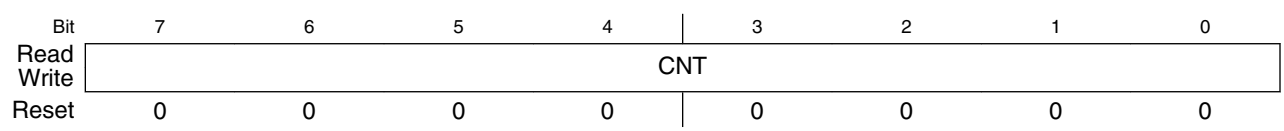
The SOF threshold register is used to program the number of USB byte times before the SOF to stop initiating token packet transactions. This register must be set to a value that ensures that other packets are not actively being transmitted when the SOF time counts to zero. When the SOF counter reaches the threshold value, no more tokens are transmitted until after the SOF has been transmitted.

The value programmed into the threshold register must reserve enough time to ensure the worst case transaction completes. In general the worst case transaction is an IN token followed by a data packet from the target followed by the response from the host. The actual time required is a function of the maximum packet size on the bus.

Typical values for the SOF threshold are:

- 64-byte packets=74;
- 32-byte packets=42;
- 16-byte packets=26;
- 8-byte packets=18.

Address: 4007_2000h base + ACh offset = 4007_20ACh




**USBx_SOFTHLD field descriptions**

| Field | Description |
|---|---|
| 7–0<br>CNT | Represents the SOF count threshold in byte times. |

## 45.4.21 BDT Page Register 2 (USBx_BDTPAGE2)

Contains an 8-bit value used to compute the address where the current Buffer Descriptor Table (BDT) resides in system memory.

Address: 4007_2000h base + B0h offset = 4007_20B0h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | BDTBA | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**USBx_BDTPAGE2 field descriptions**

| Field | Description |
|---|---|
| 7–0<br>BDTBA | Provides address bits 23 through 16 of the BDT base address that defines the location of Buffer Descriptor Table resides in system memory. |

## 45.4.22 BDT Page Register 3 (USBx_BDTPAGE3)

Contains an 8-bit value used to compute the address where the current Buffer Descriptor Table (BDT) resides in system memory.

Address: 4007_2000h base + B4h offset = 4007_20B4h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | BDTBA | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**USBx_BDTPAGE3 field descriptions**

| Field | Description |
|---|---|
| 7–0<br>BDTBA | Provides address bits 31 through 24 of the BDT base address that defines the location of Buffer Descriptor Table resides in system memory. |

## 45.4.23 Endpoint Control register (USBx_ENDPT*n*)

Contains the endpoint control bits for each of the 16 endpoints available within the USB module for a decoded address. The format for these registers is shown in the following figure. Endpoint 0 (ENDPT0) is associated with control pipe 0, which is required for all USB functions. Therefore, after a USBRST interrupt occurs the processor core should set ENDPT0 to contain 0x0D.

In Host mode ENDPT0 is used to determine the handshake, retry and low speed characteristics of the host transfer. For Control, Bulk and Interrupt transfers, the EPHSHK bit should be 1. For Isochronous transfers it should be 0. Common values to use for ENDPT0 in host mode are 0x4D for Control, Bulk, and Interrupt transfers, and 0x4C for Isochronous transfers.

Address: 4007_2000h base + C0h offset + (4d × i), where i=0d to 15d

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | HOSTWOHUB | RETRYDIS | 0 | EPCTLDIS | EPRXEN | EPTXEN | EPSTALL | EPHSHK |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**USBx_ENDPTn field descriptions**

| Field | Description |
|---|---|
| 7<br>HOSTWOHUB | This is a Host mode only field and is present in the control register for endpoint 0 (ENDPT0) only. When set this bit allows the host to communicate to a directly connected low speed device. When cleared, the host produces the PRE_PID. It then switches to low-speed signaling when sends a token to a low speed device as required to communicate with a low speed device through a hub. |
| 6<br>RETRYDIS | This is a Host mode only bit and is present in the control register for endpoint 0 (ENDPT0) only. When set this bit causes the host to not retry NAK'ed (Negative Acknowledgement) transactions. When a transaction is NAKed, the BDT PID field is updated with the NAK PID, and the TOKEN_DNE interrupt is set. When this bit is cleared NAKed transactions is retried in hardware. This bit must be set when the host is attempting to poll an interrupt endpoint. |
| 5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>EPCTLDIS | This bit, when set, disables control (SETUP) transfers. When cleared, control transfers are enabled. This applies if and only if the EPRXEN and EPTXEN bits are also set. |
| 3<br>EPRXEN | This bit, when set, enables the endpoint for RX transfers. |
| 2<br>EPTXEN | This bit, when set, enables the endpoint for TX transfers. |
| 1<br>EPSTALL | When set this bit indicates that the endpoint is called. This bit has priority over all other control bits in the EndPoint Enable Register, but it is only valid if EPTXEN=1 or EPRXEN=1. Any access to this endpoint causes the USB Module to return a STALL handshake. After an endpoint is stalled it requires intervention from the Host Controller. |
| 0<br>EPHSHK | When set this bit enables an endpoint to perform handshaking during a transaction to this endpoint. This bit is generally 1 unless the endpoint is Isochronous. |

## 45.4.24 USB Control register (USBx_USBCTRL)

Address: 4007_2000h base + 100h offset = 4007_2100h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | SUSP | PDE | 0 | | | | | |
| Reset | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

**USBx_USBCTRL field descriptions**

| Field | Description |
|---|---|
| 7<br>SUSP | Places the USB transceiver into the suspend state.<br><br>0 USB transceiver is not in suspend state.<br>1 USB transceiver is in suspend state. |
| 6<br>PDE | Enables the weak pulldowns on the USB transceiver.<br><br>0 Weak pulldowns are disabled on D+ and D–.<br>1 Weak pulldowns are enabled on D+ and D–. |
| 5–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 45.4.25 USB OTG Observe register (USBx_OBSERVE)

Provides visibility on the state of the pull-ups and pull-downs at the transceiver. Useful when interfacing to an external OTG control module via a serial interface.

Address: 4007_2000h base + 104h offset = 4007_2104h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | DPPU | DPPD | 0 | DMPD | 0 | | | 0 |
| Write | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

**USBx_OBSERVE field descriptions**

| Field | Description |
|---|---|
| 7<br>DPPU | Provides observability of the D+ Pullup . enable at the USB transceiver<br><br>0 D+ pullup disabled.<br>1 D+ pullup enabled. |
| 6<br>DPPD | Provides observability of the D+ Pulldown . enable at the USB transceiver<br><br>0 D+ pulldown disabled.<br>1 D+ pulldown enabled. |
| 5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>DMPD | Provides observability of the D Pulldown . enable at the USB transceiver<br><br>0 D– pulldown disabled.<br>1 D– pulldown enabled. |
| 3–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 45.4.26 USB OTG Control register (USBx_CONTROL)

Address: 4007_2000h base + 108h offset = 4007_2108h

| Bit | 7 | 6 | 5 | 4 |
|---|---|---|---|---|
| Read | 0 | | | DPPULLUPNONOTG |
| Write | | | | |
| Reset | 0 | 0 | 0 | 0 |

| Bit | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| Read | 0 | | | |
| Write | | | | |
| Reset | 0 | 0 | 0 | 0 |

### USBx_CONTROL field descriptions

| Field | Description |
|---|---|
| 7–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>DPPULLUPNONOTG | Provides control of the DP Pullup in the USB OTG module, if USB is configured in non-OTG device mode.<br><br>0 DP Pullup in non-OTG device mode is not enabled.<br>1 DP Pullup in non-OTG device mode is enabled. |
| 3–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 45.4.27 USB Transceiver Control Register 0 (USBx_USBTRC0)

Address: 4007_2000h base + 10Ch offset = 4007_210Ch

| Bit | 7 | 6 | 5 | 4 |
|---|---|---|---|---|
| Read | | 0 | USBRESMEN | 0 |
| Write | USBRESET | | | |
| Reset | 0 | 0 | 0 | 0 |

| Bit | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| Read | 0 | | SYNC_DET | USB_RESUME_INT |
| Write | | | | |
| Reset | 0 | 0 | 0 | 0 |

### USBx_USBTRC0 field descriptions

| Field | Description |
|---|---|
| 7<br>USBRESET | USB Reset<br><br>Generates a hard reset to the USB_OTG module. After this bit is set and the reset occurs, this bit is automatically cleared. |

*Table continues on the next page...*

**USBx_USBTRC0 field descriptions (continued)**

| Field | Description |
|---|---|
| | **NOTE: This bit is always read as zero. Wait two USB clock cycles after setting this bit.**<br><br>0 Normal USB module operation.<br>1 Returns the USB module to its reset state. |
| 6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5<br>USBRESMEN | Asynchronous Resume Interrupt Enable<br><br>This bit, when set, allows the USB module to send an asynchronous wakeup event to the MCU upon detection of resume signaling on the USB bus. The MCU then re-enables clocks to the USB module. It is used for low-power suspend mode when USB module clocks are stopped or the USB transceiver is in Suspend mode. Async wakeup only works in device mode.<br><br>0 USB asynchronous wakeup from suspend mode disabled.<br>1 USB asynchronous wakeup from suspend mode enabled. The asynchronous resume interrupt differs from the synchronous resume interrupt in that it asynchronously detects K-state using the unfiltered state of the D+ and D– pins. This interupt should only be enabled when the Transceiver is suspended. |
| 4–2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 1<br>SYNC_DET | Synchronous USB Interrupt Detect<br><br>0 Synchronous interrupt has not been detected.<br>1 Synchronous interrupt has been detected. |
| 0<br>USB_RESUME_INT | USB Asynchronous Interrupt<br><br>0 No interrupt was generated.<br>1 Interrupt was generated because of the USB asynchronous interrupt. |

## 45.4.28 Frame Adjust Register (USBx_USBFRMADJUST)

Address: 4007_2000h base + 114h offset = 4007_2114h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | ADJ | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**USBx_USBFRMADJUST field descriptions**

| Field | Description |
|---|---|
| 7–0<br>ADJ | Frame Adjustment<br><br>In Host mode, the frame adjustment is a twos complement number that adjusts the period of each USB frame in 12-MHz clock periods. A SOF is normally generated every 12,000 12-MHz clock cycles. The Frame Adjust Register can adjust this by -128 to +127 to compensate for inaccuracies in the USB 48-MHz clock. Changes to the ADJ bit take effect at the next start of the next frame. |

## 45.5 OTG and Host mode operation

The Host mode logic allows devices such as digital cameras and palmtop computers to function as a USB Host Controller. The OTG logic adds an interface to allow the OTG Host Negotiation and Session Request Protocols (HNP and SRP) to be implemented in software. Host Mode allows a peripheral such as a digital camera to be connected directly to a USB compliant printer. Digital photos can then be easily printed without having to upload them to a PC. In the palmtop computer application, a USB compliant keyboard/ mouse can be connected to the palmtop computer with the obvious advantages of easier interaction.

Host mode is intended for use in handheld-portable devices to allow easy connection to simple HID class devices such as printers and keyboards. It is not intended to perform the functions of a full OHCI or UHCI compatible host controller found on PC motherboards. The USB-FS is not supported by Windows 98 as a USB host controller. Host mode allows bulk, isochronous, interrupt and control transfers. Bulk data transfers are performed at nearly the full USB interface bandwidth. Support is provided for ISO transfers, but the number of ISO streams that can be practically supported is affected by the interrupt latency of the processor servicing the token during interrupts from the SIE. Custom drivers must be written to support Host mode operation.

Setting the HOST_MODE_EN bit in the CTL register enables Host mode. The USB-FS core can only operate as a peripheral device or in Host mode. It cannot operate in both modes simultaneously. When HOST_MODE is enabled, only endpoint zero is used. All other endpoints should be disabled by software.

## 45.6 Host Mode Operation Examples

The following sections illustrate the steps required to perform USB host functions using the USB-FS core. While it is useful to understand the interaction of the hardware and the software at a detailed level, an understanding of the interactions at this level is not required to write host applications using the API software.

To enable host mode and discover a connected device:

1. Enable Host Mode (CTL[HOST_MODE_EN]=1). The pull-down resistors are enabled, and pull-up disabled. Start of Frame (SOF) generation begins. SOF counter loaded with 12,000. Disable SOF packet generation to eliminate noise on the USB by writing the USB enable bit to 0 (CTL[USB_EN]=0).

2. Enable the ATTACH interrupt (INT_ENB[ATTACH]=1).
3. Wait for ATTACH interrupt (INT_STAT[ATTACH]). Signaled by USB Target pull-up resistor changing the state of DPLUS or DMINUS from 0 to 1 (SE0 to J or K state).
4. Check the state of the JSTATE and SE0 bits in the control register. If the connecting device is low speed (JSTATE bit is 0), set the low-speed bit in the address registers (ADDR[LS_EN]=1) and the Host Without Hub bit in endpoint 0 register control (ENDPT0[HOSTWOHUB]=1).
5. Enable RESET (CTL[RESET]=1) for 10 ms.
6. Enable SOF packet to keep the connected device from going to suspend (CTL[USB_EN=1]).
7. Start enumeration by sending a sequence of device framework commands, device frame work packets to the default control pipe of the connected device. See the *Universal Serial Bus Revision 2.0 specification,* "Chapter 9 USB Device Framework" (http://www.usb.org/developers/docs).

To complete a control transaction to a connected device:

1. Complete all the steps to discover a connected device
2. Set up the endpoint control register for bidirectional control transfers ENDPT0[4:0] = 0x0d.
3. Place a copy of the device framework setup command in a memory buffer. See the *Universal Serial Bus Revision 2.0 specification,* "Chapter 9 USB Device Framework" (http://www.usb.org/developers/docs).
4. Initialize current even or odd TX EP0 BDT to transfer the 8 bytes of command data for a device framework command (for example, a GET DEVICE DESCRIPTOR).
   - Set the BDT command word to 0x00080080 –Byte count to 8, OWN bit to 1.
   - Set the BDT buffer address field to the start address of the 8 byte command buffer.
5. Set the USB device address of the target device in the address register (ADDR[6:0]). After the USB bus reset, the device USB address is zero. It is set to some other value usually 1 by the Set Address device framework command.
6. Write the token register with a SETUP to Endpoint 0 the target device default control pipe (TOKEN=0xD0). This initiates a setup token on the bus followed by a data packet. The device handshake is returned in the BDT PID field after the packets

complete. When the BDT is written, a token done (ISTAT[TOKDNE]) interrupt is asserted. This completes the setup phase of the setup transaction. Se the *Universal Serial Bus Revision 2.0 specification,* "Chapter 9 USB Device Framework" (http://www.usb.org/developers/docs).

7. To initiate the data phase of the setup transaction (that is, get the data for the GET DEVICE DESCRIPTOR command), set up a buffer in memory for the data to be transferred.
8. Initialize the current even or odd TX EP0 BDT to transfer the data.
   - Set the BDT command word to 0x004000C0 – BC to 64 (the byte count of the data buffer in this case), OWN bit to 1, Data toggle to Data1.
   - Set the BDT buffer address field to the start address of the data buffer
9. Write the token register with a IN or OUT token to Endpoint 0 the target device default control pipe, an IN token for a GET DEVICE DESCRIPTOR command (TOKEN=0x90). This initiates an IN token on the bus followed by a data packet from the device to the host. When the data packet completes, the BDT is written and a token done (ISTAT[DNE]) interrupt is asserted. For control transfers with a single packet data phase this completes the data phase of the setup transaction. See the *Universal Serial Bus Revision 2.0 specification,* "Chapter 9 USB Device Framework" (http://www.usb.org/developers/docs).
10. To initiate the status phase of the setup transaction, set up a buffer in memory to receive or send the zero length status phase data packet.
11. Initialize the current even or odd TX EP0 BDT to transfer the status data.
    - Set the BDT command word to 0x00000080 – BC to 0 (the byte count of the data buffer in this case), OWN bit to 1, Data toggle to Data1.
    - Set the BDT buffer address field to the start address of the data buffer
12. Write the token register with a IN or OUT token to Endpoint 0 the target device default control pipe, an OUT token for a GET DEVICE DESCRIPTOR command (TOKEN=0x10). This initiates an OUT token on the bus followed by a zero length data packet from the host to the device. When the data packet completes, the BDT is written with the handshake from the device and a Token Done (ISTAT[TOKDNE]) interrupt is asserted. This completes the data phase of the setup transaction. See the *Universal Serial Bus Revision 2.0 specification,* "Chapter 9 USB Device Framework" (http://www.usb.org/developers/docs).

To send a full speed bulk data transfer to a target device:

1. Complete all steps to discover a connected device and to configure a connected device. Write the ADDR register with the address of the target device. Typically, there is only one other device on the USB bus in host mode so it is expected that the address is 0x01 and should remain constant.
2. Write 0x1D to ENDPT0 register to enable transmit and receive transfers with handshaking enabled.
3. Setup the even TX EP0 BDT to transfer up to 64 bytes.
4. Set the USB device address of the target device in the address register (ADDR[6:0]).
5. Write the TOKEN register with an OUT token to the desired endpoint. The write to this register triggers the USB-FS transmit state machines to begin transmitting the TOKEN and the data.
6. Setup the odd TX EP0 BDT to transfer up to 64 bytes.
7. Write the TOKEN register with an OUT token as in step 4. Two tokens can be queued at a time to allow the packets to be double buffered to achieve maximum throughput.
8. Wait for the TOKDNE interrupt. This indicates that one of the BDTs has been released back to the processor and the transfer has completed. If the target device asserts NAKs, the USB-FS continues to retry the transfer indefinitely without processor intervention unless the ENDPT0[RETRYDIS] is 1. If the retry disable field is set, the handshake (ACK, NAK, STALL, or ERROR (0xf)) is returned in the BDT PID field. If a stall interrupt occurs, the pending packet must be dequeued and the error condition in the target device cleared. If a Reset interrupt occurs (SE0 for more than 2.5 µs), the target has detached.
9. After the TOK_DNE interrupt occurs, the BDTs can be examined and the next data packet queued by returning to step 2.

## 45.7 On-The-Go operation

The USB-OTG core provides sensors and controls to enable On-The-Go (OTG) operation. These sensors are used by the OTG API software to implement the Host Negotiation Protocol (HNP) and Session Request Protocol (SRP). API calls are provided to give access the OTG protocol control signals, and include the OTG capabilities in the device application. The following state machines show the OTG operations involved with HNP and SRP protocols from either end of the USB cable.

## 45.7.1 OTG dual role A device operation

A device is considered the A device because of the type of cable attached. If the USB Type A connector or the USB Type Mini A connector is plugged into the device, it is considered the A device.

A dual role A device operates as the following flow diagram and state description table illustrates.

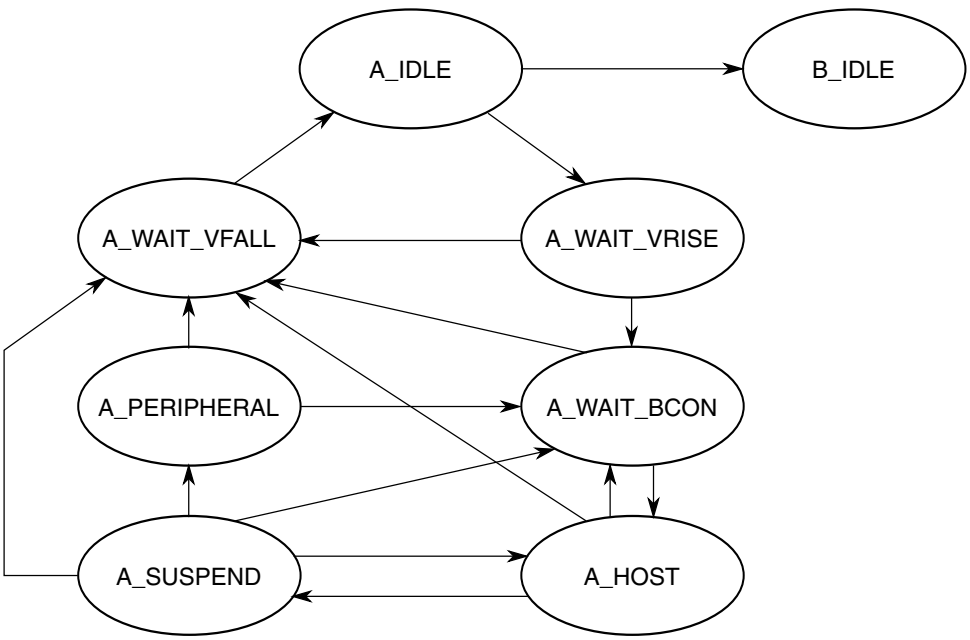



**Figure 45-93. Dual role A device flow diagram**

**Table 45-96. State descriptions for the dual role A device flow**

| State | Action | Response |
|---|---|---|
| A_IDLE | If ID Interrupt.<br>The cable has been un-plugged or a Type B cable has been attached. The device now acts as a Type B device. | Go to B_IDLE |
| | If the A application wants to use the bus or if the B device is doing an SRP as indicated by an A_SESS_VLD Interrupt or Attach or Port Status Change Interrupt check data line for 5 –10 msec pulsing. | Go to A_WAIT_VRISE<br>Turn on DRV_VBUS |
| A_WAIT_VRISE | If ID Interrupt or if A_VBUS_VLD is false after 100 msec<br>The cable has been changed or the A device cannot support the current required from the B device. | Go to A_WAIT_VFALL<br>Turn off DRV_VBUS |
| | If A_VBUS_VLD interrupt | Go to A_WAIT_BCON |

*Table continues on the next page...*

**Table 45-96. State descriptions for the dual role A device flow (continued)**

| State | Action | Response |
|---|---|---|
| A_WAIT_BCON | After 200 ms without Attach or ID Interrupt. (This could wait forever if desired.) | Go to A_WAIT_FALL<br>Turn off DRV_VBUS |
| | A_VBUS_VLD Interrupt and B device attaches | Go to A_HOST<br>Turn on Host mode |
| A_HOST | Enumerate Device determine OTG Support. | |
| | If A_VBUS_VLD/ Interrupt or A device is done and doesn't think he wants to do something soon or the B device disconnects | Go to A_WAIT_VFALL<br>Turn off Host mode<br>Turn off DRV_VBUS |
| | If the A device is finished with session or if the A device wants to allow the B device to take bus. | Go to A_SUSPEND |
| | ID Interrupt or the B device disconnects | Go to A_WAIT_BCON |
| A_SUSPEND | If ID Interrupt, or if 150 ms B disconnect timeout (This timeout value could be longer) or if A_VBUS_VLD\ Interrupt | Go to A_WAIT_VFALL<br>Turn off DRV_VBUS |
| | If HNP enabled, and B disconnects in 150 ms then B device is becoming the host. | Go to A_PERIPHERAL<br>Turn off Host mode |
| | If A wants to start another session | Go to A_HOST |
| A_PERIPHERAL | If ID Interrupt or if A_VBUS_VLD interrupt | Go to A_WAIT_VFALL<br>Turn off DRV_VBUS. |
| | If 3 –200 ms of Bus Idle | Go to A_WAIT_BCON<br>Turn on Host mode |
| A_WAIT_VFALL | If ID Interrupt or (A_SESS_VLD/ & b_conn/) | Go to A_IDLE |

## 45.7.2 OTG dual role B device operation

A device is considered a B device if it connected to the bus with a USB Type B cable or a USB Type Mini B cable.

A dual role B device operates as the following flow diagram and state description table illustrates.

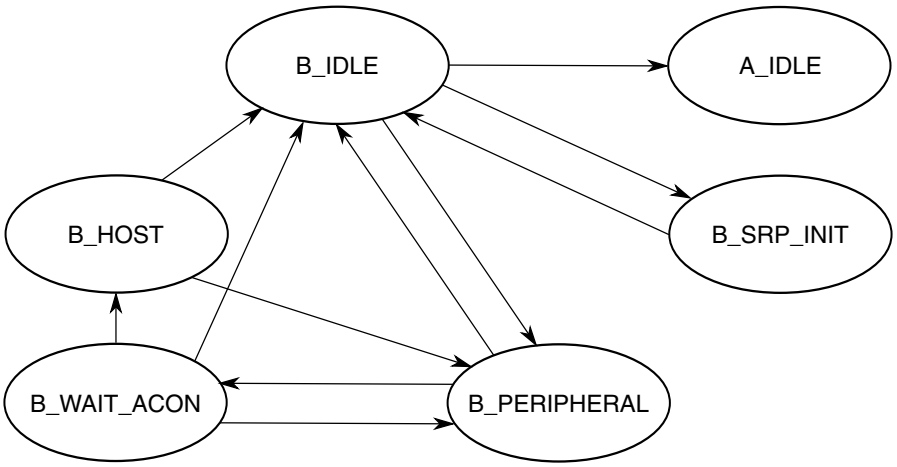



**Figure 45-94. Dual role B device flow diagram**

**Table 45-97. State descriptions for the dual role B device flow**

| State | Action | Response |
|---|---|---|
| B_IDLE | If ID\ Interrupt.<br>A Type A cable has been plugged in and the device should now respond as a Type A device. | Go to A_IDLE |
| | If B_SESS_VLD Interrupt.<br>The A device has turned on VBUS and begins a session. | Go to B_PERIPHERAL<br>Turn on DP_HIGH |
| | If B application wants the bus and Bus is Idle for 2 ms and the B_SESS_END bit is set, the B device can perform an SRP. | Go to B_SRP_INIT<br>Pulse CHRG_VBUS<br>Pulse DP_HIGH 5-10 ms |
| B_SRP_INIT | If ID\ Interrupt or SRP Done (SRP must be done in less than 100 ms.) | Go to B_IDLE |
| B_PERIPHERAL | If HNP enabled and the bus is suspended and B wants the bus, the B device can become the host. | Go to B_WAIT_ACON<br>Turn off DP_HIGH |
| B_WAIT_ACON | If A connects, an attach interrupt is received | Go to B_HOST<br>Turn on Host Mode |
| | If ID\ Interrupt or B_SESS_VLD/ Interrupt<br>If the cable changes or if VBUS goes away, the host doesn't support us.<br>Go to B_IDLE | Go to B_IDLE |
| | If 3.125 ms expires or if a Resume occurs | Go to B_PERIPHERAL |
| B_HOST | If ID\ Interrupt or B_SESS_VLD\ Interrupt<br>If the cable changes or if VBUS goes away, the host doesn't support us. | Go to B_IDLE |
| | If B application is done or A disconnects | Go to B_PERIPHERAL |

# Chapter 46
# USB Device Charger Detection Module (USBDCD)

## 46.1 Preface

### 46.1.1 References

The following publications are referenced in this document. For updates to these specifications, see http://www.usb.org.

- *USB Battery Charging Specification Revision 1.1, USB Implementers Forum*
- *USB Battery Charging Specification Revision 1.2, USB Implementers Forum*
- *Universal Serial Bus Specification Revision 2.0, USB Implementers Forum*

### 46.1.2 Acronyms and abbreviations

The following table contains acronyms and abbreviations used in this document.

**Table 46-1. Acronyms and abbreviated terms**

| Term | Meaning |
|---|---|
| FS | Full speed (12 Mbit/s) |
| HS | High speed (480 Mbit/s) |
| $I_{DEV\_DCHG}$ | Current drawn when the USB device is connected to a dedicated charging port |
| $I_{DEV\_HCHG\_LFS}$ | Current drawn when the USB device is connected to an FS charging host port |
| $I_{DM\_SINK}$ | Current sink for the D– line |
| $I_{DP\_SINK}$ | Current sink for the D+ line |
| $I_{DP\_SRC}$ | Current source for the D+ line |
| $I_{SUSP}$ | Current drawn when the USB device is suspended |
| LDO | Low dropout |
| LS | Low Speed (1.5 Mbit/s) |

*Table continues on the next page...*

**Table 46-1. Acronyms and abbreviated terms (continued)**

| Term | Meaning |
|---|---|
| N/A | Not applicable |
| OTG | On-The-Go |
| $R_{DM\_DWN}$ | D– pulldown resistance for data pin contact detect |
| $V_{DAT\_REF}$ | Data detect reference voltage for the voltage comparator |
| $V_{DP\_SRC}$ | Voltage source for the D+ line |
| $V_{DM\_SRC}$ | Voltage source for the D– line |
| $V_{LGC}$ | Threshold voltage for logic high |

### 46.1.3 Glossary

The following table shows a glossary of terms used in this document.

**Table 46-2. Glossary of terms**

| Term | Definition |
|---|---|
| Transceiver | Module that implements the physical layer of the USB standard (FS or LS only). |
| PHY | Module that implements the physical layer of the USB standard (HS capable). |
| Attached | Device is physically plugged into USB port, but has *not enabled* either D+ or D– pullup resistor. |
| Connected | Device is physically plugged into USB port, and has *enabled* either D+ or D– pullup resistor. |
| Suspended | After 3 ms of no bus activity, the USB device enters suspend mode. |
| Component | The hardware and software that make up a subsystem. |

## 46.2 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The USBDCD module works with the USB transceiver to detect whether the USB device is attached to a charging port, either a dedicated charging port or a charging host. System software coordinates the detection activites of the module and controls an off-chip integrated circuit that performs the battery charging.

## 46.2.1 Block diagram

The following figure is a high level block diagram of the module.

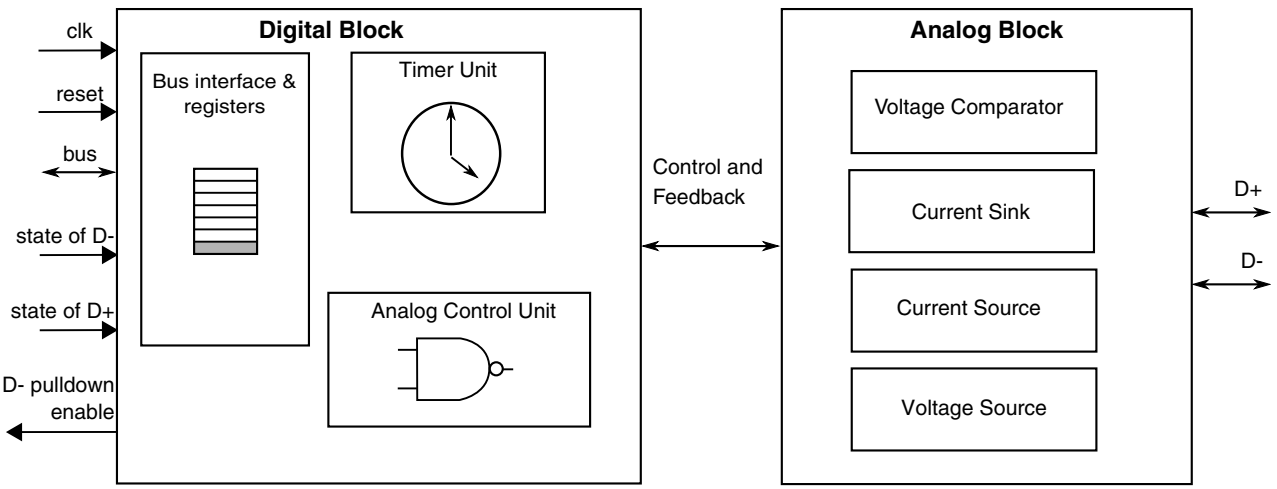



**Figure 46-1. Block diagram**

The USBDCD module consists of two main blocks:

- A digital block provides the programming interface (memory-mapped registers) and includes the timer unit and the analog control unit.
- An analog block provides the circuitry for the physical detection of the charger, including the voltage source, current source, current sink, and voltage comparator circuitry.

## 46.2.2 Features

The USBDCD module offers the following features:

- Compliant with the latest industry standard specification: *USB Battery Charging Specification, Revision 1.1 and 1.2*
- Programmable timing parameters default to values required by the industry standards:
  - Having standard default values allows for easy configuration- simply set the clock frequency before enabling the module.
  - Programmability allows the flexibility to meet future updates of the standards.

### 46.2.3 Modes of operation

The operating modes of the USBDCD module are shown in the following table.

**Table 46-3. Module modes and their conditions**

| Module mode | Description | Conditions when used |
|---|---|---|
| Enabled | The module performs the charger detection sequence. | System software should enable the module only when *all* of the following conditions are true:<br>• The system uses a rechargeable battery.<br>• The device is being used in an FS USB device application.<br>• The device has detected that it is attached to the USB cable. |
| Disabled | The module is not active and is held in a low power state. | System software should disable the module when *either* of the following conditions is true:<br>• The charger detect sequence is complete.<br>• The conditions for being enabled are not met. |
| Powered Off | The digital supply voltage dvdd is removed. | Low system performance requirements allow putting the device into a very low-power stop mode. |

Operating mode transitions are shown in the following table.

**Table 46-4. Entering and exiting module modes**

| Module mode | Entering | Exiting | Mode after exiting |
|---|---|---|---|
| Enabled | Set CONTROL[START]. | Set CONTROL[SR].[1] | Disabled |
| Disabled | Take *either* of the following actions:<br>• Set CONTROL[SR].[1]<br>• Reset the module. By default, the module is disabled. | Set CONTROL[START]. | Enabled |
| Powered Off | Perform the following actions:<br>1. Put the device into very low-power stop mode.<br>2. Adjust the supply voltages. | Perform the following actions:<br>1. Restore the supply voltages.<br>2. Take the device out of very low-power stop mode. | Disabled |

1. The effect of setting the SR bit is immediate; that is, the module is disabled even if the sequence has not completed.

## 46.3 Module signal descriptions

This section describes the module signals. The following table shows a summary of module signals that interface with the pins of the device.

**Table 46-5. Signal descriptions**

| Signal | Description | I/O |
|---|---|---|
| usb_dm | USB D– analog data signal. The analog block interfaces directly to the D– signal on the USB bus. | I/O |
| usb_dp | USB D+ analog data signal. The analog block interfaces directly to the D+ signal on the USB bus. | I/O |
| avdd33[1] | 3.3 V regulated analog supply | I |
| vss_bulk | Digital/Analog ground | I |
| dvdd | 1.2 V digital supply | I |

1. Voltage must be 3.3 V +/- 10% for full functionality of the module. That is, the charger detection function does not work when this voltage is below 3.0 V, and the CONTROL[START] bit should not be set.

**NOTE**

The transceiver module also interfaces to the usb_dm and usb_dp signals. Both modules and the USB host/hub use these signals as bidirectional, tristate signals.

Information about the signal integrity aspects of the lines including shielding, isolated return paths, input or output impedance, packaging, suggested external components, ESD, and other protections can be found in the USB 2.0 specification and in Application information.

## 46.4 Memory map/Register definition

This section describes the memory map and registers for the USBDCD module.

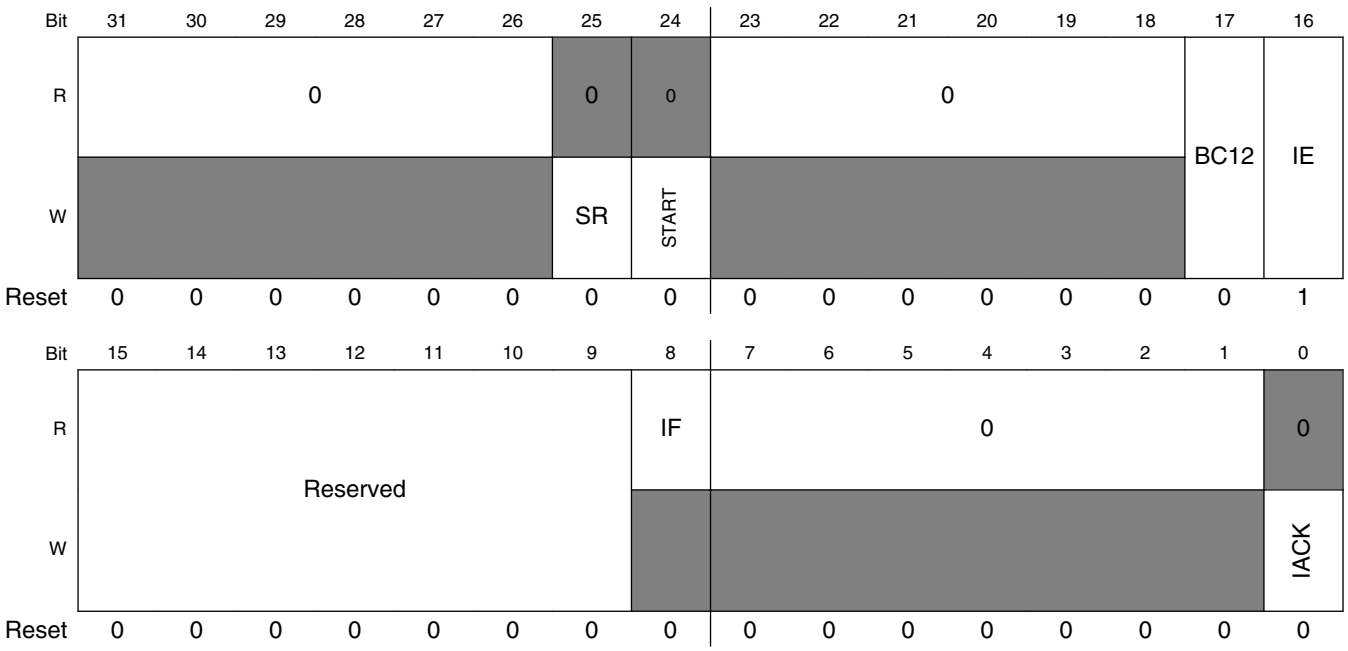
**USBDCD memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4003_5000 | Control register (USBDCD_CONTROL) | 32 | R/W | 0001_0000h | 46.4.1/1166 |
| 4003_5004 | Clock register (USBDCD_CLOCK) | 32 | R/W | 0000_00C1h | 46.4.2/1167 |
| 4003_5008 | Status register (USBDCD_STATUS) | 32 | R | 0000_0000h | 46.4.3/1169 |
| 4003_5010 | TIMER0 register (USBDCD_TIMER0) | 32 | R/W | 0010_0000h | 46.4.4/1170 |
| 4003_5014 | TIMER1 register (USBDCD_TIMER1) | 32 | R/W | 000A_0028h | 46.4.5/1171 |
| 4003_5018 | TIMER2_BC11 register (USBDCD_TIMER2_BC11) | 32 | R/W | 0028_0001h | 46.4.6/1172 |
| 4003_5018 | TIMER2_BC12 register (USBDCD_TIMER2_BC12) | 32 | R/W | 0001_0028h | 46.4.7/1173 |

## 46.4.1 Control register (USBDCD_CONTROL)

Contains the control and interrupt bit fields.

Address: 4003_5000h base + 0h offset = 4003_5000h




**USBDCD_CONTROL field descriptions**

| Field | Description |
|---|---|
| 31–26<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 25<br>SR | Software Reset<br><br>Determines whether a software reset is performed.<br><br>0 Do not perform a software reset.<br>1 Perform a software reset. |
| 24<br>START | Start Change Detection Sequence<br><br>Determines whether the charger detection sequence is initiated.<br><br>0 Do not start the sequence. Writes of this value have no effect.<br>1 Initiate the charger detection sequence. If the sequence is already running, writes of this value have no effect. |
| 23–18<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 17<br>BC12 | BC1.2 compatibility. This bit cannot be changed after start detection.<br><br>0 Compatible with BC1.1 (default)<br>1 Compatible with BC1.2 |

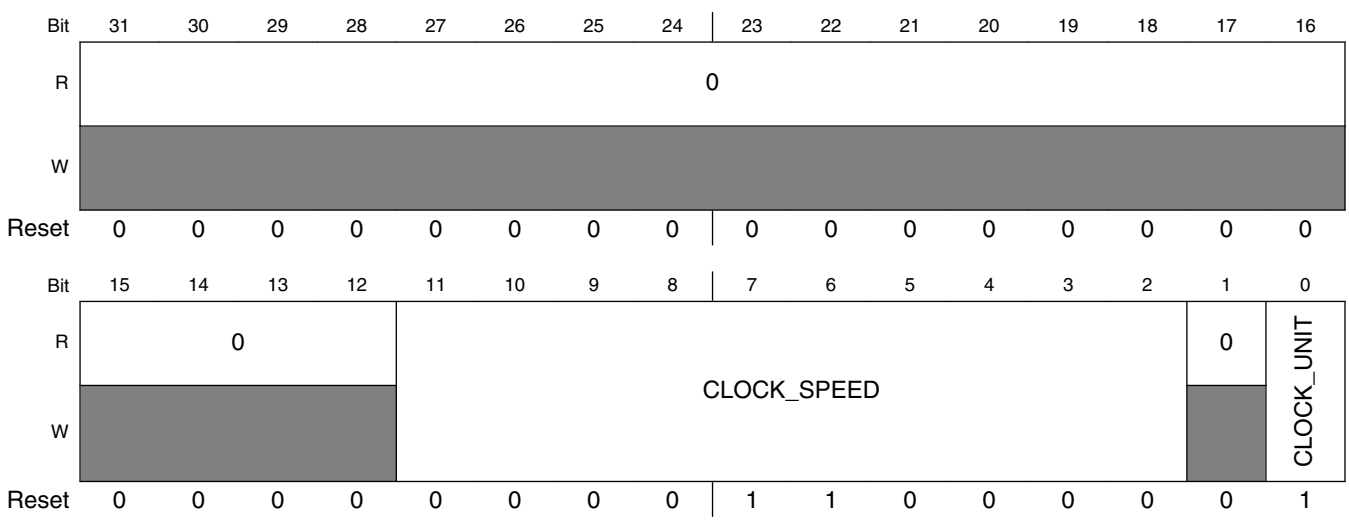
*Table continues on the next page...*

**USBDCD_CONTROL field descriptions (continued)**

| Field | Description |
|---|---|
| 16<br>IE | Interrupt Enable<br><br>Enables/disables interrupts to the system.<br><br>0 Disable interrupts to the system.<br>1 Enable interrupts to the system. |
| 15–9<br>Reserved | This field is reserved. |
| 8<br>IF | Interrupt Flag<br><br>Determines whether an interrupt is pending.<br><br>0 No interrupt is pending.<br>1 An interrupt is pending. |
| 7–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>IACK | Interrupt Acknowledge<br><br>Determines whether the interrupt is cleared.<br><br>0 Do not clear the interrupt.<br>1 Clear the IF bit (interrupt flag). |

## 46.4.2 Clock register (USBDCD_CLOCK)

Address: 4003_5000h base + 4h offset = 4003_5004h



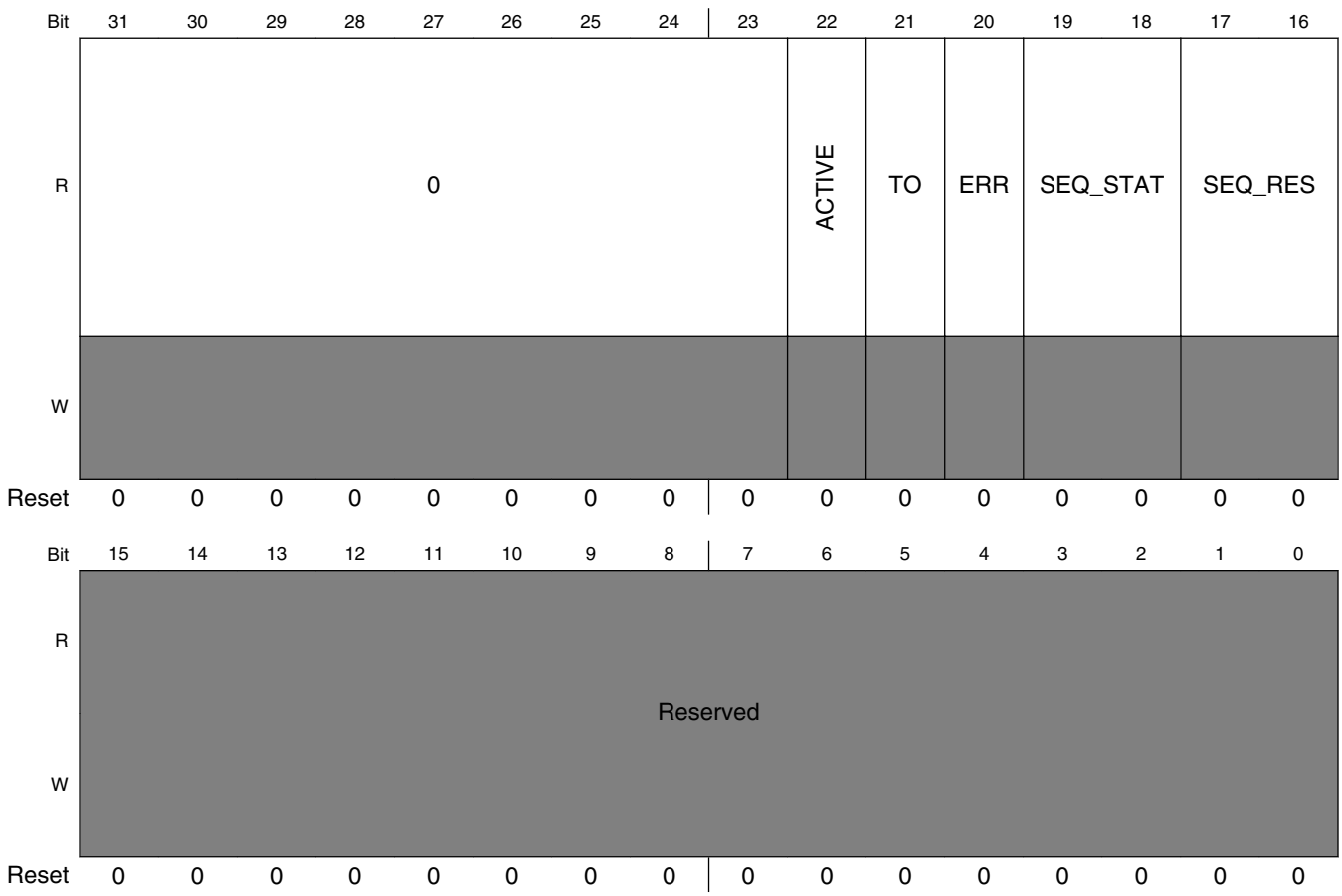
## USBDCD_CLOCK field descriptions

| Field | Description |
|---|---|
| 31–12<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 11–2<br>CLOCK_SPEED | Numerical Value of Clock Speed in Binary<br><br>The unit of measure is programmed in CLOCK_UNIT. The valid range is from 1 to 1023 when clock unit is MHz and 4 to 1023 when clock unit is kHz. Examples with CLOCK_UNIT = 1:<br>• For 48 MHz: 0b00_0011_0000 (48) (Default)<br>• For 24 MHz: 0b00_0001_1000 (24)<br><br>Examples with CLOCK_UNIT = 0:<br>• For 100 kHz: 0b00_0110_0100 (100)<br>• For 500 kHz: 0b01_1111_0100 (500) |
| 1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>CLOCK_UNIT | Unit of Measurement Encoding for Clock Speed<br><br>Specifies the unit of measure for the clock speed.<br><br>0 kHz Speed (between 1 kHz and 1023 kHz)<br>1 MHz Speed (between 1 MHz and 1023 MHz) |

## 46.4.3 Status register (USBDCD_STATUS)

Provides the current state of the module for system software monitoring.

Address: 4003_5000h base + 8h offset = 4003_5008h




### USBDCD_STATUS field descriptions

| Field | Description |
|---|---|
| 31–23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22<br>ACTIVE | Active Status Indicator<br><br>Indicates whether the sequence is running.<br><br>0 The sequence is not running.<br>1 The sequence is running. |
| 21<br>TO | Timeout Flag<br><br>Indicates whether the detection sequence has passed the timeout threshhold. |

*Table continues on the next page...*

**USBDCD_STATUS field descriptions (continued)**

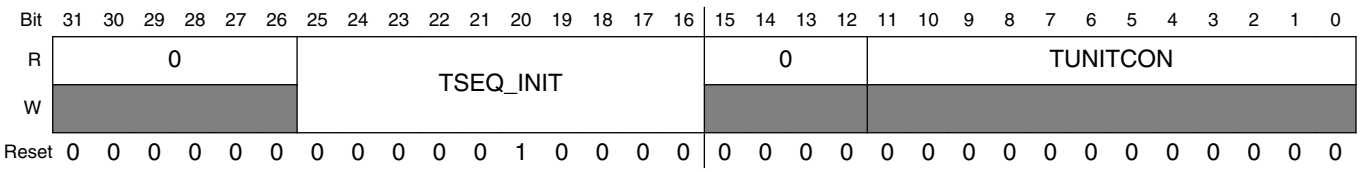
| Field | Description |
|---|---|
| | 0 The detection sequence has not been running for over 1 s.<br>1 It has been over 1 s since the data pin contact was detected and debounced. |
| 20<br>ERR | Error Flag<br><br>Indicates whether there is an error in the detection sequence.<br><br>0 No sequence errors.<br>1 Error in the detection sequence. See the SEQ_STAT field to determine the phase in which the error occurred. |
| 19–18<br>SEQ_STAT | Charger Detection Sequence Status<br><br>Indicates the status of the charger detection sequence.<br><br>00 The module is either not enabled, or the module is enabled but the data pins have not yet been detected.<br>01 Data pin contact detection is complete.<br>10 Charging port detection is complete.<br>11 Charger type detection is complete. |
| 17–16<br>SEQ_RES | Charger Detection Sequence Results<br><br>Reports how the charger detection is attached.<br><br>00 No results to report.<br>01 Attached to a standard host. Must comply with USB 2.0 by drawing only 2.5 mA (max) until connected.<br>10 Attached to a charging port. The exact meaning depends on bit 18:<br>• 0: Attached to either a charging host or a dedicated charger. The charger type detection has not completed.<br>• 1: Attached to a charging host. The charger type detection has completed.<br>11 Attached to a dedicated charger. |
| 15–0<br>Reserved | This field is reserved.<br><br>**NOTE:** Bits do not always read as 0. |

## 46.4.4 TIMER0 register (USBDCD_TIMER0)

TIMER0 has an TSEQ_INIT field that represents the system latency in ms. Latency is measured from the time when VBUS goes active until the time system software initiates charger detection sequence in USBDCD module. When software sets the CONTROL[START] bit, the Unit Connection Timer (TUNITCON) is initialized with the value of TSEQ_INIT.

Valid values are 0–1023, however the USB Battery Charging Specification requires the entire sequence, including TSEQ_INIT, to be completed in 1s or less.

Address: 4003_5000h base + 10h offset = 4003_5010h



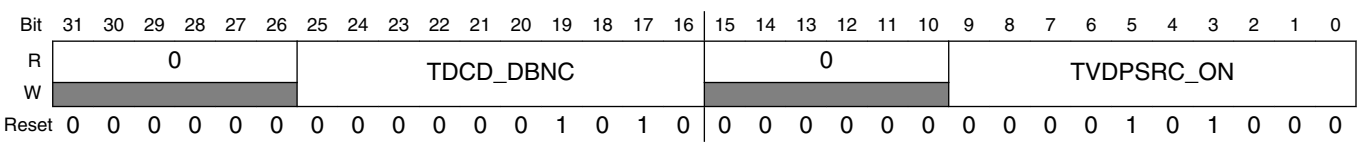

**USBDCD_TIMER0 field descriptions**

| Field | Description |
|---|---|
| 31–26<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 25–16<br>TSEQ_INIT | Sequence Initiation Time<br><br>TSEQ_INIT represents the system latency (in ms) measured from the time VBUS goes active to the time system software initiates the charger detection sequence in the USBDCD module. When software sets the CONTROL[START] bit, the Unit Connection Timer (TUNITCON) is initialized with the value of TSEQ_INIT. Valid values are 0-1023, but the USB Battery Charging Specification requires the entire sequence, including TSEQ_INIT, to be completed in 1s or less. |
| 15–12<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 11–0<br>TUNITCON | Unit Connection Timer Elapse (in ms)<br><br>Displays the amount of elapsed time since the event of setting the START bit plus the value of TSEQ_INIT. The timer is automatically initialized with the value of TSEQ_INIT before starting to count.<br><br>This timer enables compliance with the maximum time allowed to connect $T_{UNIT\_CON}$ under the USB Battery Charging Specification. If the timer reaches the one second limit, the module triggers an interrupt and sets the error flag STATUS[ERR].<br><br>The timer continues counting throughout the charger detection sequence, even when control has been passed to software. As long as the module is active, the timer continues to count until it reaches the maximum value of 0xFFF (4095 ms). The timer does not rollover to zero. A software reset clears the timer. |

## 46.4.5 TIMER1 register (USBDCD_TIMER1)

TIMER1 contains timing parameters. Note that register values can be written that are not compliant with the USB Battery Charging Specification, so care should be taken when overwriting the default values.

Address: 4003_5000h base + 14h offset = 4003_5014h




**USBDCD_TIMER1 field descriptions**

| Field | Description |
|---|---|
| 31–26<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

*Table continues on the next page...*

**USBDCD_TIMER1 field descriptions (continued)**

| Field | Description |
|---|---|
| 25–16<br>TDCD_DBNC | Time Period to Debounce D+ Signal<br><br>Sets the time period (ms) to debounce the D+ signal during the data pin contact detection phase. See "Debouncing the data pin contact"<br><br>Valid values are 1–1023, but the USB Battery Charging Specification requires a minimum value of 10 ms. |
| 15–10<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 9–0<br>TVDPSRC_ON | Time Period Comparator Enabled<br><br>This timing parameter is used after detection of the data pin. See "Charging Port Detection".<br><br>Valid values are 1–1023, but the USB Battery Charging Specification requires a minimum value of 40 ms. |

## 46.4.6 TIMER2_BC11 register (USBDCD_TIMER2_BC11)

TIMER2_BC11 contains timing parameters for USB Battery Charging Specification, v1.1.

**NOTE**

Register values can be written that are not compliant with the USB Battery Charging Specification, so care should be taken when overwriting the default values.

Address: 4003_5000h base + 18h offset = 4003_5018h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | TVDPSRC_CON | | | | | | | | | | 0 | | | | | | | | | | | | CHECK_DM | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**USBDCD_TIMER2_BC11 field descriptions**

| Field | Description |
|---|---|
| 31–26<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 25–16<br>TVDPSRC_CON | Time Period Before Enabling D+ Pullup<br><br>Sets the time period (ms) that the module waits after charging port detection before system software must enable the D+ pullup to connect to the USB host. Valid values are 1–1023, but the USB Battery Charging Specification requires a minimum value of 40 ms. |
| 15–4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3–0<br>CHECK_DM | Time Before Check of D– Line<br><br>Sets the amount of time (in ms) that the module waits after the device connects to the USB bus until checking the state of the D– line to determine the type of charging port. See "Charger Type Detection."<br>Valid values are 1–15ms. |

### 46.4.7 TIMER2_BC12 register (USBDCD_TIMER2_BC12)

TIMER2_BC12 contains timing parameters for USB Battery Charging Specification, v1.2.

**NOTE**

Register values can be written that are not compliant with the USB Battery Charging Specification, so care should be taken when overwriting the default values.

Address: 4003_5000h base + 18h offset = 4003_5018h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | TWAIT_AFTER_PRD | | | | | | | | | | 0 | | | | | | TVDMSRC_ON | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |

**USBDCD_TIMER2_BC12 field descriptions**

| Field | Description |
|---|---|
| 31–26<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 25–16<br>TWAIT_AFTER_PRD | Sets the amount of time (in ms) that the module waits after primary detection before start to secondary detection.<br>Valid values are 1–1023ms. Default is 1ms. |
| 15–10<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 9–0<br>TVDMSRC_ON | Sets the amount of time (in ms) that the module enables the $V_{DM\_SRC}$.<br>Valid values are 0–40ms. |

## 46.5 Functional description

The sequence of detecting the presence of charging port and type of charging port involves several hardware components, coordinated by system software. This collection of interacting hardware and software is called the USB Battery Charging Subsystem. The following figure shows the USBDCD module as a component of the subsystem. The following table describes the components.

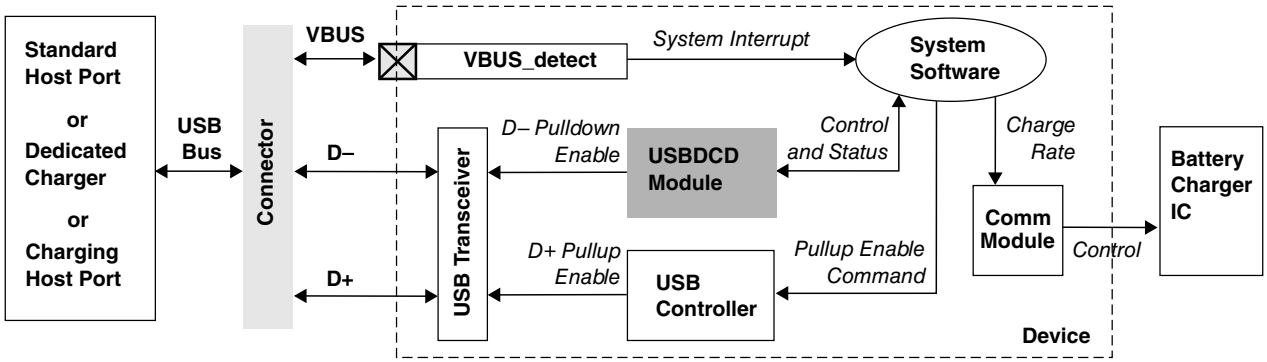



**Figure 46-9. USB battery charging subsystem**

**Table 46-14. USB battery charger subsystem components**

| Component | Description |
|---|---|
| Battery Charger IC | The external battery charger IC regulates the charge rate to the rechargable battery. System software is responsible for communicating the appropriate charge rates.<br><br>**Charger** / **Maximum current drawn**[1]<br>Standard host port — up to 500 mA<br>Charging host port — up to 1500 mA<br>Dedicated charging port — up to 1800 mA<br><br>1. If the USB host has suspended the USB device, system software must configure the system to limit the current drawn from the USB bus to 2.5 mA or less. |
| Comm Module | A communications module on the device can be used to control the charge rate of the battery charger IC. |
| System software | Coordinates the detection activities of the subsystem. |
| USB Controller | The D+ pullup enable control signal plays a role during the charger type detection phase. System software must issue a command to the USB controller to assert this signal. After this pullup is enabled, the device is considered to be connected to the USB bus. The host then attempts to enumerate it.<br><br>**NOTE:** The USB controller must be used only for USB device applications when using the USBDCD module. For USB host applications, the USBDCD module must be disabled. |
| USB Transceiver | The USB transceiver contains the pullup resistor for the USB D+ signal and the pulldown resistors for the USB D+ and D– signals. The D+ pullup and the D– pulldown are both used during the charger detection sequence in BC1.1, but it is not used during charger detection in BC1.2. The USB transceiver also outputs the digital state of the D+ and D– signals from the USB bus.<br><br>The pullup and pulldown enable signals are controlled by other modules during the charger detection sequence in BC1.1: The D+ pullup enable is output from the USB controller and is under software control. The USBDCD module controls the D– pulldown enable. |
| USBDCD Module | Detects whether the device has been plugged into either a standard host port, a charging host port, or a dedicated charger. |

*Table continues on the next page...*

**Table 46-14. USB battery charger subsystem components (continued)**

| Component | Description |
|---|---|
| VBUS_detect | This interrupt pin connected to the USB VBUS signal detects when the device has been plugged into or unplugged from the USB bus. If the system requires waking up from a low power mode on being plugged into the USB port, this interrupt should also be a low power wake up source. If this pin multiplexes other functions, such as GPIO, the pin can be configured as an interrupt so that the USB plug or unplug event can be detected. |

1. If the USB host has suspended the USB device, system software must configure the system to limit the current drawn from the USB bus to 2.5 mA or less.

## 46.5.1 The charger detection sequence

The following figure illustrates the charger detection sequence in a simplified timing diagram based on the USB Battery Charging Specification v1.2.

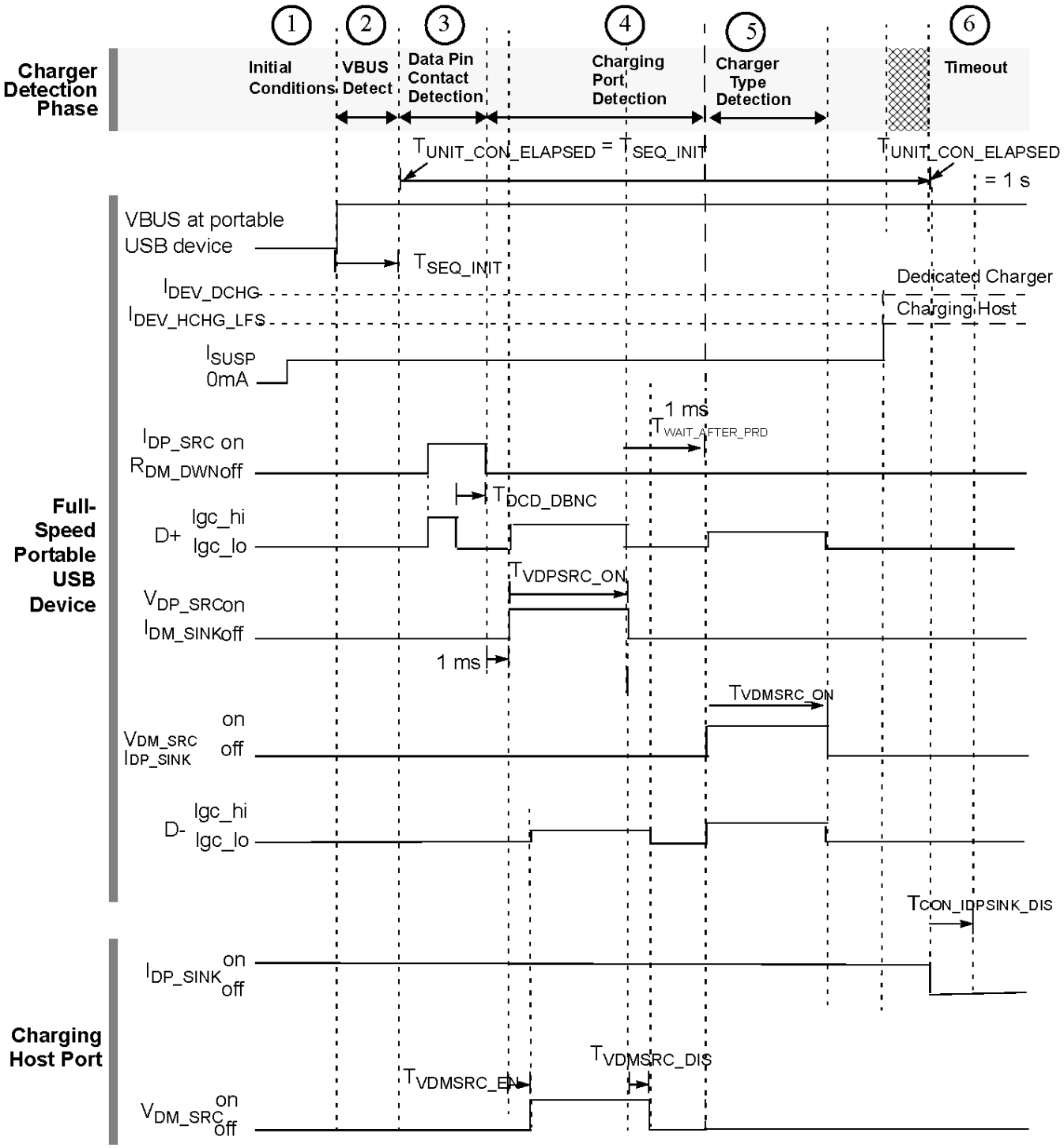



**Figure 46-10. Full speed charger detection timing for BC1.2**

Timing parameter values used in this module for BC1.2 are listed in the following table.

**Table 46-15. Timing parameters for the charger detection sequence for BC1.2**

| Parameter | USB Battery Charging Spec | Module default | Module programmable range |
|---|---|---|---|
| $T_{DCD\_DBNC}$[1] | 10 ms min (no max) | 10 ms | 0– 1023 ms |
| $T_{VDPSRC\_ON}$[1] | 40 ms min (no max) | 40 ms | 0 –1023 ms |
| $T_{WAIT\_AFTER\_PRD}$ | N/A | 1 ms | 0– 1023 ms |
| $T_{VDMSRC\_ON}$ | 40 ms | 40 ms | 0 –1023 ms |
| $T_{SEQ\_INIT}$ | N/A | 16 ms | 0 –1023 ms |
| $T_{UNIT\_CON}$[1] | 1 s | N/A | N/A |
| $T_{VDMSRC\_EN}$[1] | 1– 20 ms | From the USB host | N/A |
| $T_{VDMSRC\_DIS}$[1] | 0 – 20 ms | From the USB host | N/A |
| $T_{CON\_IDPSINK\_DIS}$[1] | 0 – 10 ms | From the USB host | N/A |

1. This parameter is defined by the *USB Battery Charging Specification, v1.2*.

The following figure illustrates the charger detection sequence in a simplified timing diagram based on the USB Battery Charging Specification v1.1.

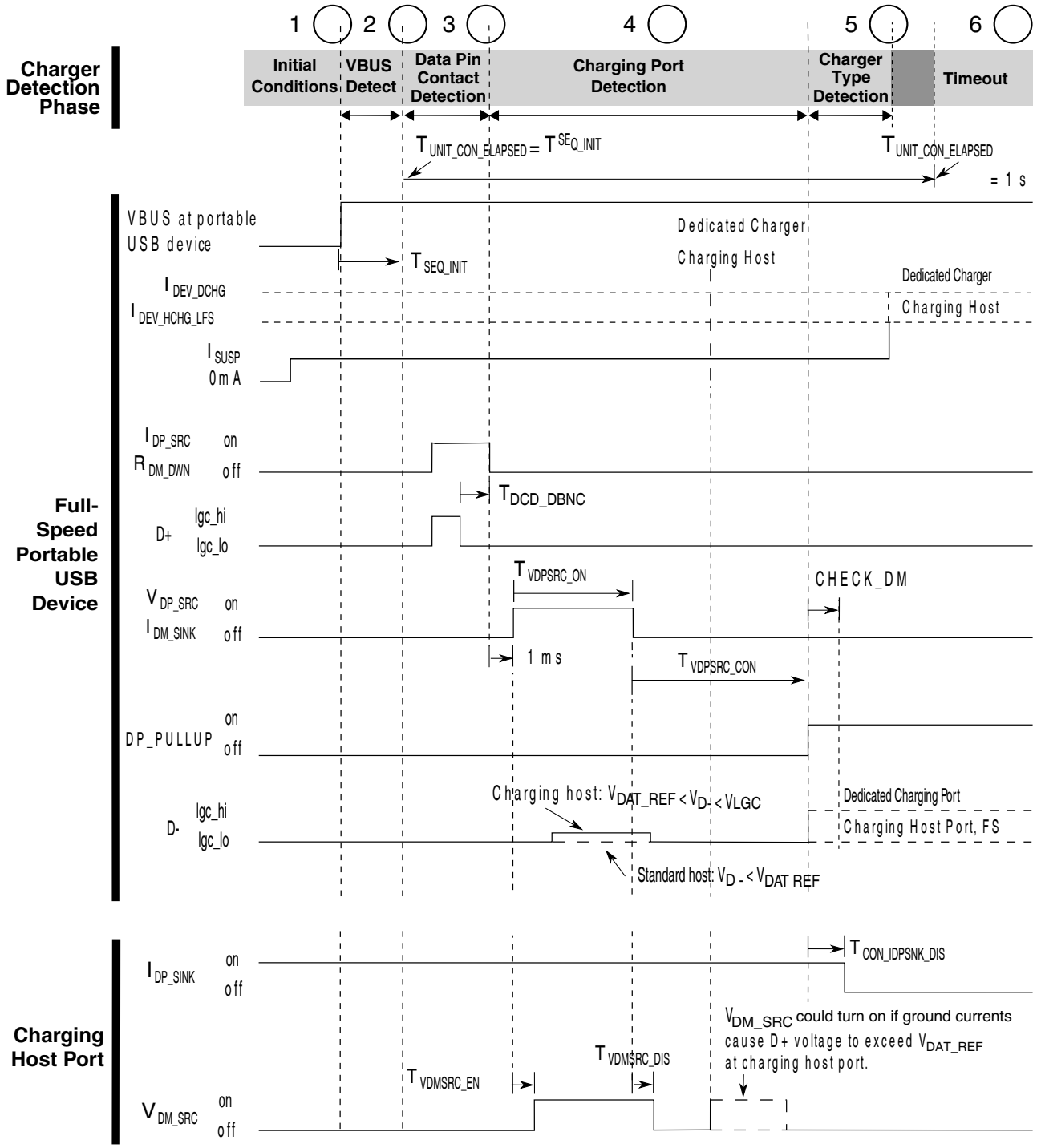



**Figure 46-11. Full speed charger detection timing for BC1.1**

Timing parameter values used in this module for BC1.1 are listed in the following table.

**Table 46-16. Timing parameters for the charger detection sequence for BC1.1**

| Parameter | USB Battery Charging Spec | Module default | Module programmable range |
|---|---|---|---|
| $T_{DCD\_DBNC}$[1] | 10 ms min (no max) | 10 ms | 0– 1023 ms |
| $T_{VDPSRC\_ON}$[1] | 40 ms min (no max) | 40 ms | 0 –1023 ms |
| $T_{VDPSRC\_CON}$[1] | 40 ms min (no max) | 40 ms | 0 –1023 ms |
| CHECK_DM | N/A | 1 ms | 0– 15 ms |
| $T_{SEQ\_INIT}$ | N/A | 16 ms | 0 –1023 ms |
| $T_{UNIT\_CON}$[1] | 1 s | N/A | N/A |
| $T_{VDMSRC\_EN}$[1] | 1– 20 ms | From the USB host | N/A |
| $T_{VDMSRC\_DIS}$[1] | 0 –20 ms | From the USB host | N/A |
| $T_{CON\_IDPSINK\_DIS}$[1] | 0– 20 ms | From the USB host | N/A |

1. This parameter is defined by the *USB Battery Charging Specification, v1.1*.

The following table provides an overview description of the charger detection sequence shown in the preceding figure.

**Table 46-17. Overview of the charger detection sequence**

| Phase | | Overview description | Full description |
|---|---|---|---|
| 1 | Initial Conditions | Initial system conditions that need to be met before the detection sequence is initiated. | Initial System Conditions |
| 2 | VBUS Detection | System software detects contact of the VBUS signal with the system interrupt pin VBUS_detect. | VBUS contact detection |
| 3 | Data Pin Contact Detection | The USBDCD module detects that the USB data pins D+ and D– have made contact with the USB port. | Data pin contact detection |
| 4 | Charging Port Detection | The USBDCD module detects if the port is a standard host or either type of charging port, that is charging host or dedicated charger. | Charging port detection |
| 5 | Charger Type Detection | The USBDCD module detects the type of charging port, if applicable. | Charger type detection |
| 6 | Sequence Timeout | The USBDCD module did not finish the detection sequence within the timeout interval. The sequence will continue until halted by software. | Charger detection sequence timeout |

### 46.5.1.1 Initial System Conditions

The USBDCD module can be used only with FS USB device applications using a rechargable battery. That is, it cannot be used with USB applications that are HS, LS, host, or OTG. In addition, before the USBDCD module's charger detection sequence can be initiated, the system must be:

- Powered-up and in run mode.

- Recently plugged into a USB port.
- Drawing not more than 2.5 mA total system current from the USB bus.

Examples of allowable precursors to this set of initial conditions include:

- A powered-down device is subsequently powered-up upon being plugged into the USB bus.
- A device in a low power mode subsequently enters run mode upon being plugged into the USB bus.

#### 46.5.1.2 VBUS contact detection

Once the device is plugged into a USB port, the VBUS_detect system interrupt is triggered. System software must do the following to initialize the module and start the charger detection sequence:

1. Restore power if the module is powered-off.
2. Set CONTROL[SR] to initiate a software reset.
3. Configure the USBDCD module by programming the CLOCK register and the timing parameters as needed.
4. Set CONTROL[IE] to enable interrupts, or clear the bit if software polling method is used.
5. Program CONTROL[BC12] based on which revision of the USB Battery Charging Specification is needed.
6. Set CONTROL[START] to start the charger detection sequence.

#### 46.5.1.3 Data pin contact detection

The module must ensure that the data pins have made contact because the detection sequence depends upon the state of the USB D+ signal. USB plugs and receptables are designed such that when the plug is inserted into the receptable, the power pins make contact before the data pins make contact. See the following figure.

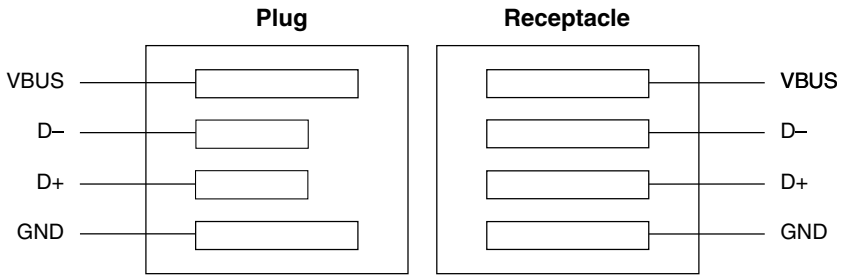



**Figure 46-12. Relative pin positions in USB plugs and receptacles**

As a result, when a portable USB device is attached to an upstream port, the portable USB device detects VBUS before the data pins have made contact. The time between power pins and data pins making contact depends on how fast the plug is inserted into the receptable. Delays of several hundred milliseconds are possible.

#### 46.5.1.3.1 Debouncing the data pin contact

When system software has initiated the charger detection sequence, as described in Initial System Conditions, the USBDCD module turns on the $I_{DP\_SRC}$ current source and enables the $R_{DM\_DWN}$ pulldown resistor. If the data pins have not made contact, the D+ line remains high. After the data pins make contact, the D+ line goes low and debouncing begins.

After the D+ line goes low, the module continuously samples the D+ line over the duration of the $T_{DCD\_DBNC}$ debounce time interval.By deafult, $T_{DCD\_DBNC}$ is 10 ms, but it can be programmed in the TIMER0[TDCD_DBNC] field. See the description of the TIMER0 Register for register information.

When it has remained low for the entire interval, the debouncing is complete. However, if the D+ line returns high during the debounce interval, the module waits until the D+ line goes low again to restart the debouncing. This cycle repeats until either of the following happens:

- The data pin contact has been successfully debounced (see Success in detecting data pin contact (phase completion)).
- A timeout occurs (see Charger detection sequence timeout).

#### 46.5.1.3.2 Success in detecting data pin contact (phase completion)

After successfully debouncing the D+ state, the module does the following:

- Updates the STATUS register to reflect phase completion (See Table 46-21 for field values.)
- Directly proceeds to the next step in the sequence: detection of a charging port (See Charging port detection.)

### 46.5.1.4 Charging port detection

After it detects that the data pins have made contact, the module waits for a fixed delay of 1 ms, and then attempts to detect whether it is plugged into a charging port. The module connects the following analog units to the USB D+ or D– lines during this phase:

- The voltage source $V_{DP\_SRC}$ connects to the D+ line
- The current sink $I_{DM\_SINK}$ connects to the D– line
- The voltage comparator connects to the USB D– line, comparing it to the voltage $V_{DAT\_REF}$.

After a time of $T_{VDPSRC\_ON}$, the module samples the D– line. The $T_{VDPSRC\_ON}$ parameter is programmable and defaults to 40 ms. After sampling the D– line, the module disconnects the voltage source, current sink, and comparator.

The next steps in the sequence depend on the voltage on the D– line as determined by the voltage comparator. See the following table.

**Table 46-18. Sampling D– in the charging port detection phase**

| If the voltage on D- is... | Then... | See... |
|---|---|---|
| Below $V_{DAT\_REF}$ | The port is a *standard host* that does not support the USB Battery Charging Specification. | Standard host port |
| Above $V_{DAT\_REF}$ but below $V_{LGC}$ | The port is a *charging port*. | Charging port |
| Above $V_{LGC}$ | This is an error condition. | Error in charging port detection |

#### 46.5.1.4.1 Standard host port

As part of the charger detection handshake with a standard USB host, the module does the following without waiting for the interval ($T_{WAIT\_AFTER\_PRD}$ or $T_{VDPSRC\_CON}$) to elapse:

- Updates the STATUS register to reflect that a standard host has been detected with SEQ_RES = 01. See Table 46-21 for field values.
- Sets CONTROL[IF].
- Generates an interrupt if enabled in CONTROL[IE].

At this point, control has been passed to system software via the interrupt. The rest of the sequence, which detects the type of charging port, is not applicable, so software should perform the following steps:

1. Read the STATUS register.
2. Set CONTROL[IACK] to acknowledge the interrupt.
3. Set CONTROL[SR] to issue a software reset to the module.
4. Disable the module.
5. Communicate the appropriate charge rate to the external battery charger IC; see Table 46-14.

#### 46.5.1.4.2 Charging port

As part of the charger detection handshake with any type of USB host, the module waits until the interval ($T_{WAIT\_AFTER\_PRD}$ or $T_{VDPSRC\_CON}$) has elapsed before it does the following:

- Enables $V_{DM\_SRC}$ (for USB Battery Charging Specifications v1.2 only).
- Updates the STATUS register to reflect that a charging port has been detected with SEQ_RES = 10. See Table 46-21 for field values.
- Sets CONTROL[IF].
- Generates an interrupt if enabled in CONTROL[IE].

At this point, control has passed to system software via the interrupt. Software should:

1. Read the STATUS register.
2. Set CONTROL[IACK] to acknowledge the interrupt.
3. Issue a command to the USB controller to pullup the USB D+ line.
4. Wait for the module to complete the final phase of the sequence. See Charger type detection.

#### 46.5.1.4.3 Error in charging port detection

For this error condition, the module does the following:

- Updates the STATUS register to reflect the error with SEQ_RES = 00. See Table 46-21 for field values.
- Sets CONTROL[IF].
- Generates an interrupt if enabled in CONTROL[IE].

Note that in this case the module does not wait for the interval ($T_{WAIT\_AFTER\_PRD}$ or $T_{VDPSRC\_CON}$) to elapse.

At this point, control has been passed to system software via the interrupt. The rest of the sequence (detecting the type of charging port) is not applicable, so software should:

1. Read the STATUS register.
2. Set CONTROL[IACK] to acknowledge the interrupt.
3. Set CONTROL[SR] to issue a software reset to the module.
4. Disable the module.

### 46.5.1.5 Charger type detection

For *USB Battery Charging Specification, v1.2*:

After the USBDCD module enables the $V_{DM\_SRC}$, the module starts the $T_{VDMSRC\_ON}$ timer counting down the time interval programmed into the TIMER2[$T_{VDMSRC\_ON}$] field.

Once the $T_{VDMSRC\_ON}$ timer has elapsed, the module samples the USB D+ line to determine the type of charger. See the following table.

**Table 46-19. Sampling D+ in the charger type detection phase (BC1.2)**

| If the voltage on D+ is... | Then... | See... |
|---|---|---|
| High (D+>$V_{DAT\_REF}$) | The port is a *dedicated charging port*.[1] | Dedicated charging port |
| Low (D+<$V_{DAT\_REF}$) | The port is a *charging host port*.[2] | Charging host port |

1. In a dedicated charger, the D+ and D– lines are shorted together through a small resistor.
2. In a charging host port, the D+ and D– lines are not shorted.

For *USB Battery Charging Specification, v1.1*:

After software enables the D+ pullup resistor, the module is notified automatically (via internal signaling) to start the CHECK_DM timer counting down the time interval programmed in the TIMER2[CHECK_DM] field.

After the CHECK_DM time has elapsed, the module samples the USB D– line to determine the type of charger. See the following table.

**Table 46-20. Sampling D– in the charger type detection phase (BC1.1)**

| If the voltage on D– is... | Then... | See... |
|---|---|---|
| High | The port is a *dedicated charging port*.[1] | Dedicated charging port |
| Low | The port is a *charging host port*.[2] | Charging host port |

1. In a dedicated charger, the D+ and D– lines are shorted together through a small resistor.
2. In a charging host port, the D+ and D– lines are not shorted.

#### 46.5.1.5.1 Dedicated charging port

For a dedicated charger, the module does the following:

- Updates the STATUS register to reflect that a dedicated charger has been detected with SEQ_RES = 11. See Table 46-21 for field values.
- Sets CONTROL[IF].
- Generates an interrupt if enabled in CONTROL[IE] bit.

At this point, control has been passed to system software via the interrupt. Software should:

1. Read the STATUS register.
2. Disable the USB controller to prevent transitions on the USB D+ or D– lines from causing spurious interrupt or wakeup events to the system.
3. Set CONTROL[IACK] to acknowledge the interrupt.
4. Set CONTROL[SR] to issue a software reset to the module.
5. Disable the module.
6. Communicate the appropriate charge rate to the external battery charger IC; see Table 46-14.

#### 46.5.1.5.2 Charging host port

For a charging host port, the module does the following:

- Updates the STATUS register to reflect that a charging host port has been detected with SEQ_RES = 10. See Table 46-21 for field values.
- Sets CONTROL[IF].
- Generates an interrupt if enabled in CONTROL[IE].

At this point, control has been passed to system software via the interrupt. Software should:

1. Read the STATUS register.
2. Set CONTROL[IACK] to acknowledge the interrupt.
3. Set CONTROL[SR] to issue a software reset to the module.
4. Disable the module.
5. Communicate the appropriate charge rate to the external battery charger IC; see Table 46-14.

### 46.5.1.6 Charger detection sequence timeout

The maximum time allowed to connect according to the *USB Battery Charging Specification* is one second. If the Unit Connection Timer reaches the one second limit and the sequence is still running as indicated by the STATUS[ACTIVE] bit, the module does the following:

- Updates the STATUS register to reflect that a timeout error has occured. See Table 46-21 for field values.
- Sets the CONTROL[IF] bit.
- Generates an interrupt if enabled in CONTROL[IE].
- The detection sequence continues until explicitly halted by software setting the CONTROL[SR] bit.
- The Unit Connection Timer continues counting. See the description of the TIMER0 Register.

At this point, control has been passed to system software via the interrupt, which has two options: ignore the interrupt and allow more time for the sequence to complete, or halt the sequence. To halt the sequence, software should:

1. Read the STATUS register.
2. Set the CONTROL[IACK] bit to acknowledge the interrupt.
3. Set the CONTROL[SR] bit to issue a software reset to the module.
4. Disable the module.

This timeout function is also useful in case software does not realize that the USB device is unplugged from USB port during the charger detection sequence. If the interrupt occurs but the $V_{BUS\_DETECT}$ input is low, software can disable and reset the module.

System software might allow the sequence to run past the timeout interrupt under these conditions:

1. The USB Battery Charging Spec is amended to allow more time. In this case, software should poll TIMER0[$T_{UNITCON}$] periodically to track elapsed time after 1s; or
2. For debug purposes.

Note that the $T_{UNITCON}$ register field will stop incrementing when it reaches its maximum value so it will not rollover to zero and start counting up again.

## 46.5.2 Interrupts and events

The USBDCD module has an interrupt to alert system software of certain events, which are listed in the following table. All events except the Phase Complete event for the Data Pin Detection phase can trigger an interrupt.

**Table 46-21. Events triggering an interrupt by sequence phase**

| Sequence phase | Event | Event description | STATUS fields[1] | Phase description |
|---|---|---|---|---|
| Data Pin Detection | Phase Complete | The module has detected data pin contact.<br>*No interrupt occurs*: CONTROL[IF] = 0. | ERR = 0<br>SEQ_STAT = 01<br>SEQ_RES = 00<br>TO = 0 | VBUS contact detection |

*Table continues on the next page...*

**Table 46-21. Events triggering an interrupt by sequence phase (continued)**

| Sequence phase | Event | Event description | STATUS fields[1] | Phase description |
|---|---|---|---|---|
| Charging Port Detection | Phase Complete | The module has completed the process of identifying if the USB port is a charging port or not.<br>**NOTE:** This only applies to USB Battery Charging Specification, v1.1. | ERR = 0<br>SEQ_STAT = 10<br>SEQ_RES = 01 or 10<br>TO = 0 | Charging port detection |
| | Error | The module cannot identify the type of port because the D– line is above the USB's VLGC threshold. | ERR = 1<br>SEQ_STAT = 10<br>SEQ_RES = 00<br>TO = 0 | Error in charging port detection |
| Charger Type Detection | Phase Complete | The module has completed the process of identifying the charger type detection.<br>**Note:** The ERR flag always reads zero because no known error conditions are checked during this phase. | ERR = 0<br>SEQ_STAT = 11<br>SEQ_RES = 11 or 10<br>TO = 0 | Charger type detection |
| Sequence Timeout | Error | The timeout interval from the time the USB device attaches to a USB port until it connects has elapsed. | ERR = 1<br>SEQ_STAT = last value[2]<br>SEQ_RES = last value[2]<br>TO = 1 | Charger detection sequence timeout. |

1. See the description of the Status register for register information.
2. The SEQ_STAT and SEQ_RES fields retain the values held at the time of the timeout error.

### 46.5.2.1 Interrupt Handling

Software can read which event caused the interrupt from the STATUS register during the interrupt service routine.

An interrupt is generated only if CONTROL[IE] is set. The CONTROL[IF] bit is always set under interrupt conditions, even if CONTROL[IE] is cleared. In this case, software can poll CONTROL[IF] to determine if an interrupt condition is pending.

Writes to CONTROL[IF] are ignored. To reset CONTROL[IF], set CONTROL[IACK] to acknowledge the interrupt. Writing to CONTROL[IACK] when CONTROL[IF] is cleared has no effect.

## 46.5.3 Resets

There are two ways to reset various register contents in this module: hardware resets and a software reset.

#### 46.5.3.1 Hardware resets

Hardware resets originate at the system or device level and propagate down to the individual module level. They include start up reset, low-voltage reset, and all other hardware reset sources.

Hardware resets cause the register contents to be restored to their default state as listed in the register descriptions.

#### 46.5.3.2 Software reset

A software reset re-initializes the module's status information, but leaves configuration information unchanged. The software reset allows software to prepare the module without needing to reprogram the same configuration each time the USB device is plugged into a USB port.

Setting CONTROL[SR] initiates a software reset. The following table shows all register fields that are reset to their default values by a software reset.

**Table 46-22. Software reset and register fields affected**

| Register | Fields affected | Fields not affected |
|---|---|---|
| CONTROL[1] | IF | IE, START |
| STATUS | All | None |
| CLOCK | None | All |
| TIMER*n* | TUNITCON | All other |

1. CONTROL[SR] and CONTROL[ IACK] are self-clearing.

A software reset also returns all internal logic, timers, and counters to their reset states. If the module is already active (STATUS[ACTIVE] = 1), a software reset stops the sequence.

**Note**

Software must always initiate a software reset before starting the sequence to ensure the module is in a known state.

## 46.6 Initialization information

This module has been designed for minimal configuration while retaining significant programmability. The CLOCK register needs to be initialized to the actual system clock frequency, unless the default value already matches the system requirements.

The other registers generally do not need to be modified, because they default to values that comply with the USB Battery Charging Specification. However, several timing parameters can be changed for a great deal of flexibility if a particular system requires it.

All module configuration must occur *before* initiating the charger detection sequence. Configuration changes made *after* setting CONTROL[START] result in undefined behavior.

## 46.7 Application information

This section provides application information.

### 46.7.1 External pullups

Any external pullups applied to the USB D+ or D- data lines must be capable of being disabled to prevent incorrect pullup values or incorrect operation of the USB subsystem.

### 46.7.2 Dead or weak battery

According to the USB Battery Charging Specification, a USB device with a dead, weak, or missing battery that is attached to a charging port can remain attached indefinitely drawing up to 1.5A, until the battery is charged to the point that the USB device can connect.

The USBDCD module is compatible with systems that do not check the strength of the battery. Therefore, this module assumes that the battery is good, so the USB device must immediately connect to the USB bus by pulling the D+ line high after the USBDCD module has determined that the device is attached to a charging port.

The module is also compatible with systems that do check the strength of the battery. In these systems, if it is known that the battery is weak or dead, software can delay connecting to the USB while charging at 1.5A. Once the battery is charged to the good battery threshold, software can then connect to the USB host by pulling the D+ line high.

### 46.7.3 Handling unplug events

If the device is unplugged from the USB bus during the charger detection sequence, the contents of the STATUS register must be ignored and the USBDCD module must get a Software Reset, as described in Software reset.

# Chapter 47
# USB Voltage Regulator

## 47.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The USB Voltage Regulator module is a LDO linear voltage regulator to provide 3.3V power from an input power supply varying from 2.7 V to 5.5 V. It consists of one 3.3 V power channel. When the input power supply is below 3.6 V, the regulator goes to pass-through mode. The following figure shows the ideal relation between the regulator output and input power supply.

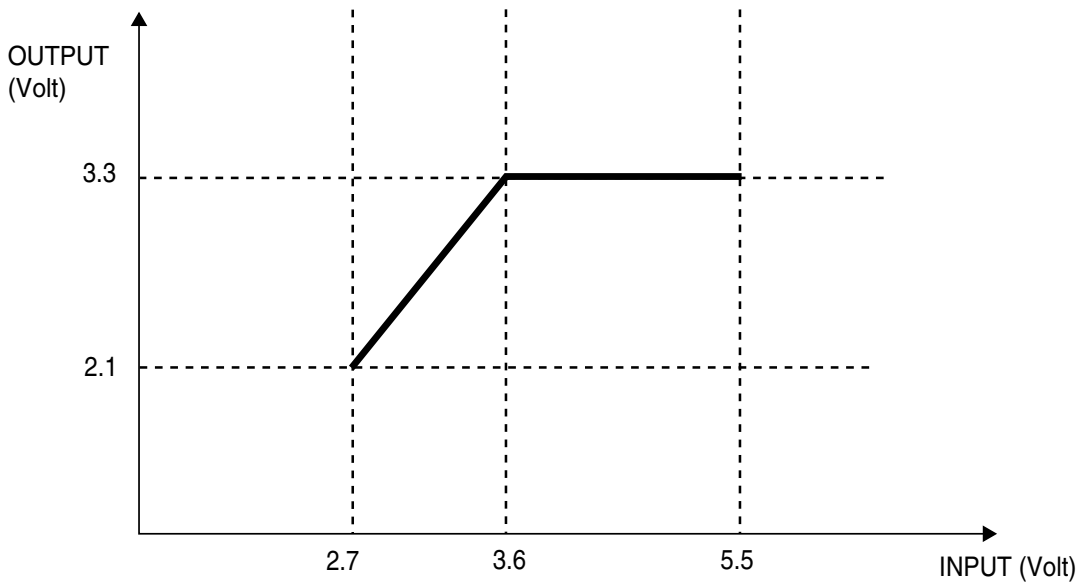



**Figure 47-1. Ideal Relation Between the Regulator Output and Input Power Supply**

## 47.1.1 Overview

A simplified block diagram for the USB Voltage Regulator module is shown below.

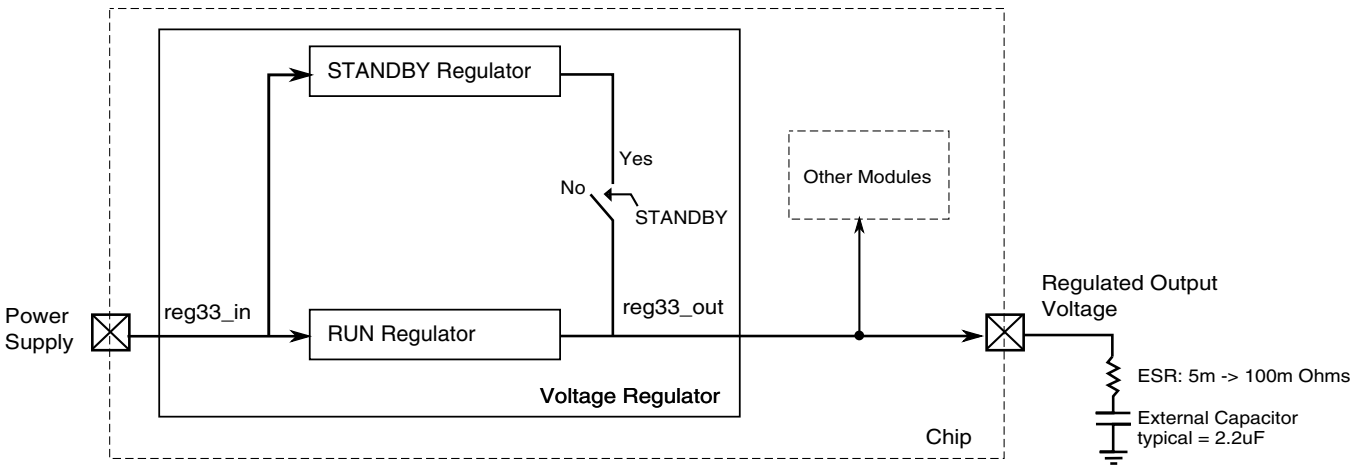



**Figure 47-2. USB Voltage Regulator Block Diagram**

This module uses 2 regulators in parallel. In run mode, the RUN regulator with the bandgap voltage reference is enabled and can provide up to 120 mA load current. In run mode, the STANDBY regulator and the low power reference are also enabled, but a switch disconnects its output from the external pin. In STANDBY mode, the RUN regulator is disabled and the STANDBY regulator output is connected to the external pin.

Internal power mode signals control whether the module is in RUN or STANDBY mode.

## 47.1.2 Features

- Low drop-out linear voltage regulator with one power channel (3.3 V).
- Low drop-out voltage: 300 mV.
- Output current: 120 mA.
- Three different power modes: RUN, STANDBY and SHUTDOWN.
- Low quiescent current in RUN mode.
  - Typical value is around 120 µA (one thousand times smaller than the maximum load current).
- Very low quiescent current in STANDBY mode.
  - Typical value is around 1 µA.
- Automatic current limiting if the load current is greater than 290 mA.

- Automatic power-up once some voltage is applied to the regulator input.
- Pass-through mode for regulator input voltages less than 3.6 V
- Small output capacitor: 2.2 µF
- Stable with aluminum, tantalum or ceramic capacitors.

### 47.1.3 Modes of Operation

The regulator has these power modes:

- RUN—The regulating loop of the RUN regulator and the STANDBY regulator are active, but the switch connecting the STANDBY regulator output to the external pin is open.
- STANDBY—The regulating loop of the RUN regulator is disabled and the standby regulator is active. The switch connecting the STANDBY regulator output to the external pin is closed.
- SHUTDOWN—The module is disabled.

The regulator is enabled by default. This means that once the power supply is provided, the module power-up sequence to RUN mode starts.

## 47.2 USB Voltage Regulator Module Signal Descriptions

The following table shows the external signals for the regulator.

**Table 47-1. USB Voltage Regulator Module Signal Descriptions**

| Signal | Description | I/O |
|---|---|---|
| reg33_in | Unregulated power supply | I |
| reg33_out | Regulator output voltage | O |

# Chapter 48
# CAN (FlexCAN)

## 48.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The FlexCAN module is a communication controller implementing the CAN protocol according to the CAN 2.0 B protocol specification. A general block diagram is shown in the following figure, which describes the main subblocks implemented in the FlexCAN module, including one associated memory for storing message buffers, Receive (Rx) Global Mask registers, Receive Individual Mask registers, Receive FIFO filters, and Receive FIFO ID filters. The functions of the submodules are described in subsequent sections.

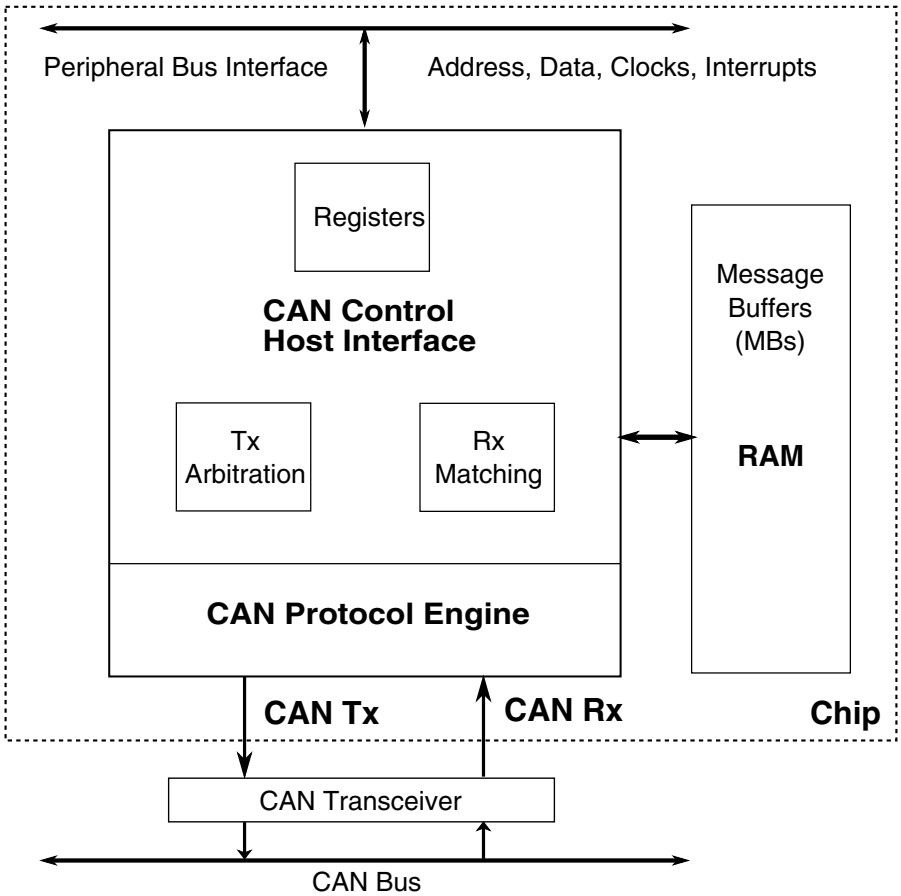



**Figure 48-1. FlexCAN block diagram**

## 48.1.1 Overview

The CAN protocol was primarily designed to be used as a vehicle serial data bus, meeting the specific requirements of this field:

- Real-time processing
- Reliable operation in the EMI environment of a vehicle
- Cost-effectiveness
- Required bandwidth

The FlexCAN module is a full implementation of the CAN protocol specification, Version 2.0 B, which supports both standard and extended message frames. The message buffers are stored in an embedded RAM dedicated to the FlexCAN module. See the chip configuration details for the actual number of message buffers configured in the MCU.

The CAN Protocol Engine (PE) submodule manages the serial communication on the CAN bus:

- Requesting RAM access for receiving and transmitting message frames

- Validating received messages
- Performing error handling

The Controller Host Interface (CHI) sub-module handles message buffer selection for reception and transmission, taking care of arbitration and ID matching algorithms.

The Bus Interface Unit (BIU) sub-module controls the access to and from the internal interface bus, in order to establish connection to the CPU and to other blocks. Clocks, address and data buses, interrupt outputs and test signals are accessed through the BIU.

## 48.1.2 FlexCAN module features

The FlexCAN module includes these distinctive legacy features:

- Full implementation of the CAN protocol specification, Version 2.0 B
  - Standard data and remote frames
  - Extended data and remote frames
  - Zero to eight bytes data length
  - Programmable bit rate up to 1 Mb/sec
  - Content-related addressing
- Flexible mailboxes of zero to eight bytes data length
- Each mailbox configurable as receive or transmit, all supporting standard and extended messages
- Individual Rx Mask registers per mailbox
- Full-featured Rx FIFO with storage capacity for up to six frames and automatic internal pointer handling
- Transmission abort capability
- Programmable clock source to the CAN Protocol Interface, either bus clock or crystal oscillator
- Unused structures space can be used as general purpose RAM space
- Listen-Only mode capability
- Programmable Loop-Back mode supporting self-test operation

- Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority
- Time stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts
- Independence from the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Low power modes, with programmable wake up on bus activity

New major features are also provided:

- Remote request frames may be handled automatically or by software
- CAN bit time settings and configuration bits can only be written in Freeze mode
- Tx mailbox status (Lowest priority buffer or empty buffer)
- Identifier Acceptance Filter Hit Indicator (IDHIT) register for received frames
- SYNCH bit available in Error in Status 1 register to inform that the module is synchronous with CAN bus
- CRC status for transmitted message
- Rx FIFO Global Mask register
- Selectable priority between mailboxes and Rx FIFO during matching process
- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 128 extended, 256 standard, or 512 partial (8 bit) IDs, with up to 32 individual masking capability
- 100% backward compatibility with previous FlexCAN version

### 48.1.3 Modes of operation

The FlexCAN module has these functional modes:

- Normal mode (User or Supervisor):

In Normal mode, the module operates receiving and/or transmitting message frames, errors are handled normally, and all CAN Protocol functions are enabled. User and Supervisor Modes differ in the access to some restricted control registers.

- Freeze mode:

  Freeze mode is enabled when the FRZ bit in MCR is asserted. If enabled, Freeze mode is entered when MCR[HALT] is set or when Debug mode is requested at MCU level and MCR[FRZ_ACK ] is asserted by the FlexCAN. In this mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost. See Freeze mode for more information.

- Listen-Only mode:

  The module enters this mode when the LOM field in the Control 1 Register is asserted. In this mode, transmission is disabled, all error counters are frozen, and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.

- Loop-Back mode:

  The module enters this mode when the LPB field in the Control 1 Register is asserted. In this mode, FlexCAN performs an internal loop back that can be used for self-test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic '1'). FlexCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

For low-power operation, the FlexCAN module has:

- Module Disable mode:

  This low-power mode is entered when the MDIS bit in the MCR Register is asserted by the CPU and the LPM_ACK is asserted by the FlexCAN. When disabled, the module requests to disable the clocks to the CAN Protocol Engine and Controller Host Interface submodules. Exit from this mode is done by negating the MDIS bit in the MCR register. See Module Disable mode for more information.

- Stop mode:

This low power mode is entered when Stop mode is requested at MCU level and the LPM_ACK bit in the MCR Register is asserted by the FlexCAN. When in Stop Mode, the module puts itself in an inactive state and then informs the CPU that the clocks can be shut down globally. Exit from this mode happens when the Stop mode request is removed, or when activity is detected on the CAN bus and the Self Wake Up mechanism is enabled. See Stop mode for more information.

## 48.2 FlexCAN signal descriptions

The FlexCAN module has two I/O signals connected to the external MCU pins. These signals are summarized in the following table and described in more detail in the next subsections.

**Table 48-1. FlexCAN signal descriptions**

| Signal | Description | I/O |
|---|---|---|
| CAN Rx | CAN Receive Pin | Input |
| CAN Tx | CAN Transmit Pin | Output |

### 48.2.1 CAN Rx

This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

### 48.2.2 CAN Tx

This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

## 48.3 Memory map/register definition

This section describes the registers and data structures in the FlexCAN module. The base address of the module depends on the particular memory map of the MCU.

### 48.3.1 FlexCAN memory mapping

The complete memory map for a FlexCAN module is shown in the following table.

The address space occupied by FlexCAN has 128 bytes for registers starting at the module base address, followed by embedded RAM starting at address 0x0080.

Each individual register is identified by its complete name and the corresponding mnemonic. The access type can be Supervisor (S) or Unrestricted (U). Most of the registers can be configured to have either Supervisor or Unrestricted access by programming the SUPV field in the MCR register. These registers are identified as S/U in the Access column of Table 48-2.

**Table 48-2. Register access and reset information**

| Register | Access type | Affected by hard reset | Affected by soft reset |
|---|---|---|---|
| Module Configuration Register (MCR) | S | Yes | Yes |
| Control 1 register (CTRL1) | S/U | Yes | No |
| Free Running Timer register (TIMER) | S/U | Yes | Yes |
| Rx Mailboxes Global Mask register (RXMGMASK) | S/U | No | No |
| Rx Buffer 14 Mask register (RX14MASK) | S/U | No | No |
| Rx Buffer 15 Mask register (RX15MASK) | S/U | No | No |
| Error Counter Register (ECR) | S/U | Yes | Yes |
| Error and Status 1 Register (ESR1) | S/U | Yes | Yes |
| Interrupt Masks 2 register (IMASK2) | S/U | Yes | Yes |
| Interrupt Masks 1 register (IMASK1) | S/U | Yes | Yes |
| Interrupt Flags 2 register (IFLAG2) | S/U | Yes | Yes |
| Interrupt Flags 1 register (IFLAG1) | S/U | Yes | Yes |
| Control 2 Register (CTRL2) | S/U | Yes | No |
| Error and Status 2 Register (ESR2) | S/U | Yes | Yes |
| CRC Register (CRCR) | S/U | Yes | Yes |
| Rx FIFO Global Mask register (RXFGMASK) | S/U | No | No |
| Rx FIFO Information Register (RXFIR) | S/U | No | No |
| Message buffers | S/U | No | No |
| Rx Individual Mask Registers | S/U | No | No |

The FlexCAN module can store CAN messages for transmission and reception using mailboxes and Rx FIFO structures.

This module's memory map includes sixteen 128-bit message buffers (MBs) that occupy the range from offset 0x80 to 0x17F.

## CAN memory map

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4002_4000 | Module Configuration Register (CAN0_MCR) | 32 | R/W | D890_000Fh | 48.3.2/1206 |
| 4002_4004 | Control 1 register (CAN0_CTRL1) | 32 | R/W | 0000_0000h | 48.3.3/1211 |
| 4002_4008 | Free Running Timer (CAN0_TIMER) | 32 | R/W | 0000_0000h | 48.3.4/1214 |
| 4002_4010 | Rx Mailboxes Global Mask Register (CAN0_RXMGMASK) | 32 | R/W | FFFF_FFFFh | 48.3.5/1215 |
| 4002_4014 | Rx 14 Mask register (CAN0_RX14MASK) | 32 | R/W | FFFF_FFFFh | 48.3.6/1216 |
| 4002_4018 | Rx 15 Mask register (CAN0_RX15MASK) | 32 | R/W | FFFF_FFFFh | 48.3.7/1217 |
| 4002_401C | Error Counter (CAN0_ECR) | 32 | R/W | 0000_0000h | 48.3.8/1217 |
| 4002_4020 | Error and Status 1 register (CAN0_ESR1) | 32 | R/W | 0000_0000h | 48.3.9/1219 |
| 4002_4028 | Interrupt Masks 1 register (CAN0_IMASK1) | 32 | R/W | 0000_0000h | 48.3.10/ 1223 |
| 4002_4030 | Interrupt Flags 1 register (CAN0_IFLAG1) | 32 | R/W | 0000_0000h | 48.3.11/ 1223 |
| 4002_4034 | Control 2 register (CAN0_CTRL2) | 32 | R/W | 00B0_0000h | 48.3.12/ 1226 |
| 4002_4038 | Error and Status 2 register (CAN0_ESR2) | 32 | R/W | 0000_0000h | 48.3.13/ 1229 |
| 4002_4044 | CRC Register (CAN0_CRCR) | 32 | R | 0000_0000h | 48.3.14/ 1231 |
| 4002_4048 | Rx FIFO Global Mask register (CAN0_RXFGMASK) | 32 | R/W | FFFF_FFFFh | 48.3.15/ 1231 |
| 4002_404C | Rx FIFO Information Register (CAN0_RXFIR) | 32 | R | Undefined | 48.3.16/ 1232 |
| 4002_4880 | Rx Individual Mask Registers (CAN0_RXIMR0) | 32 | R/W | Undefined | 48.3.17/ 1233 |
| 4002_4884 | Rx Individual Mask Registers (CAN0_RXIMR1) | 32 | R/W | Undefined | 48.3.17/ 1233 |
| 4002_4888 | Rx Individual Mask Registers (CAN0_RXIMR2) | 32 | R/W | Undefined | 48.3.17/ 1233 |
| 4002_488C | Rx Individual Mask Registers (CAN0_RXIMR3) | 32 | R/W | Undefined | 48.3.17/ 1233 |
| 4002_4890 | Rx Individual Mask Registers (CAN0_RXIMR4) | 32 | R/W | Undefined | 48.3.17/ 1233 |
| 4002_4894 | Rx Individual Mask Registers (CAN0_RXIMR5) | 32 | R/W | Undefined | 48.3.17/ 1233 |
| 4002_4898 | Rx Individual Mask Registers (CAN0_RXIMR6) | 32 | R/W | Undefined | 48.3.17/ 1233 |
| 4002_489C | Rx Individual Mask Registers (CAN0_RXIMR7) | 32 | R/W | Undefined | 48.3.17/ 1233 |
| 4002_48A0 | Rx Individual Mask Registers (CAN0_RXIMR8) | 32 | R/W | Undefined | 48.3.17/ 1233 |
| 4002_48A4 | Rx Individual Mask Registers (CAN0_RXIMR9) | 32 | R/W | Undefined | 48.3.17/ 1233 |

*Table continues on the next page...*

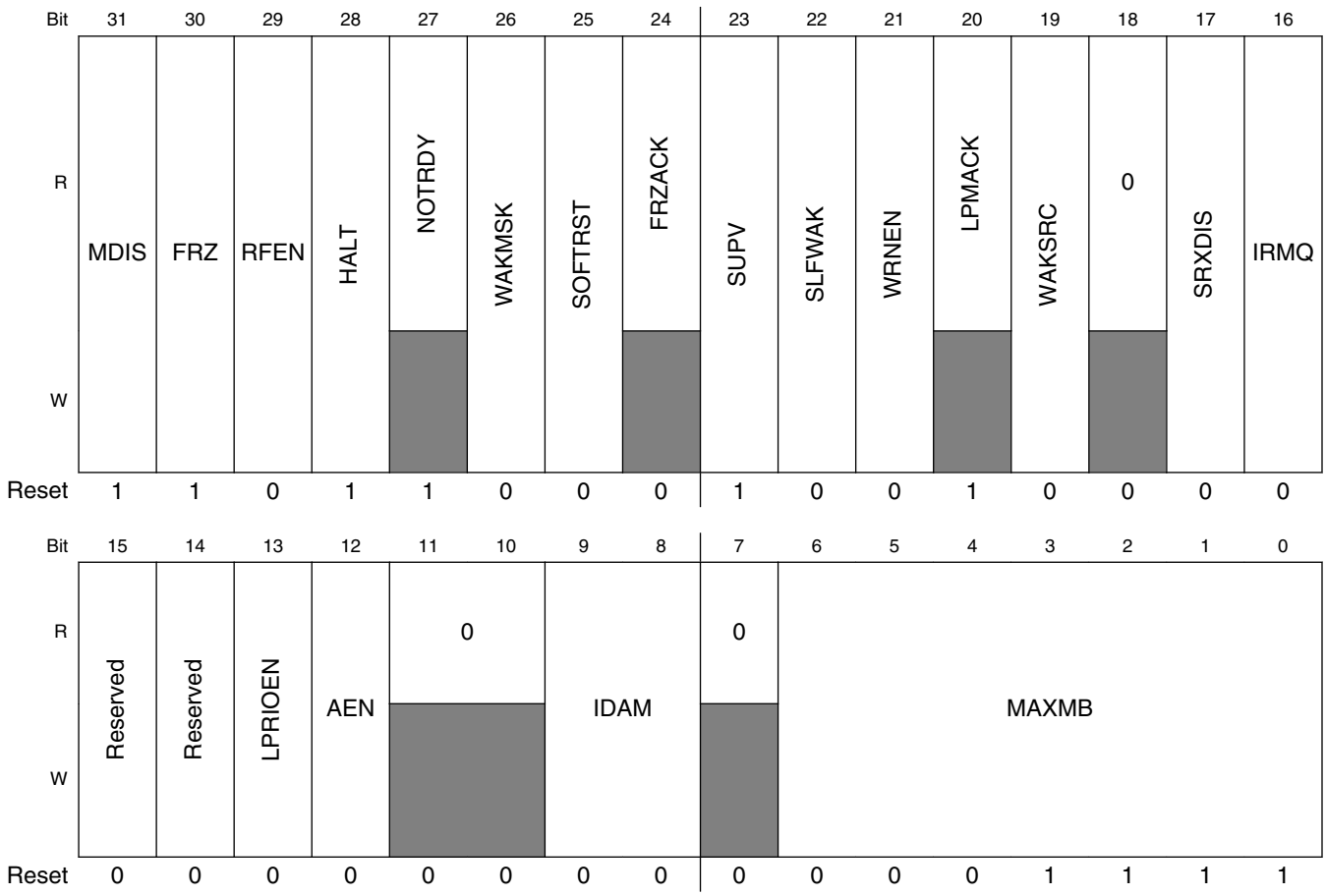
## CAN memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4002_48A8 | Rx Individual Mask Registers (CAN0_RXIMR10) | 32 | R/W | Undefined | 48.3.17/ 1233 |
| 4002_48AC | Rx Individual Mask Registers (CAN0_RXIMR11) | 32 | R/W | Undefined | 48.3.17/ 1233 |
| 4002_48B0 | Rx Individual Mask Registers (CAN0_RXIMR12) | 32 | R/W | Undefined | 48.3.17/ 1233 |
| 4002_48B4 | Rx Individual Mask Registers (CAN0_RXIMR13) | 32 | R/W | Undefined | 48.3.17/ 1233 |
| 4002_48B8 | Rx Individual Mask Registers (CAN0_RXIMR14) | 32 | R/W | Undefined | 48.3.17/ 1233 |
| 4002_48BC | Rx Individual Mask Registers (CAN0_RXIMR15) | 32 | R/W | Undefined | 48.3.17/ 1233 |

## 48.3.2 Module Configuration Register (CANx_MCR)

This register defines global system configurations, such as the module operation modes and the maximum message buffer configuration.

Address: 4002_4000h base + 0h offset = 4002_4000h




**CANx_MCR field descriptions**

| Field | Description |
|---|---|
| 31<br>MDIS | Module Disable<br><br>This bit controls whether FlexCAN is enabled or not. When disabled, FlexCAN disables the clocks to the CAN Protocol Engine and Controller Host Interface sub-modules. This is the only bit within this register not affected by soft reset.<br><br>0 Enable the FlexCAN module.<br>1 Disable the FlexCAN module. |
| 30<br>FRZ | Freeze Enable<br><br>The FRZ bit specifies the FlexCAN behavior when the HALT bit in the MCR Register is set or when Debug mode is requested at MCU level . When FRZ is asserted, FlexCAN is enabled to enter Freeze mode. Negation of this bit field causes FlexCAN to exit from Freeze mode. |

*Table continues on the next page...*

## CANx_MCR field descriptions (continued)

| Field | Description |
|---|---|
| | 0 Not enabled to enter Freeze mode.<br>1 Enabled to enter Freeze mode. |
| 29<br>RFEN | Rx FIFO Enable<br><br>This bit controls whether the Rx FIFO feature is enabled or not. When RFEN is set, MBs 0 to 5 cannot be used for normal reception and transmission because the corresponding memory region (0x80-0xDC) is used by the FIFO engine as well as additional MBs (up to 32, depending on CTRL2[RFFN] setting) which are used as Rx FIFO ID Filter Table elements. RFEN also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in the table "Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate" (in section "Arbitration and Matching Timing"). This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br><br>0 Rx FIFO not enabled.<br>1 Rx FIFO enabled. |
| 28<br>HALT | Halt FlexCAN<br><br>Assertion of this bit puts the FlexCAN module into Freeze mode. The CPU should clear it after initializing the Message Buffers and Control Register. No reception or transmission is performed by FlexCAN before this bit is cleared. Freeze mode cannot be entered while FlexCAN is in a low power mode.<br><br>0 No Freeze mode request.<br>1 Enters Freeze mode if the FRZ bit is asserted. |
| 27<br>NOTRDY | FlexCAN Not Ready<br><br>This read-only bit indicates that FlexCAN is either in Disable mode , Stop mode or Freeze mode. It is negated once FlexCAN has exited these modes.<br><br>0 FlexCAN module is either in Normal mode, Listen-Only mode or Loop-Back mode.<br>1 FlexCAN module is either in Disable mode , Stop mode or Freeze mode. |
| 26<br>WAKMSK | Wake Up Interrupt Mask<br><br>This bit enables the Wake Up Interrupt generation under Self Wake Up mechanism.<br><br>0 Wake Up Interrupt is disabled.<br>1 Wake Up Interrupt is enabled. |
| 25<br>SOFTRST | Soft Reset<br><br>When this bit is asserted, FlexCAN resets its internal state machines and some of the memory mapped registers. The following registers are reset: MCR (except the MDIS bit), TIMER , ECR, ESR1, ESR2, IMASK1, IMASK2, IFLAG1, IFLAG2 and CRCR. Configuration registers that control the interface to the CAN bus are not affected by soft reset. The following registers are unaffected: CTRL1, CTRL2, all RXIMR registers, RXMGMASK, RX14MASK, RX15MASK, RXFGMASK, RXFIR, all Message Buffers .<br><br>The SOFTRST bit can be asserted directly by the CPU when it writes to the MCR Register, but it is also asserted when global soft reset is requested at MCU level. Because soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, it may take some time to fully propagate its effect. The SOFTRST bit remains asserted while reset is pending, and is automatically negated when reset completes. Therefore, software can poll this bit to know when the soft reset has completed.<br><br>Soft reset cannot be applied while clocks are shut down in a low power mode. The module should be first removed from low power mode, and then soft reset can be applied.<br><br>0 No reset request.<br>1 Resets the registers affected by soft reset. |

*Table continues on the next page...*

## CANx_MCR field descriptions (continued)

| Field | Description |
|---|---|
| 24<br>FRZACK | Freeze Mode Acknowledge<br><br>This read-only bit indicates that FlexCAN is in Freeze mode and its prescaler is stopped. The Freeze mode request cannot be granted until current transmission or reception processes have finished. Therefore the software can poll the FRZACK bit to know when FlexCAN has actually entered Freeze mode. If Freeze Mode request is negated, then this bit is negated after the FlexCAN prescaler is running again. If Freeze mode is requested while FlexCAN is in a low power mode, then the FRZACK bit will be set only when the low-power mode is exited. See Section "Freeze Mode".<br><br>**NOTE:** FRZACK will be asserted within 178 CAN bits from the freeze mode request by the CPU, and negated within 2 CAN bits after the freeze mode request removal (see Section "Protocol Timing").<br><br>0 FlexCAN not in Freeze mode, prescaler running.<br>1 FlexCAN in Freeze mode, prescaler stopped. |
| 23<br>SUPV | Supervisor Mode<br><br>This bit configures the FlexCAN to be either in Supervisor or User mode. The registers affected by this bit are marked as S/U in the Access Type column of the module memory map. Reset value of this bit is 1, so the affected registers start with Supervisor access allowance only . This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br><br>0 FlexCAN is in User mode. Affected registers allow both Supervisor and Unrestricted accesses .<br>1 FlexCAN is in Supervisor mode. Affected registers allow only Supervisor access. Unrestricted access behaves as though the access was done to an unimplemented register location . |
| 22<br>SLFWAK | Self Wake Up<br><br>This bit enables the Self Wake Up feature when FlexCAN is in a low-power mode other than Disable mode. When this feature is enabled, the FlexCAN module monitors the bus for wake up event, that is, a recessive-to-dominant transition.<br><br>If a wake up event is detected during Stop mode, then FlexCAN generates, if enabled to do so, a Wake Up interrupt to the CPU so that it can exit Stop mode globally and FlexCAN can request to resume the clocks.<br><br>When FlexCAN is in a low-power mode other than Disable mode, this bit cannot be written as it is blocked by hardware.<br><br>0 FlexCAN Self Wake Up feature is disabled.<br>1 FlexCAN Self Wake Up feature is enabled. |
| 21<br>WRNEN | Warning Interrupt Enable<br><br>When asserted, this bit enables the generation of the TWRNINT and RWRNINT flags in the Error and Status Register. If WRNEN is negated, the TWRNINT and RWRNINT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br><br>0 TWRNINT and RWRNINT bits are zero, independent of the values in the error counters.<br>1 TWRNINT and RWRNINT bits are set when the respective error counter transitions from less than 96 to greater than or equal to 96. |
| 20<br>LPMACK | Low-Power Mode Acknowledge<br><br>This read-only bit indicates that FlexCAN is in a low-power mode (Disable mode , Stop mode ). A low-power mode cannot be entered until all current transmission or reception processes have finished, so the CPU can poll the LPMACK bit to know when FlexCAN has actually entered low power mode. |

*Table continues on the next page...*

## CANx_MCR field descriptions (continued)

| Field | Description |
|---|---|
| | **NOTE:** LPMACK will be asserted within 180 CAN bits from the low-power mode request by the CPU, and negated within 2 CAN bits after the low-power mode request removal (see Section "Protocol Timing").<br><br>0 FlexCAN is not in a low-power mode.<br>1 FlexCAN is in a low-power mode. |
| 19<br>WAKSRC | Wake Up Source<br><br>This bit defines whether the integrated low-pass filter is applied to protect the Rx CAN input from spurious wake up. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br><br>0 FlexCAN uses the unfiltered Rx input to detect recessive to dominant edges on the CAN bus.<br>1 FlexCAN uses the filtered Rx input to detect recessive to dominant edges on the CAN bus. |
| 18<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 17<br>SRXDIS | Self Reception Disable<br><br>This bit defines whether FlexCAN is allowed to receive frames transmitted by itself. If this bit is asserted, frames transmitted by the module will not be stored in any MB, regardless if the MB is programmed with an ID that matches the transmitted frame, and no interrupt flag or interrupt signal will be generated due to the frame reception. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br><br>0 Self reception enabled.<br>1 Self reception disabled. |
| 16<br>IRMQ | Individual Rx Masking And Queue Enable<br><br>This bit indicates whether Rx matching process will be based either on individual masking and queue or on masking scheme with RXMGMASK, RX14MASK and RX15MASK, RXFGMASK. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br><br>0 Individual Rx masking and queue feature are disabled. For backward compatibility with legacy applications, the reading of C/S word locks the MB even if it is EMPTY.<br>1 Individual Rx masking and queue feature are enabled. |
| 15<br>Reserved | This field is reserved. |
| 14<br>Reserved | This field is reserved. |
| 13<br>LPRIOEN | Local Priority Enable<br><br>This bit is provided for backwards compatibility with legacy applications. It controls whether the local priority feature is enabled or not. It is used to expand the ID used during the arbitration process. With this expanded ID concept, the arbitration process is done based on the full 32-bit word, but the actual transmitted ID still has 11-bit for standard frames and 29-bit for extended frames. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br><br>0 Local Priority disabled.<br>1 Local Priority enabled. |
| 12<br>AEN | Abort Enable<br><br>This bit is supplied for backwards compatibility with legacy applications. When asserted, it enables the Tx abort mechanism. This mechanism guarantees a safe procedure for aborting a pending transmission, so |

*Table continues on the next page...*

## CANx_MCR field descriptions (continued)

| Field | Description |
|---|---|
| | that no frame is sent in the CAN bus without notification. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br><br>**NOTE:** When MCR[AEN] is asserted, only the abort mechanism (see Section "Transmission Abort Mechanism") must be used for updating Mailboxes configured for transmission.<br><br>**CAUTION:** Writing the Abort code into Rx Mailboxes can cause unpredictable results when the MCR[AEN] is asserted.<br><br>0 Abort disabled.<br>1 Abort enabled. |
| 11–10<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 9–8<br>IDAM | ID Acceptance Mode<br><br>This 2-bit field identifies the format of the Rx FIFO ID Filter Table elements. Note that all elements of the table are configured at the same time by this field (they are all the same format). See Section "Rx FIFO Structure". This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br><br>00 Format A: One full ID (standard and extended) per ID Filter Table element.<br>01 Format B: Two full standard IDs or two partial 14-bit (standard and extended) IDs per ID Filter Table element.<br>10 Format C: Four partial 8-bit Standard IDs per ID Filter Table element.<br>11 Format D: All frames rejected. |
| 7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6–0<br>MAXMB | Number Of The Last Message Buffer<br><br>This 7-bit field defines the number of the last Message Buffers that will take part in the matching and arbitration processes. The reset value (0x0F) is equivalent to a 16 MB configuration. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br><br>Number of the last MB = MAXMB<br><br>**NOTE:** MAXMB must be programmed with a value smaller than the parameter NUMBER_OF_MB, otherwise the number of the last effective Message Buffer will be: (NUMBER_OF_MB - 1)<br><br>Additionally, the value of MAXMB must encompass the FIFO size defined by CTRL2[RFFN]. MAXMB also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table "Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate" (in Section "Arbitration and Matching Timing"). |

## 48.3.3 Control 1 register (CANx_CTRL1)

This register is defined for specific FlexCAN control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, Loop Back mode, Listen-Only mode, Bus Off recovery behavior and interrupt enabling (Bus-Off, Error, Warning). It also determines the Division Factor for the clock prescaler.

Address: 4002_4000h base + 4h offset = 4002_4004h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PRESDIV | | | | | | | | RJW | | PSEG1 | | | PSEG2 | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BOFFMSK | ERRMSK | CLKSRC | LPB | TWRNMSK | RWRNMSK | 0 | | SMP | BOFFREC | TSYN | LBUF | LOM | PROPSEG | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CANx_CTRL1 field descriptions**

| Field | Description |
|---|---|
| 31–24<br>PRESDIV | Prescaler Division Factor<br><br>This 8-bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency. The Maximum value of this field is 0xFF, that gives a minimum Sclock frequency equal to the PE clock frequency divided by 256. See Section "Protocol Timing". This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br><br>Sclock frequency = PE clock frequency / (PRESDIV + 1) |
| 23–22<br>RJW | Resync Jump Width<br><br>This 2-bit field defines the maximum number of time quanta that a bit time can be changed by one re-synchronization. One time quantum is equal to the Sclock period. The valid programmable values are 0–3. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br><br>Resync Jump Width = RJW + 1. |
| 21–19<br>PSEG1 | Phase Segment 1<br><br>This 3-bit field defines the length of Phase Buffer Segment 1 in the bit time. The valid programmable values are 0–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br><br>Phase Buffer Segment 1 = (PSEG1 + 1) × Time-Quanta. |
| 18–16<br>PSEG2 | Phase Segment 2 |

*Table continues on the next page...*

## CANx_CTRL1 field descriptions (continued)

| Field | Description |
|---|---|
| | This 3-bit field defines the length of Phase Buffer Segment 2 in the bit time. The valid programmable values are 1–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br>Phase Buffer Segment 2 = (PSEG2 + 1) × Time-Quanta. |
| 15<br>BOFFMSK | Bus Off Mask<br>This bit provides a mask for the Bus Off Interrupt.<br>0 Bus Off interrupt disabled.<br>1 Bus Off interrupt enabled. |
| 14<br>ERRMSK | Error Mask<br>This bit provides a mask for the Error Interrupt.<br>0 Error interrupt disabled.<br>1 Error interrupt enabled. |
| 13<br>CLKSRC | CAN Engine Clock Source<br>This bit selects the clock source to the CAN Protocol Engine (PE) to be either the peripheral clock (driven by the PLL) or the crystal oscillator clock. The selected clock is the one fed to the prescaler to generate the Serial Clock (Sclock). In order to guarantee reliable operation, this bit can be written only in Disable mode because it is blocked by hardware in other modes. See Section "Protocol Timing".<br>0 The CAN engine clock source is the oscillator clock. Under this condition, the oscillator clock frequency must be lower than the bus clock.<br>1 The CAN engine clock source is the peripheral clock. |
| 12<br>LPB | Loop Back Mode<br>This bit configures FlexCAN to operate in Loop-Back mode. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic 1). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br>**NOTE:** In this mode, the MCR[SRXDIS] cannot be asserted because this will impede the self reception of a transmitted message.<br>0 Loop Back disabled.<br>1 Loop Back enabled. |
| 11<br>TWRNMSK | Tx Warning Interrupt Mask<br>This bit provides a mask for the Tx Warning Interrupt associated with the TWRNINT flag in the Error and Status Register. This bit is read as zero when MCR[WRNEN] bit is negated. This bit can be written only if MCR[WRNEN] bit is asserted.<br>0 Tx Warning Interrupt disabled.<br>1 Tx Warning Interrupt enabled. |
| 10<br>RWRNMSK | Rx Warning Interrupt Mask |

*Table continues on the next page...*

## CANx_CTRL1 field descriptions (continued)

| Field | Description |
|---|---|
| | This bit provides a mask for the Rx Warning Interrupt associated with the RWRNINT flag in the Error and Status Register. This bit is read as zero when MCR[WRNEN] bit is negated. This bit can be written only if MCR[WRNEN] bit is asserted.<br><br>0 Rx Warning Interrupt disabled.<br>1 Rx Warning Interrupt enabled. |
| 9–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>SMP | CAN Bit Sampling<br><br>This bit defines the sampling mode of CAN bits at the Rx input. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br><br>0 Just one sample is used to determine the bit value.<br>1 Three samples are used to determine the value of the received bit: the regular one (sample point) and 2 preceding samples; a majority rule is used. |
| 6<br>BOFFREC | Bus Off Recovery<br><br>This bit defines how FlexCAN recovers from Bus Off state. If this bit is negated, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B. If the bit is asserted, automatic recovering from Bus Off is disabled and the module remains in Bus Off state until the bit is negated by the user. If the negation occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then Bus Off recovery happens as if the BOFFREC bit had never been asserted. If the negation occurs after 128 sequences of 11 recessive bits occurred, then FlexCAN will re-synchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, the BOFFREC bit can be re-asserted again during Bus Off, but it will be effective only the next time the module enters Bus Off. If BOFFREC was negated when the module entered Bus Off, asserting it during Bus Off will not be effective for the current Bus Off recovery.<br><br>0 Automatic recovering from Bus Off state enabled, according to CAN Spec 2.0 part B.<br>1 Automatic recovering from Bus Off state disabled. |
| 5<br>TSYN | Timer Sync<br><br>This bit enables a mechanism that resets the free-running timer each time a message is received in Message Buffer 0. This feature provides means to synchronize multiple FlexCAN stations with a special “SYNC” message, that is, global network time. If the RFEN bit in MCR is set (Rx FIFO enabled), the first available Mailbox, according to CTRL2[RFFN] setting, is used for timer synchronization instead of MB0. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br><br>0 Timer Sync feature disabled<br>1 Timer Sync feature enabled |
| 4<br>LBUF | Lowest Buffer Transmitted First<br><br>This bit defines the ordering mechanism for Message Buffer transmission. When asserted, the LPRIOEN bit does not affect the priority arbitration. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br><br>0 Buffer with highest priority is transmitted first.<br>1 Lowest number buffer is transmitted first. |
| 3<br>LOM | Listen-Only Mode<br><br>This bit configures FlexCAN to operate in Listen-Only mode. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN Error Passive mode. Only messages |

*Table continues on the next page...*

CANx_CTRL1 field descriptions (continued)

| Field | Description |
|---|---|
| | acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error without changing the REC, as if it was trying to acknowledge the message.<br>Listen-Only mode acknowledgement can be obtained by the state of ESR1[FLTCONF] field which is Passive Error when Listen-Only mode is entered. There can be some delay between the Listen-Only mode request and acknowledge.<br>This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br>0 Listen-Only mode is deactivated.<br>1 FlexCAN module operates in Listen-Only mode. |
| 2–0<br>PROPSEG | Propagation Segment<br>This 3-bit field defines the length of the Propagation Segment in the bit time. The valid programmable values are 0–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br>Propagation Segment Time = (PROPSEG + 1) × Time-Quanta.<br>Time-Quantum = one Sclock period. |

## 48.3.4 Free Running Timer (CANx_TIMER)

This register represents a 16-bit free running counter that can be read and written by the CPU. The timer starts from 0x0 after Reset, counts linearly to 0xFFFF, and wraps around.

The timer is clocked by the FlexCAN bit-clock, which defines the baud rate on the CAN bus. During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate. The timer is not incremented during Disable , Stop, and Freeze modes.

The timer value is captured when the second bit of the identifier field of any frame is on the CAN bus. This captured value is written into the Time Stamp entry in a message buffer after a successful reception or transmission of a message.

If bit CTRL1[TSYN] is asserted, the Timer is reset whenever a message is received in the first available Mailbox, according to CTRL2[RFFN] setting.

The CPU can write to this register anytime. However, if the write occurs at the same time that the Timer is being reset by a reception in the first Mailbox, then the write value is discarded.

Reading this register affects the Mailbox Unlocking procedure; see Section "Mailbox Lock Mechanism".

Address: 4002_4000h base + 8h offset = 4002_4008h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | TIMER | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CANx_TIMER field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–0<br>TIMER | Timer Value<br><br>Contains the free-running counter value. |

## 48.3.5 Rx Mailboxes Global Mask Register (CANx_RXMGMASK)

This register is located in RAM.

RXMGMASK is provided for legacy application support.

- When the MCR[IRMQ] bit is negated, RXMGMASK is always in effect.
- When the MCR[IRMQ] bit is asserted, RXMGMASK has no effect.

RXMGMASK is used to mask the filter fields of all Rx MBs, excluding MBs 14-15, which have individual mask registers.

This register can only be written in Freeze mode as it is blocked by hardware in other modes.

Address: 4002_4000h base + 10h offset = 4002_4010h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MG[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**CANx_RXMGMASK field descriptions**

| Field | Description |
|---|---|
| 31–0<br>MG[31:0] | Rx Mailboxes Global Mask Bits<br><br>These bits mask the Mailbox filter bits. Note that the alignment with the ID word of the Mailbox is not perfect as the two most significant MG bits affect the fields RTR and IDE, which are located in the Control and Status word of the Mailbox. The following table shows in detail which MG bits mask each Mailbox filter field. |

**CANx_RXMGMASK field descriptions (continued)**

| Field | Description |
|---|---|

| SMB[RTR] [1] | CTRL2[RRS] | CTRL2[EACEN] | Mailbox filter fields | | | |
|---|---|---|---|---|---|---|
| | | | MB[RTR] | MB[IDE] | MB[ID] | Reserved |
| 0 | - | 0 | note [2] | note [3] | MG[28:0] | MG[31:29] |
| 0 | - | 1 | MG[31] | MG[30] | MG[28:0] | MG[29] |
| 1 | 0 | - | - | - | - | MG[31:0] |
| 1 | 1 | 0 | - | - | MG[28:0] | MG[31:29] |
| 1 | 1 | 1 | MG[31] | MG[30] | MG[28:0] | MG[29] |

1. RTR bit of the Incoming Frame. It is saved into an auxiliary MB called Rx Serial Message Buffer (Rx SMB).
2. If the CTRL2[EACEN] bit is negated, the RTR bit of Mailbox is never compared with the RTR bit of the incoming frame.
3. If the CTRL2[EACEN] bit is negated, the IDE bit of Mailbox is always compared with the IDE bit of the incoming frame.

0 The corresponding bit in the filter is "don't care."
1 The corresponding bit in the filter is checked.

1. RTR bit of the Incoming Frame. It is saved into an auxiliary MB called Rx Serial Message Buffer (Rx SMB).
2. If the CTRL2[EACEN] bit is negated, the RTR bit of Mailbox is never compared with the RTR bit of the incoming frame.
3. If the CTRL2[EACEN] bit is negated, the IDE bit of Mailbox is always compared with the IDE bit of the incoming frame.

## 48.3.6 Rx 14 Mask register (CANx_RX14MASK)

This register is located in RAM.

RX14MASK is provided for legacy application support. When the MCR[IRMQ] bit is asserted, RX14MASK has no effect.

RX14MASK is used to mask the filter fields of Message Buffer 14.

This register can only be programmed while the module is in Freeze mode as it is blocked by hardware in other modes.

Address: 4002_4000h base + 14h offset = 4002_4014h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX14M[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**CANx_RX14MASK field descriptions**

| Field | Description |
|---|---|
| 31–0<br>RX14M[31:0] | Rx Buffer 14 Mask Bits<br><br>Each mask bit masks the corresponding Mailbox 14 filter field in the same way that RXMGMASK masks other Mailboxes' filters. See the description of the CAN_RXMGMASK register. |

**CANx_RX14MASK field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 The corresponding bit in the filter is "don't care."<br>1 The corresponding bit in the filter is checked. |

## 48.3.7 Rx 15 Mask register (CANx_RX15MASK)

This register is located in RAM.

RX15MASK is provided for legacy application support. When the MCR[IRMQ] bit is asserted, RX15MASK has no effect.

RX15MASK is used to mask the filter fields of Message Buffer 15.

This register can be programmed only while the module is in Freeze mode because it is blocked by hardware in other modes.

Address: 4002_4000h base + 18h offset = 4002_4018h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX15M[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**CANx_RX15MASK field descriptions**

| Field | Description |
|---|---|
| 31–0<br>RX15M[31:0] | Rx Buffer 15 Mask Bits<br><br>Each mask bit masks the corresponding Mailbox 15 filter field in the same way that RXMGMASK masks other Mailboxes' filters. See the description of the CAN_RXMGMASK register.<br><br>0 The corresponding bit in the filter is "don't care."<br>1 The corresponding bit in the filter is checked. |

## 48.3.8 Error Counter (CANx_ECR)

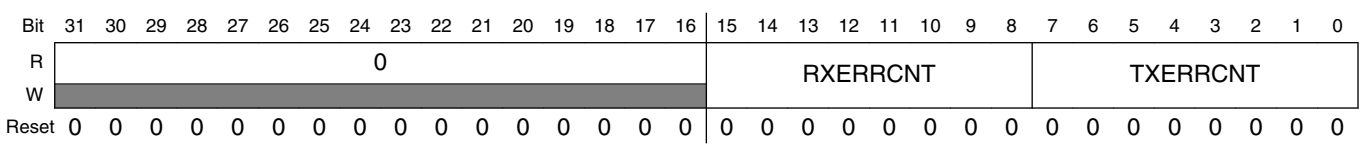
This register has two 8-bit fields reflecting the value of two FlexCAN error counters: Transmit Error Counter (TXERRCNT field) and Receive Error Counter (RXERRCNT field). The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FlexCAN module. Both counters are read-only except in Freeze mode, where they can be written by the CPU.

FlexCAN responds to any bus state as described in the protocol, for example, transmit Error Active or Error Passive flag, delay its transmission start time (Error Passive) and avoid any influence on the bus when in Bus Off state.

The following are the basic rules for FlexCAN bus state transitions:

- If the value of TXERRCNT or RXERRCNT increases to be greater than or equal to 128, the FLTCONF field in the Error and Status Register is updated to reflect ‘Error Passive’ state.
- If the FlexCAN state is ‘Error Passive’, and either TXERRCNT or RXERRCNT decrements to a value less than or equal to 127 while the other already satisfies this condition, the FLTCONF field in the Error and Status Register is updated to reflect ‘Error Active’ state.
- If the value of TXERRCNT increases to be greater than 255, the FLTCONF field in the Error and Status Register is updated to reflect ‘Bus Off’ state, and an interrupt may be issued. The value of TXERRCNT is then reset to zero.
- If FlexCAN is in ‘Bus Off’ state, then TXERRCNT is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, TXERRCNT is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the TXERRCNT. When TXERRCNT reaches the value of 128, the FLTCONF field in the Error and Status Register is updated to be ‘Error Active’ and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the TXERRCNT value.
- If during system start-up, only one node is operating, then its TXERRCNT increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ACKERR bit in the Error and Status Register). After the transition to ‘Error Passive’ state, the TXERRCNT does not increment anymore by acknowledge errors. Therefore the device never goes to the ‘Bus Off’ state.
- If the RXERRCNT increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to resume to ‘Error Active’ state.

Address: 4002_4000h base + 1Ch offset = 4002_401Ch




**CANx_ECR field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–8<br>RXERRCNT | Receive Error Counter |

*Table continues on the next page...*

**CANx_ECR field descriptions (continued)**

| Field | Description |
|---|---|
| 7–0<br>TXERRCNT | Transmit Error Counter |

## 48.3.9 Error and Status 1 register (CANx_ESR1)

This register reflects various error conditions, some general status of the device and it is the source of interrupts to the CPU.

The CPU read action clears bits 15-10. Therefore the reported error conditions (bits 15-10) are those that occurred since the last time the CPU read this register. Bits 9-3 are status bits.

The following table shows the FlexCAN state variables and their meanings. Other combinations not shown in the table are reserved.

| SYNCH | IDLE | TX | RX | FlexCAN State |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Not synchronized to CAN bus |
| 1 | 1 | x | x | Idle |
| 1 | 0 | 1 | 0 | Transmitting |
| 1 | 0 | 0 | 1 | Receiving |

Address: 4002_4000h base + 20h offset = 4002_4020h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | SYNCH | TWRNINT | RWRNINT |
| W | | | | | | | | | | | | | | | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BIT1ERR | BIT0ERR | ACKERR | CRCERR | FRMERR | STFERR | TXWRN | RXWRN | IDLE | TX | FLTCONF | | RX | BOFFINT | ERRINT | WAKINT |
| W | | | | | | | | | | | | | | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## CANx_ESR1 field descriptions

| Field | Description |
|---|---|
| 31–19<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 18<br>SYNCH | CAN Synchronization Status<br><br>This read-only flag indicates whether the FlexCAN is synchronized to the CAN bus and able to participate in the communication process. It is set and cleared by the FlexCAN. See the table in the overall CAN_ESR1 register description.<br><br>0 FlexCAN is not synchronized to the CAN bus.<br>1 FlexCAN is synchronized to the CAN bus. |
| 17<br>TWRNINT | Tx Warning Interrupt Flag<br><br>If the WRNEN bit in MCR is asserted, the TWRNINT bit is set when the TXWRN flag transitions from 0 to 1, meaning that the Tx error counter reached 96. If the corresponding mask bit in the Control Register (TWRNMSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When WRNEN is negated, this flag is masked. CPU must clear this flag before disabling the bit. Otherwise it will be set when the WRNEN is set again. Writing 0 has no effect. This flag is not generated during Bus Off state. This bit is not updated during Freeze mode.<br><br>0 No such occurrence.<br>1 The Tx error counter transitioned from less than 96 to greater than or equal to 96. |
| 16<br>RWRNINT | Rx Warning Interrupt Flag<br><br>If the WRNEN bit in MCR is asserted, the RWRNINT bit is set when the RXWRN flag transitions from 0 to 1, meaning that the Rx error counters reached 96. If the corresponding mask bit in the Control Register (RWRNMSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When WRNEN is negated, this flag is masked. CPU must clear this flag before disabling the bit. Otherwise it will be set when the WRNEN is set again. Writing 0 has no effect. This bit is not updated during Freeze mode.<br><br>0 No such occurrence.<br>1 The Rx error counter transitioned from less than 96 to greater than or equal to 96. |
| 15<br>BIT1ERR | Bit1 Error<br><br>This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. |

*Table continues on the next page...*

## CANx_ESR1 field descriptions (continued)

| Field | Description |
|---|---|
| | **NOTE:** This bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits.<br><br>0 No such occurrence.<br>1 At least one bit sent as recessive is received as dominant. |
| 14<br>BIT0ERR | Bit0 Error<br><br>This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message.<br><br>0 No such occurrence.<br>1 At least one bit sent as dominant is received as recessive. |
| 13<br>ACKERR | Acknowledge Error<br><br>This bit indicates that an Acknowledge Error has been detected by the transmitter node, that is, a dominant bit has not been detected during the ACK SLOT.<br><br>0 No such occurrence.<br>1 An ACK error occurred since last read of this register. |
| 12<br>CRCERR | Cyclic Redundancy Check Error<br><br>This bit indicates that a CRC Error has been detected by the receiver node, that is, the calculated CRC is different from the received.<br><br>0 No such occurrence.<br>1 A CRC error occurred since last read of this register. |
| 11<br>FRMERR | Form Error<br><br>This bit indicates that a Form Error has been detected by the receiver node, that is, a fixed-form bit field contains at least one illegal bit.<br><br>0 No such occurrence.<br>1 A Form Error occurred since last read of this register. |
| 10<br>STFERR | Stuffing Error<br><br>This bit indicates that a Stuffing Error has been etected.<br><br>0 No such occurrence.<br>1 A Stuffing Error occurred since last read of this register. |
| 9<br>TXWRN | TX Error Warning<br><br>This bit indicates when repetitive errors are occurring during message transmission. This bit is not updated during Freeze mode.<br><br>0 No such occurrence.<br>1 TXERRCNT is greater than or equal to 96. |
| 8<br>RXWRN | Rx Error Warning<br><br>This bit indicates when repetitive errors are occurring during message reception. This bit is not updated during Freeze mode.<br><br>0 No such occurrence.<br>1 RXERRCNT is greater than or equal to 96. |

*Table continues on the next page...*

## CANx_ESR1 field descriptions (continued)

| Field | Description |
|---|---|
| 7<br>IDLE | This bit indicates when CAN bus is in IDLE state. See the table in the overall CAN_ESR1 register description.<br><br>0 No such occurrence.<br>1 CAN bus is now IDLE. |
| 6<br>TX | FlexCAN In Transmission<br><br>This bit indicates if FlexCAN is transmitting a message. See the table in the overall CAN_ESR1 register description.<br><br>0 FlexCAN is not transmitting a message.<br>1 FlexCAN is transmitting a message. |
| 5–4<br>FLTCONF | Fault Confinement State<br><br>This 2-bit field indicates the Confinement State of the FlexCAN module.<br><br>If the LOM bit in the Control Register is asserted, after some delay that depends on the CAN bit timing the FLTCONF field will indicate “Error Passive”. The very same delay affects the way how FLTCONF reflects an update to ECR register by the CPU. It may be necessary up to one CAN bit time to get them coherent again.<br><br>Because the Control Register is not affected by soft reset, the FLTCONF field will not be affected by soft reset if the LOM bit is asserted.<br><br>00 Error Active<br>01 Error Passive<br>1x Bus Off |
| 3<br>RX | FlexCAN In Reception<br><br>This bit indicates if FlexCAN is receiving a message. See the table in the overall CAN_ESR1 register description.<br><br>0 FlexCAN is not receiving a message.<br>1 FlexCAN is receiving a message. |
| 2<br>BOFFINT | Bus Off Interrupt<br><br>This bit is set when FlexCAN enters ‘Bus Off’ state. If the corresponding mask bit in the Control Register (BOFFMSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect.<br><br>0 No such occurrence.<br>1 FlexCAN module entered Bus Off state. |
| 1<br>ERRINT | Error Interrupt<br><br>This bit indicates that at least one of the Error Bits (bits 15-10) is set. If the corresponding mask bit CTRL1[ERRMSK] is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect.<br><br>0 No such occurrence.<br>1 Indicates setting of any Error Bit in the Error and Status Register. |
| 0<br>WAKINT | Wake-Up Interrupt<br><br>This field applies when FlexCAN is in low-power mode under Self Wake Up mechanism:<br>• Stop mode |

*Table continues on the next page...*

**CANx_ESR1 field descriptions (continued)**

| Field | Description |
|---|---|
| | When a recessive-to-dominant transition is detected on the CAN bus and if the MCR[WAKMSK] bit is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1.<br>When MCR[SLFWAK] is negated, this flag is masked. The CPU must clear this flag before disabling the bit. Otherwise it will be set when the SLFWAK is set again. Writing 0 has no effect.<br>0 No such occurrence.<br>1 Indicates a recessive to dominant transition was received on the CAN bus. |

## 48.3.10 Interrupt Masks 1 register (CANx_IMASK1)

This register allows any number of a range of the 32 Message Buffer Interrupts to be enabled or disabled for MB31 to MB0. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception, that is, when the corresponding IFLAG1 bit is set.

Address: 4002_4000h base + 28h offset = 4002_4028h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BUFLM | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CANx_IMASK1 field descriptions**

| Field | Description |
|---|---|
| 31–0<br>BUFLM | Buffer MB $_i$ Mask<br>Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt for MB31 to MB0.<br>**NOTE:** Setting or clearing a bit in the IMASK1 Register can assert or negate an interrupt request, if the corresponding IFLAG1 bit is set.<br>0 The corresponding buffer Interrupt is disabled.<br>1 The corresponding buffer Interrupt is enabled. |

## 48.3.11 Interrupt Flags 1 register (CANx_IFLAG1)

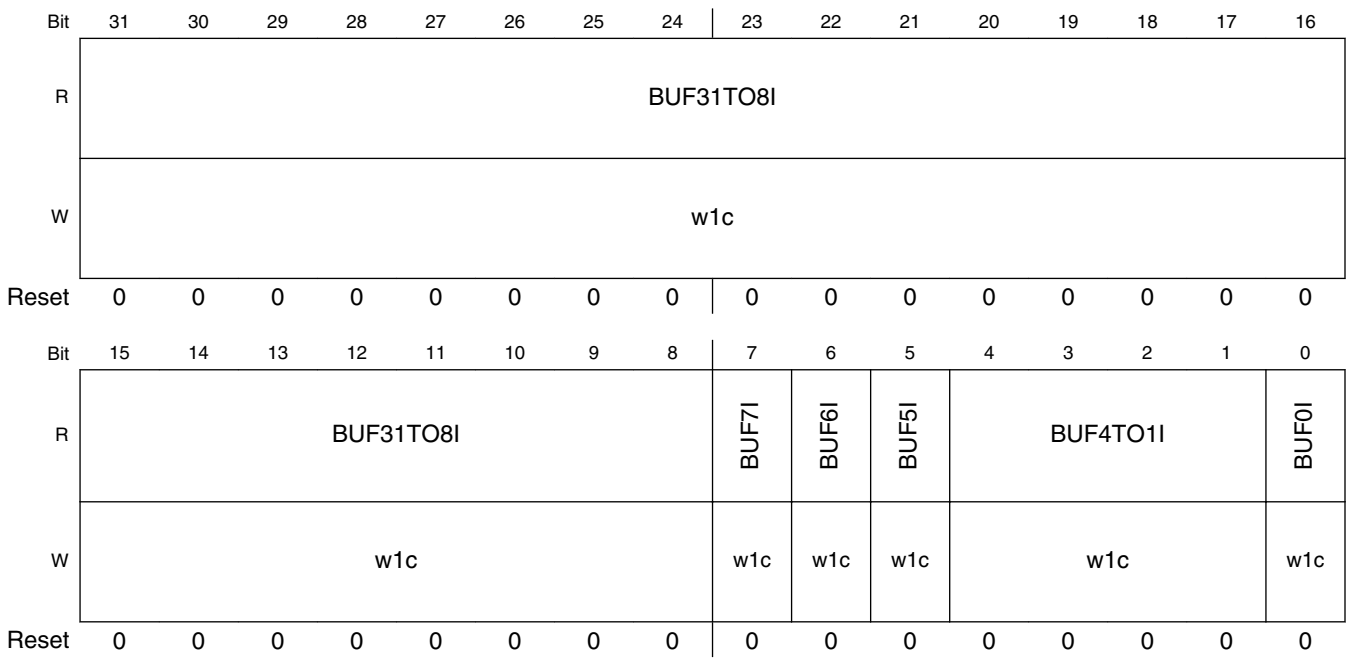
This register defines the flags for the 32 Message Buffer interrupts for MB31 to MB0. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG1 bit. If the corresponding IMASK1 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.

The BUF7I to BUF5I flags are also used to represent FIFO interrupts when the Rx FIFO is enabled. When the bit MCR[RFEN] is set, the function of the 8 least significant interrupt flags BUF[7:0]I changes: BUF7I, BUF6I and BUF5I indicate operating conditions of the FIFO, and the BUF4TO0I field is reserved.

Before enabling the RFEN, the CPU must service the IFLAG bits asserted in the Rx FIFO region; see Section "Rx FIFO". Otherwise, these IFLAG bits will mistakenly show the related MBs now belonging to FIFO as having contents to be serviced. When the RFEN bit is negated, the FIFO flags must be cleared. The same care must betaken when an RFFN value is selected extending Rx FIFO filters beyond MB7. For example, when RFFN is 0x8, the MB0-23 range is occupied by Rx FIFO filters and related IFLAG bits must be cleared.

Before updating MCR[MAXMB] field, CPU must service the IFLAG1 bits whose MB value is greater than the MCR[MAXMB] to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

Address: 4002_4000h base + 30h offset = 4002_4030h




## CANx_IFLAG1 field descriptions

| Field | Description |
|---|---|
| 31–8<br>BUF31TO8I | Buffer $MB_i$ Interrupt<br><br>Each bit flags the corresponding FlexCAN Message Buffer interrupt for MB31 to MB8.<br><br>0 The corresponding buffer has no occurrence of successfully completed transmission or reception.<br>1 The corresponding buffer has successfully completed transmission or reception. |

*Table continues on the next page...*

## CANx_IFLAG1 field descriptions (continued)

| Field | Description |
|---|---|
| 7<br>BUF7I | Buffer MB7 Interrupt Or "Rx FIFO Overflow"<br><br>When the RFEN bit in the MCR is cleared (Rx FIFO disabled), this bit flags the interrupt for MB7.<br><br>**NOTE:** This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes.<br><br>The BUF7I flag represents "Rx FIFO Overflow" when MCR[RFEN] is set. In this case, the flag indicates that a message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox.<br><br>0 No occurrence of MB7 completing transmission/reception when MCR[RFEN]=0, or of Rx FIFO overflow when MCR[RFEN]=1<br>1 MB7 completed transmission/reception when MCR[RFEN]=0, or Rx FIFO overflow when MCR[RFEN]=1 |
| 6<br>BUF6I | Buffer MB6 Interrupt Or "Rx FIFO Warning"<br><br>When the RFEN bit in the MCR is cleared (Rx FIFO disabled), this bit flags the interrupt for MB6.<br><br>**NOTE:** This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes.<br><br>The BUF6I flag represents "Rx FIFO Warning" when MCR[RFEN] is set. In this case, the flag indicates when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. Note that if the flag is cleared while the number of unread messages is greater than 4, it does not assert again until the number of unread messages within the Rx FIFO is decreased to be equal to or less than 4.<br><br>0 No occurrence of MB6 completing transmission/reception when MCR[RFEN]=0, or of Rx FIFO almost full when MCR[RFEN]=1<br>1 MB6 completed transmission/reception when MCR[RFEN]=0, or Rx FIFO almost full when MCR[RFEN]=1 |
| 5<br>BUF5I | Buffer MB5 Interrupt Or "Frames available in Rx FIFO"<br><br>When the RFEN bit in the MCR is cleared (Rx FIFO disabled), this bit flags the interrupt for MB5.<br><br>**NOTE:** This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes.<br><br>The BUF5I flag represents "Frames available in Rx FIFO" when MCR[RFEN] is set. In this case, the flag indicates that at least one frame is available to be read from the Rx FIFO.<br><br>0 No occurrence of MB5 completing transmission/reception when MCR[RFEN]=0, or of frame(s) available in the FIFO, when MCR[RFEN]=1<br>1 MB5 completed transmission/reception when MCR[RFEN]=0, or frame(s) available in the Rx FIFO when MCR[RFEN]=1 |
| 4–1<br>BUF4TO1I | Buffer MB $_i$ Interrupt Or "reserved"<br><br>When the RFEN bit in the MCR is cleared (Rx FIFO disabled), these bits flag the interrupts for MB4 to MB1.<br><br>**NOTE:** These flags are cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes.<br><br>The BUF4TO1I flags are reserved when MCR[RFEN] is set.<br><br>0 The corresponding buffer has no occurrence of successfully completed transmission or reception when MCR[RFEN]=0.<br>1 The corresponding buffer has successfully completed transmission or reception when MCR[RFEN]=0. |
| 0<br>BUF0I | Buffer MB0 Interrupt Or "reserved"<br><br>When the RFEN bit in the MCR is cleared (Rx FIFO disabled), this bit flags the interrupt for MB0. |

*Table continues on the next page...*

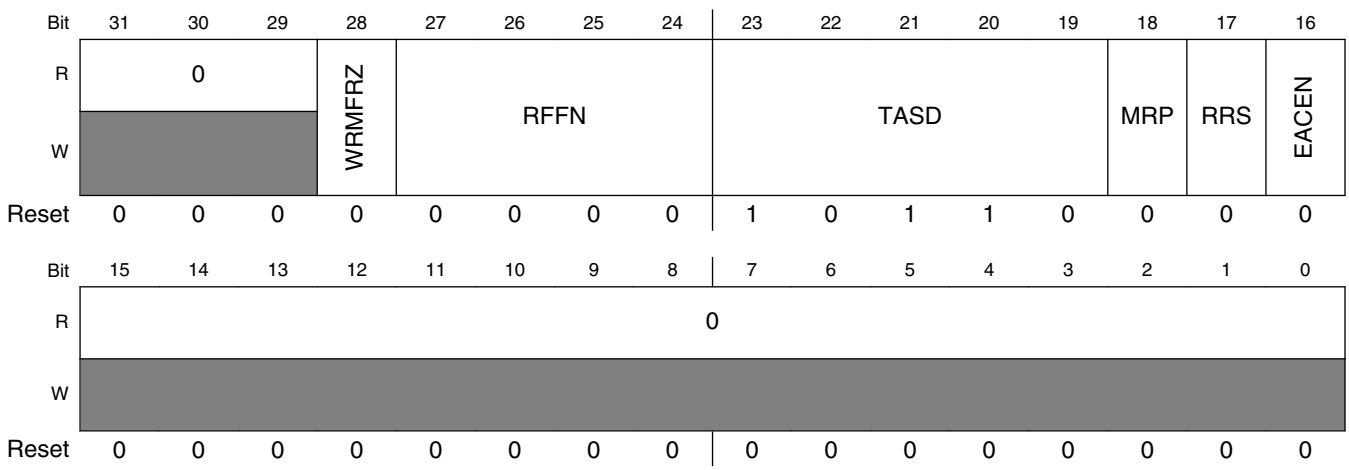
**CANx_IFLAG1 field descriptions (continued)**

| Field | Description |
|---|---|
| | **NOTE:** This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes.<br>The BUF0I flag is reserved when MCR[RFEN] is set.<br>0 The corresponding buffer has no occurrence of successfully completed transmission or reception when MCR[RFEN]=0.<br>1 The corresponding buffer has successfully completed transmission or reception when MCR[RFEN]=0. |

## 48.3.12 Control 2 register (CANx_CTRL2)

This register contains control bits for CAN errors, FIFO features, and mode selection.

Address: 4002_4000h base + 34h offset = 4002_4034h




**CANx_CTRL2 field descriptions**

| Field | Description |
|---|---|
| 31–29<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 28<br>WRMFRZ | Write-Access To Memory In Freeze Mode<br>Enable unrestricted write access to FlexCAN memory in Freeze mode. This bit can only be written in Freeze mode and has no effect out of Freeze mode.<br>0 Maintain the write access restrictions.<br>1 Enable unrestricted write access to FlexCAN memory. |
| 27–24<br>RFFN | Number Of Rx FIFO Filters<br>This 4-bit field defines the number of Rx FIFO filters, as shown in the following table. The maximum selectable number of filters is determined by the MCU. This field can only be written in Freeze mode as it is blocked by hardware in other modes. This field must not be programmed with values that make the number of Message Buffers occupied by Rx FIFO and ID Filter exceed the number of Mailboxes present, defined by MCR[MAXMB]. |

*Table continues on the next page...*

## CANx_CTRL2 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
NOTE: Each group of eight filters occupies a memory space equivalent to two Message Buffers which means that the more filters are implemented the less Mailboxes will be available.<br><br>
Considering that the Rx FIFO occupies the memory space originally reserved for MB0-5, RFFN should be programmed with a value correponding to a number of filters not greater than the number of available memory words which can be calculated as follows:<br><br>
(SETUP_MB - 6) × 4<br><br>
where SETUP_MB is the least between NUMBER_OF_MB and MAXMB.<br><br>
The number of remaining Mailboxes available will be:<br><br>
(SETUP_MB - 8) - (RFFN × 2)<br><br>
If the Number of Rx FIFO Filters programmed through RFFN exceeds the SETUP_MB value (memory space available) the exceeding ones will not be functional.<br><br>
<table>
<tr><th>RFFN[3:0]</th><th>Number of Rx FIFO filters</th><th>Message Buffers occupied by Rx FIFO and ID Filter Table</th><th>Remaining Available Mailboxes[1]</th><th>Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks[2]</th><th>Rx FIFO ID Filter Table Elements Affected by Rx FIFO Global Mask[2]</th></tr>
<tr><td>0x0</td><td>8</td><td>MB 0-7</td><td>MB 8-63</td><td>Elements 0-7</td><td>none</td></tr>
<tr><td>0x1</td><td>16</td><td>MB 0-9</td><td>MB 10-63</td><td>Elements 0-9</td><td>Elements 10-15</td></tr>
<tr><td>0x2</td><td>24</td><td>MB 0-11</td><td>MB 12-63</td><td>Elements 0-11</td><td>Elements 12-23</td></tr>
<tr><td>0x3</td><td>32</td><td>MB 0-13</td><td>MB 14-63</td><td>Elements 0-13</td><td>Elements 14-31</td></tr>
<tr><td>0x4</td><td>40</td><td>MB 0-15</td><td>MB 16-63</td><td>Elements 0-15</td><td>Elements 16-39</td></tr>
<tr><td>0x5</td><td>48</td><td>MB 0-17</td><td>MB 18-63</td><td>Elements 0-17</td><td>Elements 18-47</td></tr>
<tr><td>0x6</td><td>56</td><td>MB 0-19</td><td>MB 20-63</td><td>Elements 0-19</td><td>Elements 20-55</td></tr>
<tr><td>0x7</td><td>64</td><td>MB 0-21</td><td>MB 22-63</td><td>Elements 0-21</td><td>Elements 22-63</td></tr>
<tr><td>0x8</td><td>72</td><td>MB 0-23</td><td>MB 24-63</td><td>Elements 0-23</td><td>Elements 24-71</td></tr>
<tr><td>0x9</td><td>80</td><td>MB 0-25</td><td>MB 26-63</td><td>Elements 0-25</td><td>Elements 26-79</td></tr>
<tr><td>0xA</td><td>88</td><td>MB 0-27</td><td>MB 28-63</td><td>Elements 0-27</td><td>Elements 28-87</td></tr>
<tr><td>0xB</td><td>96</td><td>MB 0-29</td><td>MB 30-63</td><td>Elements 0-29</td><td>Elements 30-95</td></tr>
<tr><td>0xC</td><td>104</td><td>MB 0-31</td><td>MB 32-63</td><td>Elements 0-31</td><td>Elements 32-103</td></tr>
<tr><td>0xD</td><td>112</td><td>MB 0-33</td><td>MB 34-63</td><td>Elements 0-31</td><td>Elements 32-111</td></tr>
<tr><td>0xE</td><td>120</td><td>MB 0-35</td><td>MB 36-63</td><td>Elements 0-31</td><td>Elements 32-119</td></tr>
<tr><td>0xF</td><td>128</td><td>MB 0-37</td><td>MB 38-63</td><td>Elements 0-31</td><td>Elements 32-127</td></tr>
</table>
1. The number of the last remaining available mailboxes is defined by the least value between the parameter NUMBER_OF_MB minus 1 and the MCR[MAXMB] field.<br>
2. If Rx Individual Mask Registers are not enabled then all Rx FIFO filters are affected by the Rx FIFO Global Mask.
</td></tr>
<tr><td>23–19<br>TASD</td><td>Tx Arbitration Start Delay<br>This 5-bit field indicates how many CAN bits the Tx arbitration process start point can be delayed from the first bit of CRC field on CAN bus. This field can be written only in Freeze mode because it is blocked by hardware in other modes.</td></tr>
</table>

*Table continues on the next page...*

## CANx_CTRL2 field descriptions (continued)

| Field | Description |
|---|---|
| | This field is useful to optimize the transmit performance based on factors such as: peripheral/serial clock ratio, CAN bit timing and number of MBs. The duration of an arbitration process, in terms of CAN bits, is directly proportional to the number of available MBs and CAN baud rate and inversely proportional to the peripheral clock frequency.<br><br>The optimal arbitration timing is that in which the last MB is scanned right before the first bit of the Intermission field of a CAN frame. Therefore, if there are few MBs and the system/serial clock ratio is high and the CAN baud rate is low then the arbitration can be delayed and vice-versa.<br><br>If TASD is 0 then the arbitration start is not delayed, thus the CPU has less time to configure a Tx MB for the next arbitration, but more time is reserved for arbitration. On the other hand, if TASD is 24 then the CPU can configure a Tx MB later and less time is reserved for arbitration.<br><br>If too little time is reserved for arbitration the FlexCAN may be not able to find winner MBs in time to compete with other nodes for the CAN bus. If the arbitration ends too much time before the first bit of Intermission field then there is a chance that the CPU reconfigures some Tx MBs and the winner MB is not the best to be transmitted.<br><br>The optimal configuration for TASD can be calculated as:<br><br>$$TASD = 25 - \{f_{CANCLK} \times [MAXMB + 3 - (RFEN \times 8) - (RFEN \times RFFN \times 2)] \times 2\} / \{f_{SYS} \times [1+(PSEG1+1)+(PSEG2+1)+(PROPSEG+1)] \times (PRESDIV+1)\}$$<br><br>where:<br>• $f_{CANCLK}$ is the Protocol Engine (PE) Clock (see section "Protocol Timing"), in Hz<br>• $f_{SYS}$ is the peripheral clock, in Hz<br>• MAXMB is the value in CTRL1[MAXMB] field<br>• RFEN is the value in CTRL1[RFEN] bit<br>• RFFN is the value in CTRL2[RFFN] field<br>• PSEG1 is the value in CTRL1[PSEG1] field<br>• PSEG2 is the value in CTRL1[PSEG2] field<br>• PROPSEG is the value in CTRL1[PROPSEG] field<br>• PRESDIV is the value in CTRL1[PRESDIV] field<br><br>See Section "Arbitration process" and Section "Protocol Timing" for more details. |
| 18<br>MRP | Mailboxes Reception Priority<br><br>If this bit is set the matching process starts from the Mailboxes and if no match occurs the matching continues on the Rx FIFO. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br><br>0 Matching starts from Rx FIFO and continues on Mailboxes.<br>1 Matching starts from Mailboxes and continues on Rx FIFO. |
| 17<br>RRS | Remote Request Storing<br><br>If this bit is asserted Remote Request Frame is submitted to a matching process and stored in the corresponding Message Buffer in the same fashion of a Data Frame. No automatic Remote Response Frame will be generated.<br><br>If this bit is negated the Remote Request Frame is submitted to a matching process and an automatic Remote Response Frame is generated if a Message Buffer with CODE=0b1010 is found with the same ID.<br><br>This bit can be written only in Freeze mode because it is blocked by hardware in other modes. |

*Table continues on the next page...*

**CANx_CTRL2 field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Remote Response Frame is generated.<br>1 Remote Request Frame is stored. |
| 16<br>EACEN | Entire Frame Arbitration Field Comparison Enable For Rx Mailboxes<br><br>This bit controls the comparison of IDE and RTR bits whithin Rx Mailboxes filters with their corresponding bits in the incoming frame by the matching process. This bit does not affect matching for Rx FIFO. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br><br>0 Rx Mailbox filter's IDE bit is always compared and RTR is never compared despite mask bits.<br>1 Enables the comparison of both Rx Mailbox filter's IDE and RTR bit with their corresponding bits within the incoming frame. Mask bits do apply. |
| 15–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

1. The number of the last remaining available mailboxes is defined by the least value between the parameter NUMBER_OF_MB minus 1 and the MCR[MAXMB] field.
2. If Rx Individual Mask Registers are not enabled then all Rx FIFO filters are affected by the Rx FIFO Global Mask.

## 48.3.13 Error and Status 2 register (CANx_ESR2)

This register reflects various interrupt flags and some general status.

Address: 4002_4000h base + 38h offset = 4002_4038h

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

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

**CANx_ESR2 field descriptions**

| Field | Description |
|---|---|
| 31–23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22–16<br>LPTM | Lowest Priority Tx Mailbox<br><br>If ESR2[VPS] is asserted, this field indicates the lowest number inactive Mailbox (see the IMB bit description). If there is no inactive Mailbox then the Mailbox indicated depends on CTRL1[LBUF] bit value. If CTRL1[LBUF] bit is negated then the Mailbox indicated is the one that has the greatest arbitration value (see the "Highest priority Mailbox first" section). If CTRL1[LBUF] bit is asserted then the Mailbox indicated is the highest number active Tx Mailbox. If a Tx Mailbox is being transmitted it is not considered in LPTM calculation. If ESR2[IMB] is not asserted and a frame is transmitted successfully, LPTM is updated with its Mailbox number. |

*Table continues on the next page...*

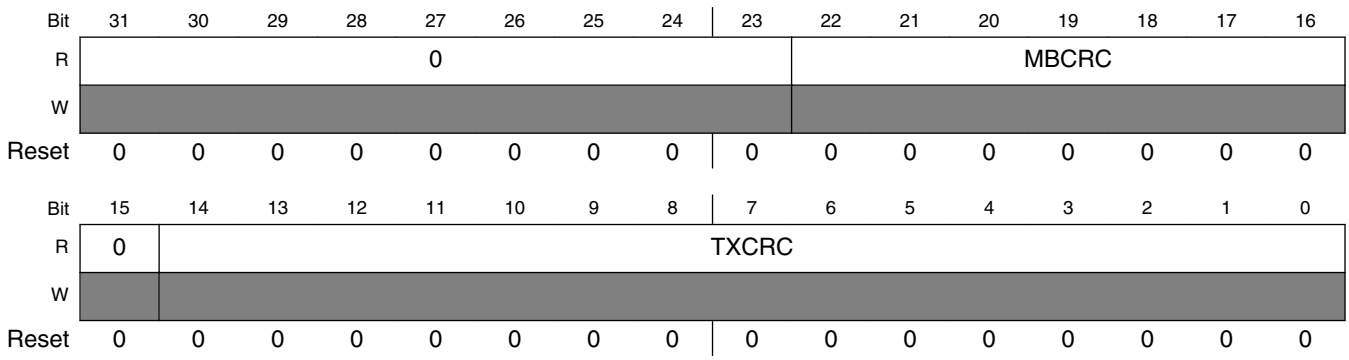
## CANx_ESR2 field descriptions (continued)

| Field | Description |
|---|---|
| 15<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 14<br>VPS | Valid Priority Status<br><br>This bit indicates whether IMB and LPTM contents are currently valid or not. VPS is asserted upon every complete Tx arbitration process unless the CPU writes to Control and Status word of a Mailbox that has already been scanned, that is, it is behind Tx Arbitration Pointer, during the Tx arbitration process. If there is no inactive Mailbox and only one Tx Mailbox that is being transmitted then VPS is not asserted. VPS is negated upon the start of every Tx arbitration process or upon a write to Control and Status word of any Mailbox.<br><br>**NOTE:** ESR2[VPS] is not affected by any CPU write into Control Status (C/S) of a MB thatis blocked by abort mechanism. When MCR[AEN] is asserted, the abort code write in C/S of a MB that is being transmitted (pending abort), or any write attempt into a Tx MB with IFLAG set is blocked.<br><br>0 Contents of IMB and LPTM are invalid.<br>1 Contents of IMB and LPTM are valid. |
| 13<br>IMB | Inactive Mailbox<br><br>If ESR2[VPS] is asserted, this bit indicates whether there is any inactive Mailbox (CODE field is either 0b1000 or 0b0000). This bit is asserted in the following cases:<br>• During arbitration, if an LPTM is found and it is inactive.<br>• If IMB is not asserted and a frame is transmitted successfully.<br><br>This bit is cleared in all start of arbitration (see Section "Arbitration process").<br><br>**NOTE:** LPTM mechanism have the following behavior: if an MB is successfully transmitted and ESR2[IMB]=0 (no inactive Mailbox), then ESR2[VPS] and ESR2[IMB] are asserted and the index related to the MB just transmitted is loaded into ESR2[LPTM].<br><br>0 If ESR2[VPS] is asserted, the ESR2[LPTM] is not an inactive Mailbox.<br>1 If ESR2[VPS] is asserted, there is at least one inactive Mailbox. LPTM content is the number of the first one. |
| 12–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 48.3.14 CRC Register (CANx_CRCR)

This register provides information about the CRC of transmitted messages.

Address: 4002_4000h base + 44h offset = 4002_4044h




**CANx_CRCR field descriptions**

| Field | Description |
|---|---|
| 31–23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22–16<br>MBCRC | CRC Mailbox<br><br>This field indicates the number of the Mailbox corresponding to the value in TXCRC field. |
| 15<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 14–0<br>TXCRC | CRC Transmitted<br><br>This field indicates the CRC value of the last message transmitted. This field is updated at the same time the Tx Interrupt Flag is asserted. |

## 48.3.15 Rx FIFO Global Mask register (CANx_RXFGMASK)

This register is located in RAM.

If Rx FIFO is enabled RXFGMASK is used to mask the Rx FIFO ID Filter Table elements that do not have a corresponding RXIMR according to CTRL2[RFFN] field setting.

This register can only be written in Freeze mode as it is blocked by hardware in other modes.

Address: 4002_4000h base + 48h offset = 4002_4048h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FGM[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### CANx_RXFGMASK field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>31–0<br>FGM[31:0]</td><td>Rx FIFO Global Mask Bits<br><br>These bits mask the ID Filter Table elements bits in a perfect alignment.<br><br>The following table shows how the FGM bits correspond to each IDAF field.<br><br>
<table>
<tr><th rowspan="2">Rx FIFO ID Filter Table Elements Format (MCR[IDAM])</th><th colspan="6">Identifier Acceptance Filter Fields</th></tr>
<tr><th>RTR</th><th>IDE</th><th>RXIDA</th><th>RXIDB [1]</th><th>RXIDC [2]</th><th>Reserved</th></tr>
<tr><td>A</td><td>FGM[31]</td><td>FGM[30]</td><td>FGM[29:1]</td><td>-</td><td>-</td><td>FGM[0]</td></tr>
<tr><td>B</td><td>FGM[31],<br>FGM[15]</td><td>FGM[30],<br>FGM[14]</td><td>-</td><td>FGM[29:16],<br>FGM[13:0]</td><td></td><td>-</td></tr>
<tr><td>C</td><td>-</td><td>-</td><td></td><td>-</td><td>FGM[31:24],<br>FGM[23:16],<br>FGM[15:8],<br>FGM[7:0]</td><td></td></tr>
</table>
1. If MCR[IDAM] field is equivalent to the format B only the fourteen most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.<br>2. If MCR[IDAM] field is equivalent to the format C only the eight most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.<br><br>0 The corresponding bit in the filter is "don’t care."<br>1 The corresponding bit in the filter is checked.</td></tr>
</table>

1. If MCR[IDAM] field is equivalent to the format B only the fourteen most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.
2. If MCR[IDAM] field is equivalent to the format C only the eight most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.

## 48.3.16 Rx FIFO Information Register (CANx_RXFIR)

RXFIR provides information on Rx FIFO.

This register is the port through which the CPU accesses the output of the RXFIR FIFO located in RAM. The RXFIR FIFO is written by the FlexCAN whenever a new message is moved into the Rx FIFO as well as its output is updated whenever the output of the Rx FIFO is updated with the next message. See Section "Rx FIFO" for instructions on reading this register.

Address: 4002_4000h base + 4Ch offset = 4002_404Ch

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | | | | | | | | IDHIT | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**CANx_RXFIR field descriptions**

| Field | Description |
|---|---|
| 31–9<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 8–0<br>IDHIT | Identifier Acceptance Filter Hit Indicator<br><br>This field indicates which Identifier Acceptance Filter was hit by the received message that is in the output of the Rx FIFO. If multiple filters match the incoming message ID then the first matching IDAF found (lowest number) by the matching process is indicated. This field is valid only while the IFLAG[BUF5I] is asserted. |

## 48.3.17 Rx Individual Mask Registers (CANx_RXIMR*n*)

These registers are located in RAM.

RXIMR are used as acceptance masks for ID filtering in Rx MBs and the Rx FIFO. If the Rx FIFO is not enabled, one mask register is provided for each available Mailbox, providing ID masking capability on a per Mailbox basis.

When the Rx FIFO is enabled (MCR[RFEN] bit is asserted), up to 32 Rx Individual Mask Registers can apply to the Rx FIFO ID Filter Table elements on a one-to-one correspondence depending on the setting of CTRL2[RFFN].

RXIMR can only be written by the CPU while the module is in Freeze mode; otherwise, they are blocked by hardware.

The Individual Rx Mask Registers are not affected by reset and must be explicitly initialized prior to any reception.

Address: 4002_4000h base + 880h offset + (4d × i), where i=0d to 15d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MI[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* | x* |

* Notes:
- x = Undefined at reset.

**CANx_RXIMR*n* field descriptions**

| Field | Description |
|---|---|
| 31–0<br>MI[31:0] | Individual Mask Bits<br><br>Each Individual Mask Bit masks the corresponding bit in both the Mailbox filter and Rx FIFO ID Filter Table element in distinct ways.<br><br>For Mailbox filters, see the RXMGMASK register description.<br><br>For Rx FIFO ID Filter Table elements, see the RXFGMASK register description.<br><br>0 The corresponding bit in the filter is "don't care."<br>1 The corresponding bit in the filter is checked. |

## 48.3.34 Message buffer structure

The message buffer structure used by the FlexCAN module is represented in the following figure. Both Extended (29-bit identifier) and Standard (11-bit identifier) frames used in the CAN specification (Version 2.0 Part B) are represented. Each individual MB is formed by 16 bytes.

The memory area from 0x80 to 0x17F is used by the mailboxes.

**Table 48-69. Message buffer structure**

| | 31 30 29 28 | 27 24 | 23 | 22 | 21 | 20 | 19 18 17 16 | 15 8 | 7 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0 | | CODE | | SRR | IDE | RTR | DLC | TIME STAMP | |
| 0x4 | PRIO | ID (Standard/Extended) | | | | | ID (Extended) | | |
| 0x8 | Data Byte 0 | | Data Byte 1 | | | | | Data Byte 2 | Data Byte 3 |
| 0xC | Data Byte 4 | | Data Byte 5 | | | | | Data Byte 6 | Data Byte 7 |
| | = Unimplemented or Reserved | | | | | | | | |

**CODE** — Message Buffer Code

This 4-bit field can be accessed (read or write) by the CPU and by the FlexCAN module itself, as part of the message buffer matching and arbitration process. The encoding is shown in Table 48-70 and Table 48-71. See Functional description for additional information.

## Table 48-70. Message buffer code for Rx buffers

| CODE description | Rx code BEFORE receive new frame | SRV[1] | Rx code AFTER successful reception[2] | RRS[3] | Comment |
|---|---|---|---|---|---|
| 0b0000: INACTIVE — MB is not active. | INACTIVE | — | — | — | MB does not participate in the matching process. |
| 0b0100: EMPTY — MB is active and empty. | EMPTY | — | FULL | — | When a frame is received successfully (after the Move-in) process), the CODE field is automatically updated to FULL. |
| 0b0010: FULL — MB is full. | FULL | Yes | FULL | — | The act of reading the C/S word followed by unlocking the MB (SRV) does not make the code return to EMPTY. It remains FULL. If a new frame is moved to the MB after the MB was serviced, the code still remains FULL. See Matching process for matching details related to FULL code. |
| | | No | OVERRUN | — | If the MB is FULL and a new frame is moved to this MB before the CPU services it, the CODE field is automatically updated to OVERRUN. See Matching process for details about overrun behavior. |

*Table continues on the next page...*

## Table 48-70. Message buffer code for Rx buffers (continued)

| CODE description | Rx code BEFORE receive new frame | SRV[1] | Rx code AFTER successful reception[2] | RRS[3] | Comment |
|---|---|---|---|---|---|
| 0b0110: OVERRUN — MB is being overwritten into a full buffer. | OVERRUN | Yes | FULL | — | If the CODE field indicates OVERRUN and CPU has serviced the MB, when a new frame is moved to the MB then the code returns to FULL. |
| | | No | OVERRUN | — | If the CODE field already indicates OVERRUN, and another new frame must be moved, the MB will be overwritten again, and the code will remain OVERRUN. See Matching process for details about overrun behavior. |
| 0b1010: RANSWER[4] — A frame was configured to recognize a Remote Request Frame and transmit a Response Frame in return. | RANSWER | — | TANSWER(0b1110) | 0 | A Remote Answer was configured to recognize a remote request frame received. After that an MB is set to transmit a response frame. The code is automatically changed to TANSWER (0b1110). See Matching process for details. If CTRL2[RRS] is negated, transmit a response frame whenever a remote request frame with the same ID is received. |
| | | — | — | 1 | This code is ignored during matching and arbitration process. See Matching process for details. |

*Table continues on the next page...*

## Table 48-70. Message buffer code for Rx buffers (continued)

| CODE description | Rx code BEFORE receive new frame | SRV[1] | Rx code AFTER successful reception[2] | RRS[3] | Comment |
|---|---|---|---|---|---|
| CODE[0]=1b1: BUSY — FlexCAN is updating the contents of the MB. The CPU must not access the MB. | BUSY[5] | — | FULL | — | Indicates that the MB is being updated. It will be negated automatically and does not interfere with the next CODE. |
| | | — | OVERRUN | — | |

1. SRV: Serviced MB. MB was read and unlocked by reading TIMER or other MB.
2. A frame is considered a successful reception after the frame to be moved to MB (move-in process). See Move-in for details.
3. Remote Request Stored bit from CTRL2 register. See section "Control 2 Register (CTRL2)" for details.
4. Code 0b1010 is not considered Tx and an MB with this code should not be aborted.
5. Note that for Tx MBs, the BUSY bit should be ignored upon read, except when AEN bit is set in the MCR register. If this bit is asserted, the corresponding MB does not participate in the matching process.

## Table 48-71. Message buffer code for Tx buffers

| CODE Description | Tx Code BEFORE tx frame | MB RTR | Tx Code AFTER successful transmission | Comment |
|---|---|---|---|---|
| 0b1000: INACTIVE — MB is not active | INACTIVE | — | — | MB does not participate in arbitration process. |
| 0b1001: ABORT — MB is aborted | ABORT | — | — | MB does not participate in arbitration process. |
| 0b1100: DATA — MB is a Tx Data Frame (MB RTR must be 0) | DATA | 0 | INACTIVE | Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state. |
| 0b1100: REMOTE — MB is a Tx Remote Request Frame (MB RTR must be 1) | REMOTE | 1 | EMPTY | Transmit remote request frame unconditionally once. After transmission, the MB automatically becomes an Rx Empty MB with the same ID. |

*Table continues on the next page...*

**Table 48-71. Message buffer code for Tx buffers (continued)**

| CODE Description | Tx Code BEFORE tx frame | MB RTR | Tx Code AFTER successful transmission | Comment |
|---|---|---|---|---|
| 0b1110: TANSWER — MB is a Tx Response Frame from an incoming Remote Request Frame | TANSWER | — | RANSWER | This is an intermediate code that is automatically written to the MB by the CHI as a result of a match to a remote request frame. The remote response frame will be transmitted unconditionally once, and then the code will automatically return to RANSWER (0b1010). The CPU can also write this code with the same effect. The remote response frame can be either a data frame or another remote request frame depending on the RTR bit value. See Matching process and Arbitration process for details. |

**SRR** — Substitute Remote Request

Fixed recessive bit, used only in extended format. It must be set to one by the user for transmission (Tx Buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If FlexCAN receives this bit as dominant, then it is interpreted as an arbitration loss.

1 = Recessive value is compulsory for transmission in extended format frames

0 = Dominant is not a valid value for transmission in extendedfFormat frames

**IDE** — ID Extended Bit

This field identifies whether the frame format is standard or extended.

1 = Frame format is extended

0 = Frame format is standard

**RTR** — Remote Transmission Request

This bit affects the behavior of remote frames and is part of the reception filter. See Table 48-70, Table 48-71, and the description of the RRS bit in Control 2 Register (CTRL2) for additional details.

If FlexCAN transmits this bit as '1' (recessive) and receives it as '0' (dominant), it is interpreted as an arbitration loss. If this bit is transmitted as '0' (dominant), then if it is received as '1' (recessive), the FlexCAN module treats it as a bit error. If the value received matches the value transmitted, it is considered a successful bit transmission.

1 = Indicates the current MB may have a remote request frame to be transmitted if MB is Tx. If the MB is Rx then incoming remote request frames may be stored.

0 = Indicates the current MB has a data frame to be transmitted.. In Rx MB it may be considered in matching processes.

**DLC** — Length of Data in Bytes

This 4-bit field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x8 through 0xF of the MB space (see the Table 48-69). In reception, this field is written by the FlexCAN module, copied from the DLC (Data Length Code) field of the received frame. In transmission, this field is written by the CPU and corresponds to the DLC field value of the frame to be transmitted. When RTR = 1, the frame to be transmitted is a remote frame and does not include the data field, regardless of the DLC field (see Table 48-72).

**TIME STAMP** — Free-Running Counter Time Stamp

This 16-bit field is a copy of the Free-Running Timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the CAN bus.

**PRIO** — Local priority

This 3-bit field is used only when LPRIO_EN bit is set in MCR, and it only makes sense for Tx mailboxes. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See Arbitration process.

**ID** — Frame Identifier

In standard frame format, only the 11 most significant bits (28 to 18) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In extended frame format, all bits are used for frame identification in both receive and transmit cases.

**DATA BYTE 0–7** — Data Field

Up to eight bytes can be used for a data frame.

For Rx frames, the data is stored as it is received from the CAN bus. DATA BYTE (*n*) is valid only if *n* is less than DLC as shown in the table below.

For Tx frames, the CPU prepares the data field to be transmitted within the frame.

**Table 48-72. DATA BYTEs validity**

| DLC | Valid DATA BYTEs |
|---|---|
| 0 | none |
| 1 | DATA BYTE 0 |
| 2 | DATA BYTE 0–1 |
| 3 | DATA BYTE 0–2 |
| 4 | DATA BYTE 0–3 |
| 5 | DATA BYTE 0–4 |
| 6 | DATA BYTE 0–5 |
| 7 | DATA BYTE 0–6 |
| 8 | DATA BYTE 0–7 |

## 48.3.35 Rx FIFO structure

When the MCR[RFEN] bit is set, the memory area from 0x80 to 0xDC (which is normally occupied by MBs 0–5) is used by the reception FIFO engine.

The region 0x80-0x8C contains the output of the FIFO which must be read by the CPU as a message buffer. This output contains the oldest message that has been received but not yet read. The region 0x90-0xDC is reserved for internal use of the FIFO engine.

An additional memory area, which starts at 0xE0 and may extend up to 0x2DC (normally occupied by MBs 6–37) depending on the CTRL2[RFFN] field setting, contains the ID filter table (configurable from 8 to 128 table elements) that specifies filtering criteria for accepting frames into the FIFO.

Out of reset, the ID filter table flexible memory area defaults to 0xE0 and extends only to 0xFC, which corresponds to MBs 6 to 7 for RFFN = 0, for backward compatibility with previous versions of FlexCAN.

The following shows the Rx FIFO data structure.

**Table 48-73. Rx FIFO structure**

| | 31 28 24 23 | 22 | 21 | 20 | 19 18 17 16 | 15 8 | 7 0 |
|---|---|---|---|---|---|---|---|
| 0x80 | | SRR | IDE | RTR | DLC | TIME STAMP | |
| 0x84 | | ID standard | | | | ID extended | |
| 0x88 | Data byte 0 | Data byte 1 | | | | Data byte 2 | Data byte 3 |
| 0x8C | Data byte 4 | Data byte 5 | | | | Data byte 6 | Data byte 7 |
| 0x90 to 0xDC | Reserved | | | | | | |

*Table continues on the next page...*

**Table 48-73. Rx FIFO structure (continued)**

| | |
|---|---|
| 0xE0 | ID filter table element 0 |
| 0xE4 | ID filter table element 1 |
| 0xE8 to 0x2D4 | ID filter table elements 2 to 125 |
| 0x2D8 | ID filter table element 126 |
| 0x2DC | ID filter table element 127 |
| | = Unimplemented or reserved |

Each ID filter table element occupies an entire 32-bit word and can be compounded by one, two, or four Identifier Acceptance Filters (IDAF) depending on the MCR[IDAM] field setting. The following figures show the IDAF indexation.

The following table shows the three different formats of the ID table elements. Note that all elements of the table must have the same format. See Rx FIFO for more information.

**Table 48-74. ID table structure**

| Format | 31 | 30 | 29 24 | 23 16 | 15 | 14 | 13 8 | 7 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| A | RTR | IDE | RXIDA (standard = 29–19, extended = 29–1) | | | | | | |
| B | RTR | IDE | RXIDB_0 (standard = 29–19, extended = 29–16) | | RTR | IDE | RXIDB_1 (standard = 13–3, extended = 13–0) | | |
| C | RXIDC_0 (std/ext = 31–24) | | RXIDC_1 (std/ext = 23–16) | | RXIDC_2 (std/ext = 15–8) | | RXIDC_3 (std/ext = 7–0) | | |
| | = Unimplemented or Reserved | | | | | | | | |

**RTR** — Remote Frame

This bit specifies if Remote Frames are accepted into the FIFO if they match the target ID.

1 = Remote Frames can be accepted and data frames are rejected

0 = Remote Frames are rejected and data frames can be accepted

**IDE** — Extended Frame

Specifies whether extended or standard frames are accepted into the FIFO if they match the target ID.

1 = Extended frames can be accepted and standard frames are rejected

0 = Extended frames are rejected and standard frames can be accepted

**RXIDA** — Rx Frame Identifier (Format A)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, only the 11 most significant bits (29 to 19) are used for frame identification. In the extended frame format, all bits are used.

**RXIDB_0, RXIDB_1** — Rx Frame Identifier (Format B)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (29 to 19 and 13 to 3) are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID.

**RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3** — Rx Frame Identifier (Format C)

Specifies an ID to be used as acceptance criteria for the FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID.

## 48.4 Functional description

The FlexCAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system is composed by a set of Message Buffers (MB) that store configuration and control data, time stamp, message ID and data (see Message buffer structure). The memory corresponding to the first 38 MBs can be configured to support a FIFO reception scheme with a powerful ID filtering mechanism, capable of checking incoming frames against a table of IDs (up to 128 extended IDs or 256 standard IDs or 512 8-bit ID slices), with individual mask register for up to 32 ID Filter Table elements. Simultaneous reception through FIFO and mailbox is supported. For mailbox reception, a matching algorithm makes it possible to store received frames only into MBs that have the same ID programmed on its ID field. A masking scheme makes it possible to match the ID programmed on the MB with a range of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the MB ordering.

Before proceeding with the functional description, an important concept must be explained. A Message Buffer is said to be "active" at a given time if it can participate in both the Matching and Arbitration processes. An Rx MB with a 0b0000 code is inactive (refer to Table 48-70). Similarly, a Tx MB with a 0b1000 or 0b1001 code is also inactive (refer to Table 48-71).

### 48.4.1 Transmit process

To transmit a CAN frame, the CPU must prepare a Message Buffer for transmission by executing the following procedure:

1. Check whether the respective interrupt bit is set and clear it.
2. If the MB is active (transmission pending), write the ABORT code (0b1001) to the CODE field of the Control and Status word to request an abortion of the transmission. Wait for the corresponding IFLAG to be asserted by polling the IFLAG register or by the interrupt request if enabled by the respective IMASK. Then read back the CODE field to check if the transmission was aborted or transmitted (see Transmission abort mechanism). If backwards compatibility is desired (MCR[AEN] bit is negated), just write the INACTIVE code (0b1000) to the CODE field to inactivate the MB but then the pending frame may be transmitted without notification (see Mailbox inactivation).
3. Write the ID word.
4. Write the data bytes.
5. Write the DLC, Control, and CODE fields of the Control and Status word to activate the MB.

When the MB is activated, it will participate into the arbitration process and eventually be transmitted according to its priority. At the end of the successful transmission, the value of the Free Running Timer is written into the Time Stamp field, the CODE field in the Control and Status word is updated, the CRC Register is updated, a status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit. The new CODE field after transmission depends on the code that was used to activate the MB (see Table 48-70 and Table 48-71 in Message buffer structure).

When the Abort feature is enabled (MCR[AEN] is asserted), after the Interrupt Flag is asserted for a Mailbox configured as transmit buffer, the Mailbox is blocked, therefore the CPU is not able to update it until the Interrupt Flag is negated by CPU. This means that the CPU must clear the corresponding IFLAG before starting to prepare this MB for a new transmission or reception.

### 48.4.2 Arbitration process

The arbitration process scans the Mailboxes searching the Tx one that holds the message to be sent in the next opportunity. This Mailbox is called the *arbitration winner*.

The scan starts from the lowest number Mailbox and runs toward the higher ones.

The arbitration process is triggered in the following events:

- From the CRC field of the CAN frame. The start point depends on the CTRL2[TASD] field value.
- During the Error Delimiter field of a CAN frame.
- During the Overload Delimiter field of a CAN frame.
- When the winner is inactivated and the CAN bus has still not reached the first bit of the Intermission field.
- When there is CPU write to the C/S word of a winner MB and the CAN bus has still not reached the first bit of the Intermission field.
- When CHI is in Idle state and the CPU writes to the C/S word of any MB.
- When FlexCAN exits Bus Off state.
- Upon leaving Freeze mode or Low Power mode.

If the arbitration process does not manage to evaluate all Mailboxes before the CAN bus has reached the first bit of the Intermission field the temporary arbitration winner is invalidated and the FlexCAN will not compete for the CAN bus in the next opportunity.

The arbitration process selects the winner among the active Tx Mailboxes at the end of the scan according to both CTRL1[LBUF] and MCR[LPRIOEN] bits settings.

#### 48.4.2.1 Lowest-number Mailbox first

If CTRL1[LBUF] bit is asserted the first (lowest number) active Tx Mailbox found is the arbitration winner. MCR[LPRIOEN] bit has no effect when CTRL1[LBUF] is asserted.

### 48.4.2.2 Highest-priority Mailbox first

If CTRL1[LBUF] bit is negated, then the arbitration process searches the active Tx Mailbox with the highest priority, which means that this Mailbox's frame would have a higher probability to win the arbitration on CAN bus when multiple external nodes compete for the bus at the same time.

The sequence of bits considered for this arbitration is called the *arbitration value* of the Mailbox. The highest-priority Tx Mailbox is the one that has the lowest arbitration value among all Tx Mailboxes.

If two or more Mailboxes have equivalent arbitration values, the Mailbox with the lowest number is the arbitration winner.

The composition of the arbitration value depends on MCR[LPRIOEN] bit setting.

#### 48.4.2.2.1 Local Priority disabled

If MCR[LPRIOEN] bit is negated the arbitration value is built in the exact sequence of bits as they would be transmitted in a CAN frame (see the following table) in such a way that the Local Priority is disabled.

**Table 48-75. Composition of the arbitration value when Local Priority is disabled**

| Format | Mailbox Arbitration Value (32 bits) | | | | |
|---|---|---|---|---|---|
| Standard (IDE = 0) | Standard ID (11 bits) | RTR (1 bit) | IDE (1 bit) | - (18 bits) | - (1 bit) |
| Extended (IDE = 1) | Extended ID[28:18] (11 bits) | SRR (1 bit) | IDE (1 bit) | Extended ID[17:0] (18 bits) | RTR (1 bit) |

#### 48.4.2.2.2 Local Priority enabled

If Local Priority is desired MCR[LPRIOEN] must be asserted. In this case the Mailbox PRIO field is included at the very left of the arbitration value (see the following table).

**Table 48-76. Composition of the arbitration value when Local Priority is enabled**

| Format | Mailbox Arbitration Value (35 bits) | | | | | |
|---|---|---|---|---|---|---|
| Standard (IDE = 0) | PRIO (3 bits) | Standard ID (11 bits) | RTR (1 bit) | IDE (1 bit) | - (18 bits) | - (1 bit) |
| Extended (IDE = 1) | PRIO (3 bits) | Extended ID[28:18] (11 bits) | SRR (1 bit) | IDE (1 bit) | Extended ID[17:0] (18 bits) | RTR (1 bit) |

As the PRIO field is the most significant part of the arbitration value Mailboxes with low PRIO values have higher priority than Mailboxes with high PRIO values regardless the rest of their arbitration values.

Note that the PRIO field is not part of the frame on the CAN bus. Its purpose is only to affect the internal arbitration process.

### 48.4.2.3 Arbitration process (continued)

After the arbitration winner is found, its content is copied to a hidden auxiliary MB called Tx Serial Message Buffer (Tx SMB), which has the same structure as a normal MB but is not user accessible. This operation is called move-out and after it is done, write access to the corresponding MB is blocked (if the AEN bit in MCR is asserted). The write access is released in the following events:

- After the MB is transmitted
- FlexCAN enters in Freeze mode or Bus Off
- FlexCAN loses the bus arbitration or there is an error during the transmission

At the first opportunity window on the CAN bus, the message on the Tx SMB is transmitted according to the CAN protocol rules. FlexCAN transmits up to eight data bytes, even if the Data Length Code (DLC) field value is greater than that.

Arbitration process can be triggered in the following situations:

- During Rx and Tx frames from CAN CRC field to end of frame. TASD value may be changed to optimize the arbitration start point.
- During CAN BusOff state from TX_ERR_CNT=124 to 128. TASD value may be changed to optimize the arbitration start point.
- During C/S write by CPU in BusIdle. First C/S write starts arbitration process and a second C/S write during this same arbitration restarts the process. If other C/S writes are performed, Tx arbitration process is pending. If there is no arbitration winner after arbitration process has finished, then TX arbitration machine begins a new arbitration process.
- 
  - If there is a pending arbitration and BusIdle state starts then an arbitration process is triggered. In this case the first and second C/S write in BusIdle will not restart the arbitration process. It is possible that there is not enough time to finish arbitration in WaitForBusIdle state and the next state is Idle. In this case the scan is not interrupted, and it is completed during BusIdle state. During this arbitration C/S write does not cause arbitration restart.
- Arbitration winner deactivation during a valid arbitration window.
- Upon Leave Freeze mode (first bit of the WaitForBusIdle state). If there is a re-synchronization during WaitForBusIdle arbitration process is restarted.

Arbitration process stops in the following situation:

- All Mailboxes were scanned
- A Tx active Mailbox is found in case of Lowest Buffer feature enabled
- Arbitration winner inactivation or abort during any arbitration process
- There was not enough time to finish Tx arbitration process (for instance, when a deactivation was performed near the end of frame). In this case arbitration process is pending.
- Error or Overload flag in the bus
- Low Power or Freeze mode request in Idle state

Arbitration is considered pending as described below:

- It was not possible to finish arbitration process in time
- C/S write during arbitration if write is performed in a MB which number is lower than the Tx arbitration pointer
- Any C/S write if there is no Tx Arbitration process in progress
- Rx Match has just updated a Rx Code to Tx Code
- Entering Busoff state

C/S write during arbitration has the following effect:

- If C/S write is performed in the arbitration winner, a new process is restarted immediately.
- If C/S write is performed in a MB whose number is higher than the Tx arbitration pointer, the ongoing arbitration process will scan this MB as normal.

### 48.4.3 Receive process

To be able to receive CAN frames into a Mailbox, the CPU must prepare it for reception by executing the following steps:

1. If the Mailbox is active (either Tx or Rx) inactivate the Mailbox (see Mailbox inactivation), preferably with a safe inactivation (see Transmission abort mechanism).
2. Write the ID word
3. Write the EMPTY code (0b0100) to the CODE field of the Control and Status word to activate the Mailbox.

After the MB is activated, it will be able to receive frames that match the programmed filter. At the end of a successful reception, the Mailbox is updated by the *move-in* process (see Move-in) as follows:

1. The received Data field (8 bytes at most) is stored.
2. The received Identifier field is stored.
3. The value of the Free Running Timer at the time of the second bit of frame's Identifier field is written into the Mailbox's Time Stamp field.
4. The received SRR, IDE, RTR, and DLC fields are stored.
5. The CODE field in the Control and Status word is updated (see Table 48-70 and Table 48-71 in Section Message buffer structure).
6. A status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit.

The recommended way for CPU servicing (read) the frame received in an Mailbox is using the following procedure:

1. Read the Control and Status word of that Mailbox.
2. Check if the BUSY bit is deasserted, indicating that the Mailbox is locked. Repeat step 1) while it is asserted. See Mailbox lock mechanism.
3. Read the contents of the Mailbox. Once Mailbox is locked now, its contents won't be modified by FlexCAN Move-in processes. See Move-in.
4. Acknowledge the proper flag at IFLAG registers.
5. Read the Free Running Timer. It is optional but recommended to unlock Mailbox as soon as possible and make it available for reception.

The CPU should synchronize to frame reception by the status flag bit for the specific Mailbox in one of the IFLAG Registers and not by the CODE field of that Mailbox. Polling the CODE field does not work because once a frame was received and the CPU services the Mailbox (by reading the C/S word followed by unlocking the Mailbox), the CODE field will not return to EMPTY. It will remain FULL, as explained in Table 48-70 . If the CPU tries to workaround this behavior by writing to the C/S word to force an EMPTY code after reading the Mailbox without a prior *safe inactivation*, a newly received frame matching the filter of that Mailbox may be lost.

**CAUTION**

In summary: *never do polling by reading directly the C/S word of the Mailboxes. Instead, read the IFLAG registers.*

Note that the received frame's Identifier field is always stored in the matching Mailbox, thus the contents of the ID field in an Mailbox may change if the match was due to masking. Note also that FlexCAN does receive frames transmitted by itself if there exists a matching Rx Mailbox, provided the MCR[SRXDIS] bit is not asserted. If the MCR[SRXDIS] bit is asserted, FlexCAN will not store frames transmitted by itself in any MB, even if it contains a matching MB, and no interrupt flag or interrupt signal will be generated due to the frame reception.

To be able to receive CAN frames through the Rx FIFO, the CPU must enable and configure the Rx FIFO during Freeze mode (see Rx FIFO). Upon receiving the Frames Available in Rx FIFO interrupt (see the description of the IFLAG[BUF5I] "Frames available in Rx FIFO" bit in the IMASK1 register), the CPU should service the received frame using the following procedure:

1. Read the Control and Status word (optional – needed only if a mask was used for IDE and RTR bits)
2. Read the ID field (optional – needed only if a mask was used)
3. Read the Data field
4. Read the RXFIR register (optional)
5. Clear the Frames Available in Rx FIFO interrupt by writing 1 to IFLAG[BUF5I] bit (mandatory – releases the MB and allows the CPU to read the next Rx FIFO entry)

### 48.4.4 Matching process

The matching process scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. If the FIFO is enabled, the priority of scanning can be selected between Mailboxes and FIFO filters. In any case, the matching starts from the lowest number Message Buffer toward the higher ones. If no match is found within the first structure then the other is scanned subsequently. In the event that the FIFO is full, the matching algorithm will always look for a matching MB outside the FIFO region.

As the frame is being received, it is stored in a hidden auxiliary MB called Rx Serial Message Buffer (Rx SMB).

The matching process start point depends on the following conditions:

- If the received frame is a remote frame, the start point is the CRC field of the frame

- If the received frame is a data frame with DLC field equal to zero, the start point is the CRC field of the frame
- If the received frame is a data frame with DLC field different than zero, the start point is the DATA field of the frame

If a matching ID is found in the FIFO table or in one of the Mailboxes, the contents of the SMB will be transferred to the FIFO or to the matched Mailbox by the move-in process. If any CAN protocol error is detected then no match results will be transferred to the FIFO or to the matched Mailbox at the end of reception.

The matching process scans all matching elements of both Rx FIFO (if enabled) and active Rx Mailboxes (CODE is EMPTY, FULL, OVERRUN or RANSWER) in search of a successful comparison with the matching elements of the Rx SMB that is receiving the frame on the CAN bus. The SMB has the same structure of a Mailbox. The reception structures (Rx FIFO or Mailboxes) associated with the matching elements that had a successful comparison are the *matched structures*. The *matching winner* is selected at the end of the scan among those matched structures and depends on conditions described ahead. See the following table.

**Table 48-77. Matching architecture**

| Structure | SMB[RTR] | CTRL2[RRS] | CTRL2[EACEN] | MB[IDE] | MB[RTR] | MB[ID[1]] | MB[CODE] |
|---|---|---|---|---|---|---|---|
| Mailbox | 0 | - | 0 | cmp[2] | no_cmp[3] | cmp_msk[4] | EMPTY or FULL or OVERRUN |
| Mailbox | 0 | - | 1 | cmp_msk | cmp_msk | cmp_msk | EMPTY or FULL or OVERRUN |
| Mailbox | 1 | 0 | - | cmp | no_cmp | cmp | RANSWER |
| Mailbox | 1 | 1 | 0 | cmp | no_cmp | cmp_msk | EMPTY or FULL or OVERRUN |
| Mailbox | 1 | 1 | 1 | cmp_msk | cmp_msk | cmp_msk | EMPTY or FULL or OVERRUN |
| FIFO[5] | - | - | - | cmp_msk | cmp_msk | cmp_msk | - |

1. For Mailbox structure, If SMB[IDE] is asserted, the ID is 29 bits (ID Standard + ID Extended). If SMB[IDE] is negated, the ID is only 11 bits (ID Standard). For FIFO structure, the ID depends on IDAM.
2. cmp: Compares the SMB contents with the MB contents regardless the masks.
3. no_cmp: The SMB contents are not compared with the MB contents
4. cmp_msk: Compares the SMB contents with MB contents taking into account the masks.
5. SMB[IDE] and SMB[RTR] are not taken into account when IDAM is type C.

A reception structure is *free-to-receive* when any of the following conditions is satisfied:

- The CODE field of the Mailbox is EMPTY

- The CODE field of the Mailbox is either FULL or OVERRUN and it has already been serviced (the C/S word was read by the CPU and unlocked as described in Mailbox lock mechanism)
- The CODE field of the Mailbox is either FULL or OVERRUN and an inactivation (see Mailbox inactivation) is performed
- The Rx FIFO is not full

The scan order for Mailboxes and Rx FIFO is from the matching element with lowest number to the higher ones.

The matching winner search for Mailboxes is affected by the MCR[IRMQ] bit. If it is negated the matching winner is the first matched Mailbox regardless if it is free-to-receive or not. If it is asserted, the matching winner is selected according to the priority below:

1. the first free-to-receive matched Mailbox;
2. the last non free-to-receive matched Mailbox.

It is possible to select the priority of scan between Mailboxes and Rx FIFO by the CTRL2[MRP] bit.

If the selected priority is Rx FIFO first:

- If the Rx FIFO is a matched structure and is free-to-receive then the Rx FIFO is the matching winner regardless of the scan for Mailboxes
- Otherwise (the Rx FIFO is not a matched structure or is not free-to-receive), then the matching winner is searched among Mailboxes as described above

If the selected priority is Mailboxes first:

- If a free-to-receive matched Mailbox is found, it is the matching winner regardless the scan for Rx FIFO
- If no matched Mailbox is found, then the matching winner is searched in the scan for the Rx FIFO
- If both conditions above are not satisfied and a non free-to-receive matched Mailbox is found then the matching winner determination is conditioned by the MCR[IRMQ] bit:
-
  - If MCR[IRMQ] bit is negated the matching winner is the first matched Mailbox
  - If MCR[IRMQ] bit is asserted the matching winner is the Rx FIFO if it is a free-to-receive matched structure, otherwise the matching winner is the last non free-to-receive matched Mailbox

See the following table for a summary of matching possibilities.

## Table 48-78. Matching possibilities and resulting reception structures

| RFEN | IRMQ | MRP | Matched in MB | Matched in FIFO | Reception structure | Description |
|---|---|---|---|---|---|---|
| **No FIFO, only MB, match is always MB first** | | | | | | |
| 0 | 0 | X[1] | None[2] | -[3] | None | Frame lost by no match |
| 0 | 0 | X | Free[4] | - | FirstMB | |
| 0 | 1 | X | None | - | None | Frame lost by no match |
| 0 | 1 | X | Free | - | FirstMb | |
| 0 | 1 | X | NotFree | - | LastMB | Overrun |
| **FIFO enabled, no match in FIFO is as if FIFO does not exist** | | | | | | |
| 1 | 0 | X | None | None[5] | None | Frame lost by no match |
| 1 | 0 | X | Free | None | FirstMB | |
| 1 | 1 | X | None | None | None | Frame lost by no match |
| 1 | 1 | X | Free | None | FirstMb | |
| 1 | 1 | X | NotFree | None | LastMB | Overrun |
| **FIFO enabled, Queue disabled** | | | | | | |
| 1 | 0 | 0 | X | NotFull[6] | FIFO | |
| 1 | 0 | 0 | None | Full[7] | None | Frame lost by FIFO full (FIFO Overflow) |
| 1 | 0 | 0 | Free | Full | FirstMB | |
| 1 | 0 | 0 | NotFree | Full | FirstMB | |
| 1 | 0 | 1 | None | NotFull | FIFO | |
| 1 | 0 | 1 | None | Full | None | Frame lost by FIFO full (FIFO Overflow) |
| 1 | 0 | 1 | Free | X | FirstMB | |
| 1 | 0 | 1 | NotFree | X | FirtsMb | Overrun |
| **FIFO enabled, Queue enabled** | | | | | | |
| 1 | 1 | 0 | X | NotFull | FIFO | |
| 1 | 1 | 0 | None | Full | None | Frame lost by FIFO full (FIFO Overflow) |
| 1 | 1 | 0 | Free | Full | FirstMB | |
| 1 | 1 | 0 | NotFree | Full | LastMb | Overrun |
| 1 | 1 | 1 | None | NotFull | FIFO | |
| 1 | 1 | 1 | Free | X | FirstMB | |
| 1 | 1 | 1 | NotFree | NotFull | FIFO | |
| 1 | 1 | 1 | NotFree | Full | LastMb | Overrun |

1. This is a don't care condition.
2. Matched in MB "None" means that the frame has not matched any MB (free-to-receive or non-free-to-receive).

3. This is a forbidden condition.
4. Matched in MB “Free” means that the frame matched at least one MB free-to-receive regardless of whether it has matched MBs non-free-to-receive.
5. Matched in FIFO “None” means that the frame has not matched any filter in FIFO. It is as if the FIFO didn’t exist (CTRL2[RFEN]=0).
6. Matched in FIFO “NotFull” means that the frame has matched a FIFO filter and has empty slots to receive it.
7. Matched in FIFO “Full” means that the frame has matched a FIFO filter but couldn’t store it because it has no empty slots to receive it.

If a non-safe Mailbox inactivation (see Mailbox inactivation) occurs during matching process and the Mailbox inactivated is the temporary matching winner then the temporary matching winner is invalidated. The matching elements scan is not stopped nor restarted, it continues normally. The consequence is that the current matching process works as if the matching elements compared before the inactivation did not exist, therefore a message may be lost.

Suppose, for example, that the FIFO is disabled, IRMQ is enabled and there are two MBs with the same ID, and FlexCAN starts receiving messages with that ID. Let us say that these MBs are the second and the fifth in the array. When the first message arrives, the matching algorithm will find the first match in MB number 2. The code of this MB is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm will find MB number 2 again, but it is not "free-to-receive", so it will keep looking and find MB number 5 and store the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are "free-to-receive", so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the CODE field of the MB to indicate OVERRUN.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full featured FIFO) to allow more time for the CPU to service the MBs. By programming more than one MB with the same ID, received messages will be queued into the MBs. The CPU can examine the Time Stamp field of the MBs to determine the order in which the messages arrived.

Matching to a range of IDs is possible by using ID Acceptance Masks. FlexCAN supports individual masking per MB. Refer to the description of the Rx Individual Mask Registers (RXIMRx). During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is "don't care". Please note that the Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed while the module is in Freeze mode; otherwise, they are blocked by hardware.

FlexCAN also supports an alternate masking scheme with only four mask registers (RGXMASK, RX14MASK, RX15MASK and RXFGMASK) for backwards compatibility with legacy applications. This alternate masking scheme is enabled when the IRMQ bit in the MCR Register is negated.

### 48.4.5 Move process

There are two types of move process: move-in and move-out.

#### 48.4.5.1 Move-in

The move-in process is the copy of a message received by an Rx SMB to a Rx Mailbox or FIFO that has matched it. If the move destination is the Rx FIFO, attributes of the message are also copied to the RXFIR FIFO. Each Rx SMB has its own move-in process, but only one is performed at a given time as described ahead. The move-in starts only when the message held by the Rx SMB has a corresponding matching winner (see Matching process) and all of the following conditions are true:

- The CAN bus has reached or let past either:
  - The second bit of Intermission field next to the frame that carried the message that is in the Rx SMB
  - The first bit of an overload frame next to the frame that carried the message that is in the Rx SMB
- There is no ongoing matching process
- The destination Mailbox is not locked by the CPU
- There is no ongoing move-in process from another Rx SMB. If more than one move-in processes are to be started at the same time both are performed and the newest substitutes the oldest.

The term *pending move-in* is used throughout the documentation and stands for a move-to-be that still does not satisfy all of the aforementioned conditions.

The move-in is cancelled and the Rx SMB is able to receive another message if any of the following conditions is satisfied:

- The destination Mailbox is inactivated after the CAN bus has reached the first bit of Intermission field next to the frame that carried the message and its matching process has finished
- There is a previous pending move-in to the same destination Mailbox
- The Rx SMB is receiving a frame transmitted by the FlexCAN itself and the self-reception is disabled (MCR[SRXDIS] bit is asserted)
- Any CAN protocol error is detected

Note that the pending move-in is not cancelled if the module enters Freeze or Low-Power mode. It only stays on hold waiting for exiting Freeze and Low-Power mode and to be unlocked. If an MB is unlocked during Freeze mode, the move-in happens immediately.

The move-in process is the execution by the FlexCAN of the following steps:

1. if the message is destined to the Rx FIFO, push IDHIT into the RXFIR FIFO;
2. reads the words DATA0-3 and DATA4-7 from the Rx SMB;
3. writes it in the words DATA0-3 and DATA4-7 of the Rx Mailbox;
4. reads the words Control/Status and ID from the Rx SMB;
5. writes it in the words Control/Status and ID of the Rx Mailbox, updating the CODE field.

The move-in process is not atomic, in such a way that it is immediately cancelled by the inactivation of the destination Mailbox (see Mailbox inactivation) and in this case the Mailbox may be left partially updated, thus incoherent. The exception is if the move-in destination is an Rx FIFO Message Buffer, then the process cannot be cancelled.

The BUSY Bit (least significant bit of the CODE field) of the destination Message Buffer is asserted while the move-in is being performed to alert the CPU that the Message Buffer content is temporarily incoherent.

#### 48.4.5.2 Move-out

The move-out process is the copy of the content from a Tx Mailbox to the Tx SMB when a message for transmission is available (see Section "Arbitration process"). The move-out occurs in the following conditions:

- The first bit of Intermission field
- During Bus Off state when TX Error Counter is in the 124 to 128 range
- During Bus Idle state
- During Wait For Bus Idle state

The move-out process is not atomic. Only the CPU has priority to access the memory concurrently out of Bus Idle state. In Bus Idle, the move-out has the lowest priority to the concurrent memory accesses.

### 48.4.6 Data coherence

In order to maintain data coherency and FlexCAN proper operation, the CPU must obey the rules described in Transmit process and Receive process.

### 48.4.6.1 Transmission abort mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform the CPU if the transmission was aborted or if the frame could not be aborted and was transmitted instead.

Two primary conditions must be fulfilled in order to abort a transmission:

- MCR[AEN] bit must be asserted
- The first CPU action must be the writing of abort code (0b1001) into the CODE field of the Control and Status word.

The active MBs configured as transmission must be aborted first and then they may be updated. If the abort code is written to a Mailbox that is currently being transmitted, or to a Mailbox that was already loaded into the SMB for transmission, the write operation is blocked and the MB is kept active, but the abort request is captured and kept pending until one of the following conditions are satisfied:

- The module loses the bus arbitration
- There is an error during the transmission
- The module is put into Freeze mode
- The module enters in BusOff state
- There is an overload frame

If none of the conditions above are reached, the MB is transmitted correctly, the interrupt flag is set in the IFLAG register, and an interrupt to the CPU is generated (if enabled). The abort request is automatically cleared when the interrupt flag is set. On the other hand, if one of the above conditions is reached, the frame is not transmitted; therefore, the abort code is written into the CODE field, the interrupt flag is set in the IFLAG, and an interrupt is (optionally) generated to the CPU.

If the CPU writes the abort code before the transmission begins internally, then the write operation is not blocked; therefore, the MB is updated and the interrupt flag is set. In this way the CPU just needs to read the abort code to make sure the active MB was *safely inactivated*. Although the AEN bit is asserted and the CPU wrote the abort code, in this case the MB is inactivated and not aborted, because the transmission did not start yet. One Mailbox is only aborted when the abort request is captured and kept pending until one of the previous conditions are satisfied.

The abort procedure can be summarized as follows:

- CPU checks the corresponding IFLAG and clears it, if asserted.

- CPU writes 0b1001 into the CODE field of the C/S word.
- CPU waits for the corresponding IFLAG indicating that the frame was either transmitted or aborted.
- CPU reads the CODE field to check if the frame was either transmitted (CODE=0b1000) or aborted (CODE=0b1001).
- It is necessary to clear the corresponding IFLAG in order to allow the MB to be reconfigured.

### 48.4.6.2 Mailbox inactivation

Inactivation is a mechanism provided to protect the Mailbox against updates by the FlexCAN internal processes, thus allowing the CPU to rely on Mailbox data coherence after having updated it, even in Normal mode.

Inactivation of transmission Mailboxes must be performed just when MCR[AEN] bit is deasserted.

If a Mailbox is inactivated, it participates in neither the arbitration process nor the matching process until it is reactivated. See Transmit process and Receive process for more detailed instruction on how to inactivate and reactivate a Mailbox.

To inactivate a Mailbox, the CPU must update its CODE field to INACTIVE (either 0b0000 or 0b1000).

Because the user is not able to synchronize the CODE field update with the FlexCAN internal processes, an inactivation can lead to undesirable results:

- A frame in the bus that matches the filtering of the inactivated Rx Mailbox may be lost without notice, even if there are other Mailboxes with the same filter
- A frame containing the message within the inactivated Tx Mailbox may be transmitted without notice

In order to eliminate such risk and perform a *safe inactivation* the CPU must use the following mechanism along with the inactivation itself:

- For Tx Mailboxes, the Transmission Abort (see Transmission abort mechanism)

The inactivation automatically unlocks the Mailbox (see Mailbox lock mechanism).

**NOTE**

Message Buffers that are part of the Rx FIFO cannot be inactivated. There is no write protection on the FIFO region by

FlexCAN. CPU must maintain data coherency in the FIFO region when RFEN is asserted.

### 48.4.6.3 Mailbox lock mechanism

Other than Mailbox inactivation, FlexCAN has another data coherence mechanism for the receive process. When the CPU reads the Control and Status word of an Rx MB with codes FULL or OVERRUN, FlexCAN assumes that the CPU wants to read the whole MB in an atomic operation, and therefore it sets an internal lock flag for that MB. The lock is released when the CPU reads the Free Running Timer (global unlock operation), or when it reads the Control and Status word of another MB regardless of its code. A CPU write into C/S word also unlocks the MB, but this procedure is not recommended for normal unlock use because it cancels a pending-move and potentially may lose a received message. The MB locking is done to prevent a new frame to be written into the MB while the CPU is reading it.

**NOTE**

The locking mechanism applies only to Rx MBs that are not part of FIFO and have a code different than INACTIVE (0b0000) or EMPTY[1] (0b0100). Also, Tx MBs can not be locked.

Suppose, for example, that the FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and FlexCAN has already received and stored messages into these two MBs. Suppose now that the CPU decides to read MB number 5 and at the same time another message with the same ID is arriving. When the CPU reads the Control and Status word of MB number 5, this MB is locked. The new message arrives and the matching algorithm finds out that there are no "free-to-receive" MBs, so it decides to override MB number 5. However, this MB is locked, so the new message can not be written there. It will remain in the SMB waiting for the MB to be unlocked, and only then will be written to the MB.

If the MB is not unlocked in time and yet another new message with the same ID arrives, then the new message overwrites the one on the SMB and there will be no indication of lost messages either in the CODE field of the MB or in the Error and Status Register.

While the message is being moved-in from the SMB to the MB, the BUSY bit on the CODE field is asserted. If the CPU reads the Control and Status word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.

1. In previous FlexCAN versions, reading the C/S word locked the MB even if it was EMPTY. This behavior is maintained when the IRMQ bit is negated.

**Note**

If the BUSY bit is asserted or if the MB is empty, then reading the Control and Status word does not lock the MB.

Inactivation takes precedence over locking. If the CPU inactivates a locked Rx MB, then its lock status is negated and the MB is marked as invalid for the current matching round. Any pending message on the SMB will not be transferred anymore to the MB. An MB is unlocked when the CPU reads the Free Running Timer Register (see Section "Free Running Timer Register (TIMER)"), or the C/S word of another MB.

Lock and unlock mechanisms have the same functionality in both Normal and Freeze modes.

An unlock during Normal or Freeze mode results in the move-in of the pending message. However, the move-in is postponed if an unlock occurs during a low power mode (see Modes of operation) and it will take place only when the module resumes to Normal or Freeze modes.

### 48.4.7 Rx FIFO

The Rx FIFO is receive-only and is enabled by asserting the MCR[RFEN] bit. The reset value of this bit is zero to maintain software backward compatibility with previous versions of the module that did not have the FIFO feature. The FIFO is 6-message deep. The memory region occupied by the FIFO structure (both Message Buffers and FIFO engine) is described in Rx FIFO structure. The CPU can read the received messages sequentially, in the order they were received, by repeatedly reading a Message Buffer structure at the output of the FIFO.

The IFLAG[BUF5I] (Frames available in Rx FIFO) is asserted when there is at least one frame available to be read from the FIFO. An interrupt is generated if it is enabled by the corresponding mask bit. Upon receiving the interrupt, the CPU can read the message (accessing the output of the FIFO as a Message Buffer) and the RXFIR register and then clear the interrupt. If there are more messages in the FIFO the act of clearing the interrupt updates the output of the FIFO with the next message and update the RXFIR with the attributes of that message, reissuing the interrupt to the CPU. Otherwise, the flag remains negated. The output of the FIFO is only valid whilst the IFLAG[BUF5I] is asserted.

The IFLAG[BUF6I] (Rx FIFO Warning) is asserted when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. The flag remains asserted until the CPU clears it.

The IFLAG[BUF7I] (Rx FIFO Overflow) is asserted when an incoming message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. The flag remains asserted until the CPU clears it.

Clearing one of those three flags does not affect the state of the other two.

An interrupt is generated if an IFLAG bit is asserted and the corresponding mask bit is asserted too.

A powerful filtering scheme is provided to accept only frames intended for the target application, reducing the interrupt servicing work load. The filtering criteria is specified by programming a table of up to 128 32-bit registers, according to CTRL2[RFFN] setting, that can be configured to one of the following formats (see also Rx FIFO structure):

- Format A: 128 IDAFs (extended or standard IDs including IDE and RTR)
- Format B: 256 IDAFs (standard IDs or extended 14-bit ID slices including IDE and RTR)
- Format C: 512 IDAFs (standard or extended 8-bit ID slices)

**Note**

A chosen format is applied to all entries of the filter table. It is not possible to mix formats within the table.

Every frame available in the FIFO has a corresponding IDHIT (Identifier Acceptance Filter Hit Indicator) that can be read by accessing the RXFIR register. The RXFIR[IDHIT] field refers to the message at the output of the FIFO and is valid while the IFLAG[BUF5I] flag is asserted. The RXFIR register must be read only before clearing the flag, which guarantees that the information refers to the correct frame within the FIFO.

Up to 32 elements of the filter table are individually affected by the Individual Mask Registers (RXIMRx), according to the setting of CTRL2[RFFN], allowing very powerful filtering criteria to be defined. If the IRMQ bit is negated, then the FIFO filter table is affected by RXFGMASK.

## 48.4.8 CAN protocol related features

This section describes the CAN protocol related features.

#### 48.4.8.1 Remote frames

Remote frame is a special kind of frame. The user can program a mailbox to be a Remote Request Frame by writing the mailbox as Transmit with the RTR bit set to '1'. After the remote request frame is transmitted successfully, the mailbox becomes a Receive Message Buffer, with the same ID as before.

When a remote request frame is received by FlexCAN, it can be treated in three ways, depending on Remote Request Storing (CTRL2[RRS]) and Rx FIFO Enable (MCR[RFEN]) bits:

- If RRS is negated the frame's ID is compared to the IDs of the Transmit Message Buffers with the CODE field 0b1010. If there is a matching ID, then this mailbox frame will be transmitted. Note that if the matching mailbox has the RTR bit set, then FlexCAN will transmit a remote frame as a response. The received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match. In the case that a remote request frame was received and matched a mailbox, this message buffer immediately enters the internal arbitration process, but is considered as normal Tx mailbox, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.
- If RRS is asserted the frame's ID is compared to the IDs of the receive mailboxes with the CODE field 0b0100, 0b0010 or 0b0110. If there is a matching ID, then this mailbox will store the remote frame in the same fashion of a data frame. No automatic remote response frame will be generated. The mask registers are used in the matching process.
- If RFEN is asserted FlexCAN will not generate an automatic response for remote request frames that match the FIFO filtering criteria. If the remote frame matches one of the target IDs, it will be stored in the FIFO and presented to the CPU. Note that for filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID). Remote Request Frames are considered as normal frames, and generate a FIFO overflow when a successful reception occurs and the FIFO is already full.

#### 48.4.8.2 Overload frames

FlexCAN does transmit overload frames due to detection of following conditions on CAN bus:

- Detection of a dominant bit in the first/second bit of Intermission
- Detection of a dominant bit at the 7th bit (last) of End of Frame field (Rx frames)
- Detection of a dominant bit at the 8th bit (last) of Error Frame Delimiter or Overload Frame Delimiter

### 48.4.8.3 Time stamp

The value of the Free Running Timer is sampled at the beginning of the Identifier field on the CAN bus, and is stored at the end of "move-in" in the TIME STAMP field, providing network behavior with respect to time.

The Free Running Timer can be reset upon a specific frame reception, enabling network time synchronization. See the TSYN description in the description of the Control 1 Register (CTRL1).

### 48.4.8.4 Protocol timing

The following figure shows the structure of the clock generation circuitry that feeds the CAN Protocol Engine (PE) submodule. The clock source bit CLKSRC in the CTRL1 Register defines whether the internal clock is connected to the output of a crystal oscillator (Oscillator Clock) or to the Peripheral Clock (generally from a PLL). In order to guarantee reliable operation, the clock source should be selected while the module is in Disable Mode (bit MDIS set in the Module Configuration Register).

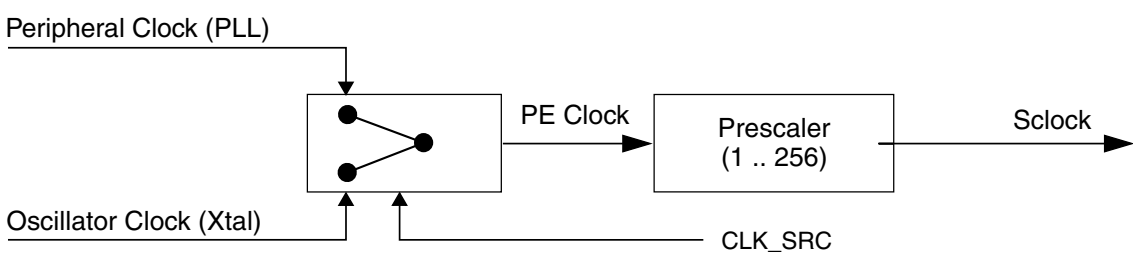



**Figure 48-66. CAN engine clocking scheme**

The crystal oscillator clock should be selected whenever a tight tolerance (up to 0.1%) is required in the CAN bus timing. The crystal oscillator clock has better jitter performance than PLL generated clocks.

The FlexCAN module supports a variety of means to setup bit timing parameters that are required by the CAN protocol. The Control Register has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW. See the description of the Control 1 Register (CTRL1).

The PRESDIV field controls a prescaler that generates the Serial Clock (Sclock), whose period defines the 'time quantum' used to compose the CAN waveform. A time quantum is the atomic unit of time handled by the CAN engine.

$$f_{Tq} = \frac{f_{CANCLK}}{(\text{Prescaler Value})}$$

A bit time is subdivided into three segments[2] (see Figure 48-67 and Table 48-79):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section
- Time Segment 1: This segment includes the Propagation Segment and the Phase Segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CTRL1 Register so that their sum (plus 2) is in the range of 4 to 16 time quanta
- Time Segment 2: This segment represents the Phase Segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CTRL1 Register (plus 1) to be 2 to 8 time quanta long

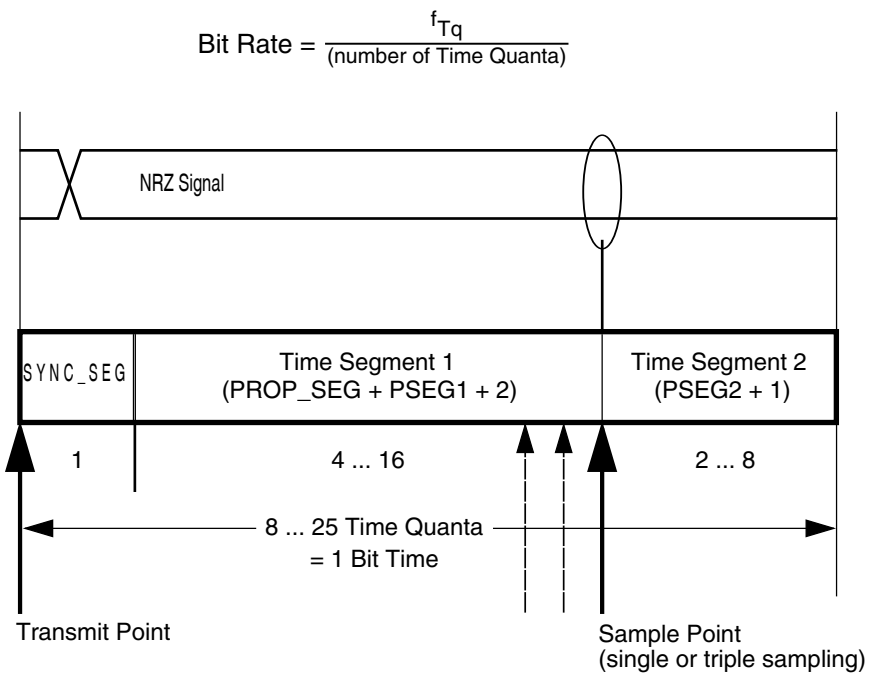



**Figure 48-67. Segments within the bit time**

2. For further explanation of the underlying concepts, see ISO/DIS 11519–1, Section 10.3. See also the CAN 2.0A/B protocol specification for bit timing.

Whenever CAN bit is used as a measure of duration (e.g. MCR[FRZACK] and MCR[LPMACK]), the number of peripheral clocks in one CAN bit can be calculated as:

$$NCCP = \frac{f_{SYS} \times [1 + (PSEG1 + 1) + (PSEG2 + 1) + (PROPSEG + 1)] \times (PRESDIV + 1)}{f_{CANCLK}}$$

where:

- NCCP is the number of peripheral clocks in one CAN bit;
- $f_{CANCLK}$ is the Protocol Engine (PE) Clock (see Figure "CAN Engine Clocking Scheme"), in Hz;
- $f_{SYS}$ is the frequency of operation of the system (CHI) clock, in Hz;
- PSEG1 is the value in CTRL1[PSEG1] field;
- PSEG2 is the value in CTRL1[PSEG2] field;
- PROPSEG is the value in CTRL1[PROPSEG] field;
- PRESDIV is the value in CTRL1[PRESDIV] field.

For example, 180 CAN bits = 180 x NCCP peripheral clock periods.

**Table 48-79. Time segment syntax**

| Syntax | Description |
|---|---|
| SYNC_SEG | System expects transitions to occur on the bus during this period. |
| Transmit Point | A node in transmit mode transfers a new value to the CAN bus at this point. |
| Sample Point | A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample. |

The following table gives an overview of the CAN compliant segment settings and the related parameter values.

**Table 48-80. CAN standard compliant bit time segment settings**

| Time segment 1 | Time segment 2 | Re-synchronization jump width |
|---|---|---|
| 5 .. 10 | 2 | 1 .. 2 |
| 4 .. 11 | 3 | 1 .. 3 |
| 5 .. 12 | 4 | 1 .. 4 |
| 6 .. 13 | 5 | 1 .. 4 |
| 7 .. 14 | 6 | 1 .. 4 |
| 8 .. 15 | 7 | 1 .. 4 |
| 9 .. 16 | 8 | 1 .. 4 |

**Note**

The user must ensure the bit time settings are in compliance with the CAN standard. For bit time calculations, use an IPT (Information Processing Time) of 2, which is the value implemented in the FlexCAN module.

### 48.4.8.5 Arbitration and matching timing

During normal reception and transmission of frames, the matching, arbitration, move-in and move-out processes are executed during certain time windows inside the CAN frame, as shown in the following figures.

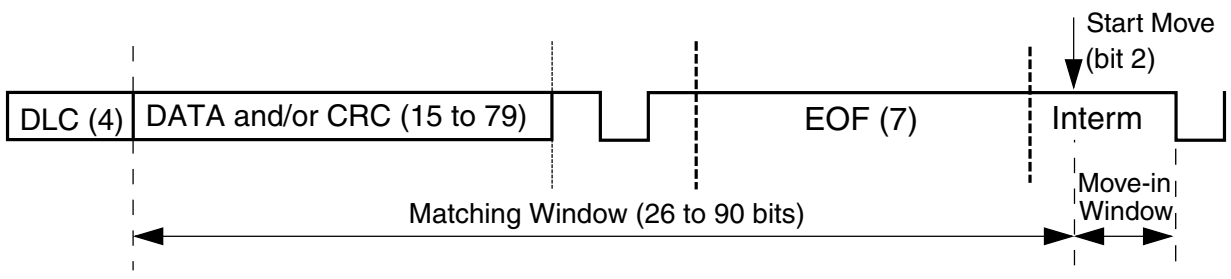



**Figure 48-68. Matching and move-in time windows**

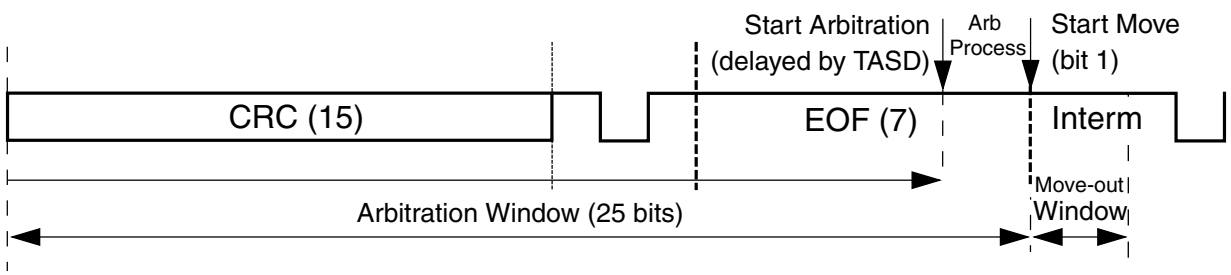



**Figure 48-69. Arbitration and move-out time windows**

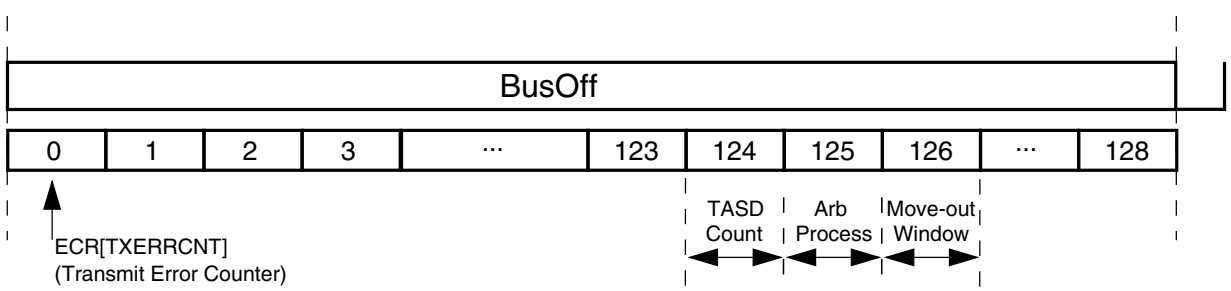



**Figure 48-70. Arbitration at the end of bus off and move-out time windows**

**NOTE**

The matching and arbitration timing shown in the preceding figures do not take into account the delay caused by the

concurrent memory access due to the CPU or other internal FlexCAN sub-blocks.

When doing matching and arbitration, FlexCAN needs to scan the whole Message Buffer memory during the available time slot. In order to have sufficient time to do that, the following requirements must be observed:

- A valid CAN bit timing must be programmed, as indicated in Table 48-80
- The peripheral clock frequency can not be smaller than the oscillator clock frequency, that is, the PLL can not be programmed to divide down the oscillator clock; see Clock domains and restrictions
- There must be a minimum ratio between the peripheral clock frequency and the CAN bit rate, as specified in the following table:

**Table 48-81. Minimum ratio between peripheral clock frequency and CAN bit rate**

| Number of Message Buffers | RFEN | Minimum number of peripheral clocks per CAN bit |
|---|---|---|
| 16 and 32 | 0 | 16 |
| 64 | 0 | 25 |
| 16 | 1 | 16 |
| 32 | 1 | 17 |
| 64 | 1 | 30 |

A direct consequence of the first requirement is that the minimum number of time quanta per CAN bit must be 8, therefore the oscillator clock frequency should be at least 8 times the CAN bit rate. The minimum frequency ratio specified in the preceding table can be achieved by choosing a high enough peripheral clock frequency when compared to the oscillator clock frequency, or by adjusting one or more of the bit timing parameters (PRESDIV, PROPSEG, PSEG1, PSEG2) contained in the Control 1 Register (CTRL1).

In case of synchronous operation (when the peripheral clock frequency is equal to the oscillator clock frequency), the number of peripheral clocks per CAN bit can be adjusted by selecting an adequate value for PRESDIV in order to meet the requirement in the preceding table. In case of asynchronous operation (the peripheral clock frequency greater than the oscillator clock frequency), the number of peripheral clocks per CAN bit can be adjusted by both PRESDIV and/or the frequency ratio.

As an example, taking the case of 64 MBs, if the oscillator and peripheral clock frequencies are equal and the CAN bit timing is programmed to have 8 time quanta per bit, then the prescaler factor (PRESDIV + 1) should be at least 2. For prescaler factor equal to one and CAN bit timing with 8 time quanta per bit, the ratio between peripheral and oscillator clock frequencies should be at least 2.

## 48.4.9 Clock domains and restrictions

The FlexCAN module has two clock domains asynchronous to each other:

- The Bus Domain feeds the Control Host Interface (CHI) submodule and is derived from the peripheral clock.
- The Oscillator Domain feeds the CAN Protocol Engine (PE) submodule and is derived directly from a crystal oscillator clock, so that very low jitter performance can be achieved on the CAN bus.

When CTRL1[CLKSRC] bit is set, synchronous operation occurs because both domains are connected to the peripheral clock (creating a 1:1 ratio between the peripheral and oscillator domain clocks).

When the two domains are connected to clocks with different frequencies and/or phases, there are restrictions on the frequency relationship between the two clock domains. In the case of asynchronous operation, the Bus Domain clock frequency must always be greater than the Oscillator Domain clock frequency.

**NOTE**

Asynchronous operation with a 1:1 ratio between peripheral and oscillator clocks is not allowed.

## 48.4.10 Modes of operation details

The FlexCAN module has functional modes and low-power modes. See Modes of operation for an introductory description of all the modes of operation. The following sub-sections contain functional details on Freeze mode and the low-power modes.

**CAUTION**

“Permanent Dominant” failure on CAN Bus line is not supported by FlexCAN. If a Low-Power request or Freeze mode request is done during a “Permanent Dominant”, the corresponding acknowledge can never be asserted.

### 48.4.10.1 Freeze mode

This mode is requested by the CPU through the assertion of the HALT bit in the MCR Register or when the MCU is put into Debug mode. In both cases it is also necessary that the FRZ bit is asserted in the MCR Register and the module is not in a low-power mode. The acknowledgement is obtained through the assertion by the FlexCAN of FRZ_ACK bit in the same register. The CPU must only consider the FlexCAN in Freeze mode when both request and acknowledgement conditions are satisfied.

When Freeze mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Intermission, Passive Error, Bus Off or Idle state
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.
- Ignores the Rx input pin and drives the Tx pin as recessive
- Stops the prescaler, thus halting all CAN protocol activities
- Grants write access to the Error Counters Register, which is read-only in other modes
- Sets the NOT_RDY and FRZ_ACK bits in MCR

After requesting Freeze mode, the user must wait for the FRZ_ACK bit to be asserted in MCR before executing any other action, otherwise FlexCAN may operate in an unpredictable way. In Freeze mode, all memory mapped registers are accessible, except for CTRL1[CLK_SRC] bit that can be read but cannot be written.

Exiting Freeze mode is done in one of the following ways:

- CPU negates the FRZ bit in the MCR Register
- The MCU is removed from Debug Mode and/or the HALT bit is negated

The FRZ_ACK bit is negated after the protocol engine recognizes the negation of the freeze request. When out of Freeze mode, FlexCAN tries to re-synchronize to the CAN bus by waiting for 11 consecutive recessive bits.

### 48.4.10.2 Module Disable mode

This low power mode is normally used to temporarily disable a complete FlexCAN block, with no power consumption. It is requested by the CPU through the assertion of the MDIS bit in the MCR Register and the acknowledgement is obtained through the assertion by the FlexCAN of the LPM_ACK bit in the same register. The CPU must only consider the FlexCAN in Disable mode when both request and acknowledgement conditions are satisfied.

If the module is disabled during Freeze mode, it requests to disable the clocks to the PE and CHI sub-modules, sets the LPM_ACK bit and negates the FRZ_ACK bit.

If the module is disabled during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and then checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive
- Shuts down the clocks to the PE and CHI sub-modules
- Sets the NOTRDY and LPMACK bits in MCR

The Bus Interface Unit continues to operate, enabling the CPU to access memory mapped registers, except the Rx Mailboxes Global Mask Registers, the Rx Buffer 14 Mask Register, the Rx Buffer 15 Mask Register, the Rx FIFO Global Mask Register. The Rx FIFO Information Register, the Message Buffers, the Rx Individual Mask Registers, and the reserved words within RAM may not be accessed when the module is in Disable Mode. Exiting from this mode is done by negating the MDIS bit by the CPU, which causes the FlexCAN to request to resume the clocks and negate the LPM_ACK bit after the CAN protocol engine recognizes the negation of disable mode requested by the CPU.

### 48.4.10.3 Stop mode

This is a system low-power mode in which all MCU clocks can be stopped for maximum power savings. The Stop mode is globally requested by the CPU and the acknowledgement is obtained through the assertion by the FlexCAN of a Stop Acknowledgement signal. The CPU must only consider the FlexCAN in Stop mode when both request and acknowledgement conditions are satisfied.

If FlexCAN receives the global Stop mode request during Freeze mode, it sets the LPMACK bit, negates the FRZACK bit and then sends the Stop Acknowledge signal to the CPU, in order to shut down the clocks globally.

If Stop mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive
- Sets the NOTRDY and LPMACK bits in MCR
- Sends a Stop Acknowledge signal to the CPU, so that it can shut down the clocks globally

Stop mode is exited when the CPU resumes the clocks and removes the Stop Mode request. This can be as a result of the Self Wake mechanism.

In the Self Wake mechanism, if the SLFWAK bit in MCR Register was set at the time FlexCAN entered Stop mode, then upon detection of a recessive to dominant transition on the CAN bus, FlexCAN sets the WAKINT bit in the ESR Register and, if enabled by the WAKMSK bit in MCR, generates a Wake Up interrupt to the CPU. Upon receiving the interrupt, the CPU should resume the clocks and remove the Stop mode request. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up. The following table details the effect of SLFWAK and WAKMSK upon wake-up from Stop mode. Note that wake-up from Stop mode only works when both bits are asserted.

After the CAN protocol engine recognizes the negation of the Stop mode request, the FlexCAN negates the LPMACK bit. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up.

**Table 48-82. Wake-up from Stop Mode**

| SLFWAK | WAKINT | WAKMSK | MCU clocks enabled | Wake-up interrupt generated |
|---|---|---|---|---|
| 0 | - | - | No | No |
| 0 | - | - | No | No |
| 1 | 0 | 0 | No | No |
| 1 | 0 | 1 | No | No |

*Table continues on the next page...*

**Table 48-82. Wake-up from Stop Mode (continued)**

| SLFWAK | WAKINT | WAKMSK | MCU clocks enabled | Wake-up interrupt generated |
|---|---|---|---|---|
| 1 | 1 | 0 | No | No |
| 1 | 1 | 1 | Yes | Yes |

## 48.4.11 Interrupts

The module has many interrupt sources: interrupts due to message buffers and interrupts due to the ORed interrupts from MBs, Bus Off, Error, Wake Up, Tx Warning, and Rx Warning.

Each one of the message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has assigned a flag bit in the IFLAG Registers. The bit is set when the corresponding buffer completes a successful transfer and is cleared when the CPU writes it to 1 (unless another interrupt is generated at the same time).

**Note**

It must be guaranteed that the CPU clears only the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

If the Rx FIFO is enabled (MCR[RFEN] = 1), the interrupts corresponding to MBs 0 to 7 have different meanings. Bit 7 of the IFLAG1 becomes the "FIFO Overflow" flag; bit 6 becomes the FIFO Warning flag, bit 5 becomes the "Frames Available in FIFO flag" and bits 4-0 are unused. See the description of the Interrupt Flags 1 Register (IFLAG1) for more information.

For a combined interrupt where multiple MB interrupt sources are OR'd together, the interrupt is generated when any of the associated MBs (or FIFO, if applicable) generates an interrupt. In this case, the CPU must read the IFLAG registers to determine which MB or FIFO source caused the interrupt.

The interrupt sources for Bus Off, Error, Wake Up, Tx Warning and Rx Warning generate interrupts like the MB interrupt sources, and can be read from both the Error and Status Register 1 and 2. The Bus Off, Error, Tx Warning, and Rx Warning interrupt mask bits are located in the Control 1 Register; the Wake-Up interrupt mask bit is located in the MCR.

### 48.4.12 Bus interface

The CPU access to FlexCAN registers are subject to the following rules:

- Unrestricted read and write access to supervisor registers (registers identified with S/U in Table "Module Memory Map" in Supervisor Mode or with S only) results in access error.
- Read and write access to implemented reserved address space results in access error.
- Write access to positions whose bits are all currently read-only results in access error. If at least one of the bits is not read-only then no access error is issued. Write permission to positions or some of their bits can change depending on the mode of operation or transitory state. Refer to register and bit descriptions for details.
- Read and write access to unimplemented address space results in access error.
- Read and write access to RAM located positions during Low Power Mode results in access error.
- If MAXMB is programmed with a value smaller than the available number of MBs, then the unused memory space can be used as general purpose RAM space. Note that reserved words within RAM cannot be used. As an example, suppose FlexCAN is configured with 16 MBs, RFFN is 0x0, and MAXMB is programmed with zero. The maximum number of MBs in this case becomes one. The RAM starts at 0x0080, and the space from 0x0080 to 0x008F is used by the one MB. The memory space from 0x0090 to 0x017F is available. The space between 0x0180 and 0x087F is reserved. The space from 0x0880 to 0x0883 is used by the one Individual Mask and the available memory in the Mask Registers space would be from 0x0884 to 0x08BF. From 0x08C0 through 0x09DF there are reserved words for internal use which cannot be used as general purpose RAM. As a general rule, free memory space for general purpose depends only on MAXMB.

## 48.5 Initialization/application information

This section provide instructions for initializing the FlexCAN module.

### 48.5.1 FlexCAN initialization sequence

The FlexCAN module may be reset as follows:

- MCU level hard reset, which resets all memory mapped registers asynchronously
- SOFTRST bit in MCR, which resets some of the memory mapped registers synchronously. See Table 48-2 to see what registers are affected by soft reset.
- MCU level soft reset, which has the same effect as the SOFTRST bit in MCR

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. The SOFTRST bit remains asserted while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset can not be applied while clocks are shut down in a low power mode. The low power mode should be exited and the clocks resumed before applying soft reset.

The clock source (CLKSRC bit) should be selected while the module is in Disable mode. After the clock source is selected and the module is enabled (MDIS bit negated), FlexCAN automatically goes to Freeze mode. In Freeze mode, FlexCAN is un-synchronized to the CAN bus, the HALT and FRZ bits in MCR Register are set, the internal state machines are disabled and the FRZACK and NOTRDY bits in the MCR Register are set. The Tx pin is in recessive state and FlexCAN does not initiate any transmission or reception of CAN frames. Note that the Message Buffers and the Rx Individual Mask Registers are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization it is required that FlexCAN is put into Freeze mode; see Freeze mode. The following is a generic initialization sequence applicable to the FlexCAN module:

- Initialize the Module Configuration Register
  - Enable the individual filtering per MB and reception queue features by setting the IRMQ bit
  - Enable the warning interrupts by setting the WRNEN bit

  - If required, disable frame self reception by setting the SRXDIS bit
  - Enable the Rx FIFO by setting the RFEN bit
  - Enable the abort mechanism by setting the AEN bit
  - Enable the local priority feature by setting the LPRIOEN bit
- Initialize the Control Register
  - Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW
  - Determine the bit rate by programming the PRESDIV field
  - Determine the internal arbitration mode (LBUF bit)
- Initialize the Message Buffers
  - The Control and Status word of all Message Buffers must be initialized
  - If Rx FIFO was enabled, the ID filter table must be initialized
  - Other entries in each Message Buffer should be initialized as required
- Initialize the Rx Individual Mask Registers
- Set required interrupt mask bits in the IMASK Registers (for all MB interrupts), in MCR Register for Wake-Up interrupt and in CTRL Register (for Bus Off and Error interrupts)
- Negate the HALT bit in MCR

Starting with the last event, FlexCAN attempts to synchronize to the CAN bus.

# Chapter 49
# Serial Peripheral Interface (SPI)

## 49.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The serial peripheral interface (SPI) module provides a synchronous serial bus for communication between an MCU and an external peripheral device.

### 49.1.1 Block Diagram

The block diagram of this module is as follows:

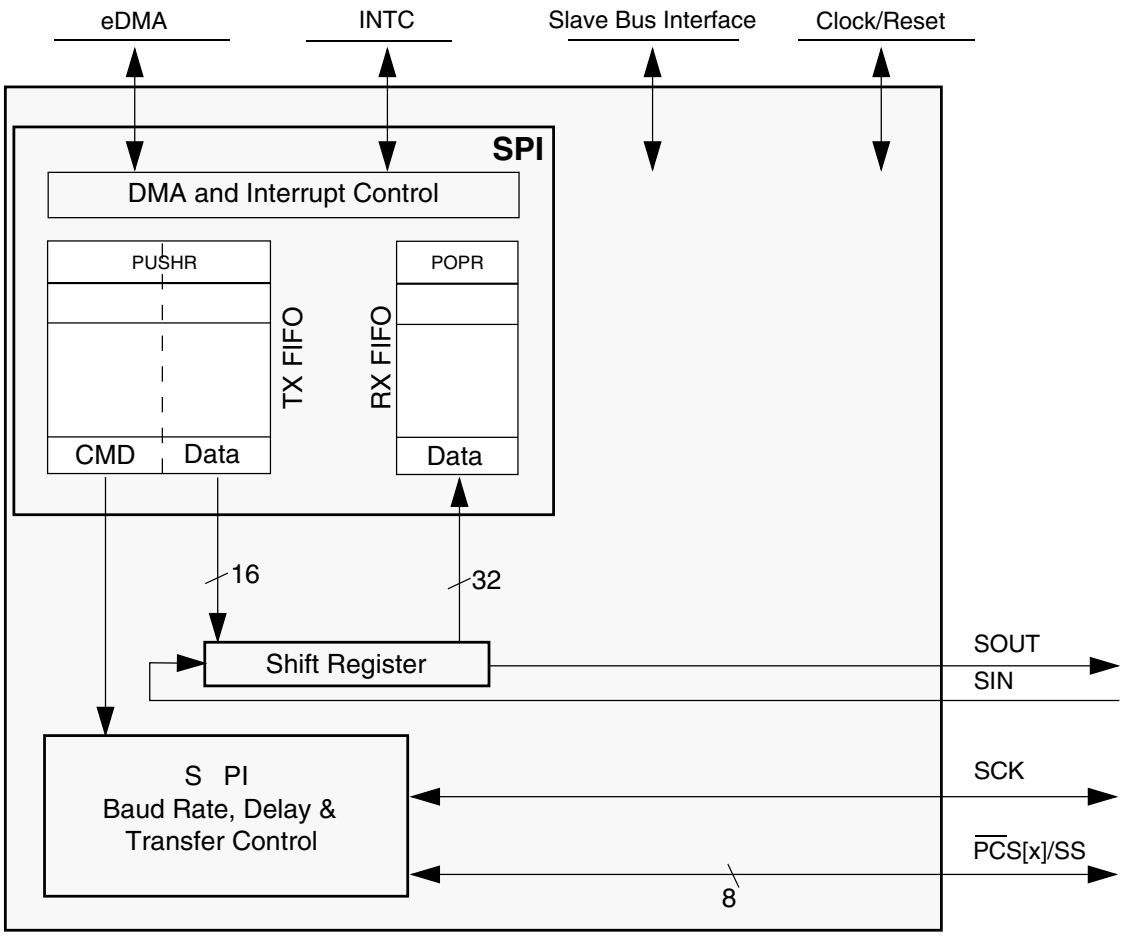



**Figure 49-1. SPI Block Diagram**

## 49.1.2 Features

The module supports the following features:

- Full-duplex, three-wire synchronous transfers
- Master and Slave modes
  - Data streaming operation in Slave mode with continuous slave selection
- Buffered transmit operation using the transmit first in first out (TX FIFO) with depth of four entries
- Buffered receive operation using the receive FIFO (RX FIFO) with depth of four entries
- TX and RX FIFOs can be disabled individually for low-latency updates to SPI queues

- Visibility into TX and RX FIFOs for ease of debugging
- Programmable transfer attributes on a per-frame basis:
  - two transfer attribute registers
  - Serial clock (SCK) with programmable polarity and phase
  - Various programmable delays
  - Programmable serial frame size of 4–16 bits, expandable by software control
    - SPI frames longer than 16 bits can be supported using the continuous selection format
  - Continuously held chip select capability
- 6 peripheral chip selects (PCSs), expandable to 64 with external demultiplexer
- Deglitching support for up to 32 PCS with external demultiplexer
- DMA support for adding entries to TX FIFO and removing entries from RX FIFO:
  - TX FIFO is not full (TFFF)
  - RX FIFO is not empty (RFDF)
- Interrupt conditions:
  - End of Queue reached (EOQF)
  - TX FIFO is not full (TFFF)
  - Transfer of current frame complete (TCF)
  - Attempt to transmit with an empty Transmit FIFO (TFUF)
  - RX FIFO is not empty (RFDF)
  - Frame received while Receive FIFO is full (RFOF)
- Global interrupt request line
- Modified SPI transfer formats for communication with slower peripheral devices
- Power-saving architectural features:
  - Support for Stop mode
  - Support for Doze mode

## 49.1.3 SPI Configuration

The SPI configuration allows the module to send and receive serial data. This configuration allows the module to operate as a basic SPI block with internal FIFOs supporting external queue operation. Transmitted data and received data reside in separate FIFOs. The host CPU or a DMA controller read the received data from the Receive FIFO and write transmit data to the Transmit FIFO.

For queued operations, the SPI queues can reside in system RAM, external to the module. Data transfers between the queues and the module FIFOs are accomplished by a DMA controller or host CPU. The following figure shows a system example with DMA, SPI, and external queues in system RAM.

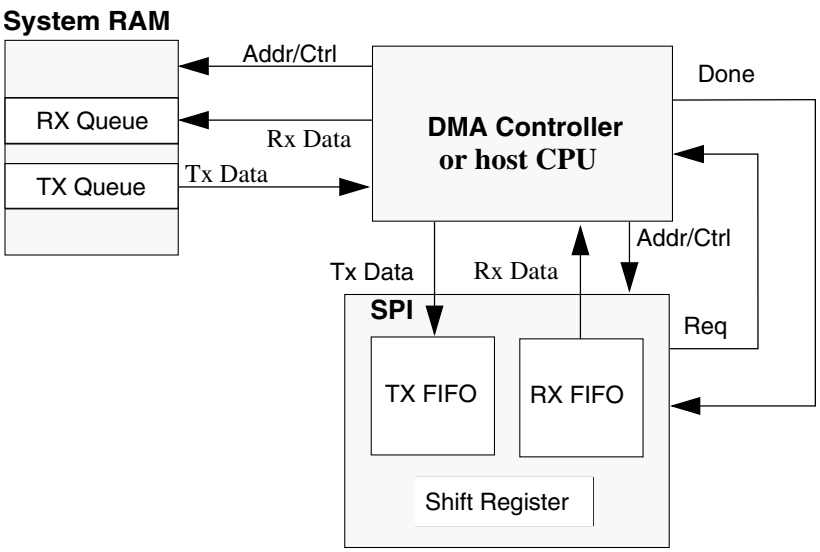



**Figure 49-2. SPI with queues and DMA**

## 49.1.4 Modes of Operation

The module supports the following modes of operation that can be divided into two categories:

- Module-specific modes:
  - Master mode
  - Slave mode
  - Module Disable mode
- MCU-specific modes:

- External Stop mode
- Debug mode

The module enters module-specific modes when the host writes a module register. The MCU-specific modes are controlled by signals external to the module. The MCU-specific modes are modes that an MCU may enter in parallel to the block-specific modes.

#### 49.1.4.1 Master Mode

Master mode allows the module to initiate and control serial communication. In this mode, the SCK signal, SOUT signal, and the PCS[*x*] signals are controlled by the module and configured as outputs.

#### 49.1.4.2 Slave Mode

Slave mode allows the module to communicate with SPI bus masters. In this mode, the module responds to externally controlled serial transfers. The SCK signal and the PCS[0]/$\overline{SS}$ signals are configured as inputs and driven by an SPI bus master.

#### 49.1.4.3 Module Disable Mode

The Module Disable mode can be used for MCU power management. The clock to the non-memory mapped logic in the module can be stopped while in the Module Disable mode.

#### 49.1.4.4 External Stop Mode

External Stop mode is used for MCU power management. The module supports the Peripheral Bus Stop mode mechanism. When a request is made to enter External Stop mode, it acknowledges the request and completes the transfer that is in progress. When the module reaches the frame boundary, it signals that the system clock to the module may be shut off.

#### 49.1.4.5 Debug Mode

Debug mode is used for system development and debugging. The MCR[FRZ] bit controls module behavior in the Debug mode:

- If the bit is set, the module stops all serial transfers, when the MCU is in debug mode.
- If the bit is cleared, the MCU debug mode has no effect on the module.

## 49.2 Module signal descriptions

This section provides description of the module signals.

The following table lists the signals that may connect off chip depending on device implementation.

**Table 49-1. Module signal descriptions**

| Signal | Master Mode | Slave Mode | Port Direction |
|---|---|---|---|
| PCS0/$\overline{SS}$ | Peripheral Chip Select 0 output | Slave Select input | I/O |
| PCS[3:1] | Peripheral Chip Select 1 – 3 | Unused | O |
| PCS4 | Peripheral Chip Select 4 | Unused | O |
| PCS5/ $\overline{PCSS}$ | Peripheral Chip Select 5 /Peripheral Chip Select Strobe | Unused | O |
| SIN | Serial Data In | Serial Data In | I |
| SOUT | Serial Data Out | Serial Data Out | O |
| SCK | Master mode: Serial Clock (output) | Serial Clock (input) | I/O |

### 49.2.1 PCS0/$\overline{SS}$ — Peripheral Chip Select/Slave Select

In Master mode, the PCS0 signal is an output that selects which slave device the current transmission is intended for.

In Slave mode, the active low $\overline{SS}$ signal is an input signal that allows an SPI master to select the module as the target for transmission.

### 49.2.2 PCS1 – PCS3 — Peripheral Chip Selects 1 – 3

PCS1 – PCS3 are output signals in Master mode.

In Slave mode, these signals are unused.

### 49.2.3 PCS4 — Peripheral Chip Select 4

In Master mode, PCS4 is an output signal.

In Slave mode, this signal is unused.

### 49.2.4 PCS5/$\overline{\text{PCSS}}$ — Peripheral Chip Select 5/Peripheral Chip Select Strobe

PCS5 is a Peripheral Chip Select output signal. When the module is in Master mode and the MCR[PCSSE] bit is cleared, this signal selects the slave device the current transfer is intended for.

When the module is in Master mode and the MCR[PCSSE] bit is set, the $\overline{\text{PCSS}}$ signal acts as a strobe to an external peripheral chip select demultiplexer, which decodes the PCS0 – PCS4 signals, preventing glitches on the demultiplexer outputs.

This signal is not used in Slave mode.

### 49.2.5 SIN — Serial Input

SIN is a serial data input signal.

### 49.2.6 SOUT — Serial Output

SOUT is a serial data output signal.

### 49.2.7 SCK — Serial Clock

SCK is a serial communication clock signal.

In Master mode, the module generates the SCK.

In Slave mode, SCK is an input from an external bus master.

## 49.3 Memory Map/Register Definition

Register accesses to memory addresses that are reserved or undefined result in a transfer error. Write access to the POPR also results in a transfer error.

## SPI memory map

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4002_C000 | Module Configuration Register (SPI0_MCR) | 32 | R/W | 0000_4001h | 49.3.1/1284 |
| 4002_C008 | Transfer Count Register (SPI0_TCR) | 32 | R/W | 0000_0000h | 49.3.2/1287 |
| 4002_C00C | Clock and Transfer Attributes Register (In Master Mode) (SPI0_CTAR0) | 32 | R/W | 7800_0000h | 49.3.3/1288 |
| 4002_C00C | Clock and Transfer Attributes Register (In Slave Mode) (SPI0_CTAR0_SLAVE) | 32 | R/W | 7800_0000h | 49.3.4/1292 |
| 4002_C010 | Clock and Transfer Attributes Register (In Master Mode) (SPI0_CTAR1) | 32 | R/W | 7800_0000h | 49.3.3/1288 |
| 4002_C02C | Status Register (SPI0_SR) | 32 | R/W | 0200_0000h | 49.3.5/1294 |
| 4002_C030 | DMA/Interrupt Request Select and Enable Register (SPI0_RSER) | 32 | R/W | 0000_0000h | 49.3.6/1297 |
| 4002_C034 | PUSH TX FIFO Register In Master Mode (SPI0_PUSHR) | 32 | R/W | 0000_0000h | 49.3.7/1299 |
| 4002_C034 | PUSH TX FIFO Register In Slave Mode (SPI0_PUSHR_SLAVE) | 32 | R/W | 0000_0000h | 49.3.8/1301 |
| 4002_C038 | POP RX FIFO Register (SPI0_POPR) | 32 | R | 0000_0000h | 49.3.9/1301 |
| 4002_C03C | Transmit FIFO Registers (SPI0_TXFR0) | 32 | R | 0000_0000h | 49.3.10/ 1302 |
| 4002_C040 | Transmit FIFO Registers (SPI0_TXFR1) | 32 | R | 0000_0000h | 49.3.10/ 1302 |
| 4002_C044 | Transmit FIFO Registers (SPI0_TXFR2) | 32 | R | 0000_0000h | 49.3.10/ 1302 |
| 4002_C048 | Transmit FIFO Registers (SPI0_TXFR3) | 32 | R | 0000_0000h | 49.3.10/ 1302 |
| 4002_C07C | Receive FIFO Registers (SPI0_RXFR0) | 32 | R | 0000_0000h | 49.3.11/ 1302 |
| 4002_C080 | Receive FIFO Registers (SPI0_RXFR1) | 32 | R | 0000_0000h | 49.3.11/ 1302 |
| 4002_C084 | Receive FIFO Registers (SPI0_RXFR2) | 32 | R | 0000_0000h | 49.3.11/ 1302 |
| 4002_C088 | Receive FIFO Registers (SPI0_RXFR3) | 32 | R | 0000_0000h | 49.3.11/ 1302 |
| 4002_D000 | Module Configuration Register (SPI1_MCR) | 32 | R/W | 0000_4001h | 49.3.1/1284 |
| 4002_D008 | Transfer Count Register (SPI1_TCR) | 32 | R/W | 0000_0000h | 49.3.2/1287 |
| 4002_D00C | Clock and Transfer Attributes Register (In Master Mode) (SPI1_CTAR0) | 32 | R/W | 7800_0000h | 49.3.3/1288 |
| 4002_D00C | Clock and Transfer Attributes Register (In Slave Mode) (SPI1_CTAR0_SLAVE) | 32 | R/W | 7800_0000h | 49.3.4/1292 |
| 4002_D010 | Clock and Transfer Attributes Register (In Master Mode) (SPI1_CTAR1) | 32 | R/W | 7800_0000h | 49.3.3/1288 |
| 4002_D02C | Status Register (SPI1_SR) | 32 | R/W | 0200_0000h | 49.3.5/1294 |
| 4002_D030 | DMA/Interrupt Request Select and Enable Register (SPI1_RSER) | 32 | R/W | 0000_0000h | 49.3.6/1297 |
| 4002_D034 | PUSH TX FIFO Register In Master Mode (SPI1_PUSHR) | 32 | R/W | 0000_0000h | 49.3.7/1299 |

*Table continues on the next page...*

## SPI memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4002_D034 | PUSH TX FIFO Register In Slave Mode (SPI1_PUSHR_SLAVE) | 32 | R/W | 0000_0000h | 49.3.8/1301 |
| 4002_D038 | POP RX FIFO Register (SPI1_POPR) | 32 | R | 0000_0000h | 49.3.9/1301 |
| 4002_D03C | Transmit FIFO Registers (SPI1_TXFR0) | 32 | R | 0000_0000h | 49.3.10/ 1302 |
| 4002_D040 | Transmit FIFO Registers (SPI1_TXFR1) | 32 | R | 0000_0000h | 49.3.10/ 1302 |
| 4002_D044 | Transmit FIFO Registers (SPI1_TXFR2) | 32 | R | 0000_0000h | 49.3.10/ 1302 |
| 4002_D048 | Transmit FIFO Registers (SPI1_TXFR3) | 32 | R | 0000_0000h | 49.3.10/ 1302 |
| 4002_D07C | Receive FIFO Registers (SPI1_RXFR0) | 32 | R | 0000_0000h | 49.3.11/ 1302 |
| 4002_D080 | Receive FIFO Registers (SPI1_RXFR1) | 32 | R | 0000_0000h | 49.3.11/ 1302 |
| 4002_D084 | Receive FIFO Registers (SPI1_RXFR2) | 32 | R | 0000_0000h | 49.3.11/ 1302 |
| 4002_D088 | Receive FIFO Registers (SPI1_RXFR3) | 32 | R | 0000_0000h | 49.3.11/ 1302 |
| 400A_C000 | Module Configuration Register (SPI2_MCR) | 32 | R/W | 0000_4001h | 49.3.1/1284 |
| 400A_C008 | Transfer Count Register (SPI2_TCR) | 32 | R/W | 0000_0000h | 49.3.2/1287 |
| 400A_C00C | Clock and Transfer Attributes Register (In Master Mode) (SPI2_CTAR0) | 32 | R/W | 7800_0000h | 49.3.3/1288 |
| 400A_C00C | Clock and Transfer Attributes Register (In Slave Mode) (SPI2_CTAR0_SLAVE) | 32 | R/W | 7800_0000h | 49.3.4/1292 |
| 400A_C010 | Clock and Transfer Attributes Register (In Master Mode) (SPI2_CTAR1) | 32 | R/W | 7800_0000h | 49.3.3/1288 |
| 400A_C02C | Status Register (SPI2_SR) | 32 | R/W | 0200_0000h | 49.3.5/1294 |
| 400A_C030 | DMA/Interrupt Request Select and Enable Register (SPI2_RSER) | 32 | R/W | 0000_0000h | 49.3.6/1297 |
| 400A_C034 | PUSH TX FIFO Register In Master Mode (SPI2_PUSHR) | 32 | R/W | 0000_0000h | 49.3.7/1299 |
| 400A_C034 | PUSH TX FIFO Register In Slave Mode (SPI2_PUSHR_SLAVE) | 32 | R/W | 0000_0000h | 49.3.8/1301 |
| 400A_C038 | POP RX FIFO Register (SPI2_POPR) | 32 | R | 0000_0000h | 49.3.9/1301 |
| 400A_C03C | Transmit FIFO Registers (SPI2_TXFR0) | 32 | R | 0000_0000h | 49.3.10/ 1302 |
| 400A_C040 | Transmit FIFO Registers (SPI2_TXFR1) | 32 | R | 0000_0000h | 49.3.10/ 1302 |
| 400A_C044 | Transmit FIFO Registers (SPI2_TXFR2) | 32 | R | 0000_0000h | 49.3.10/ 1302 |
| 400A_C048 | Transmit FIFO Registers (SPI2_TXFR3) | 32 | R | 0000_0000h | 49.3.10/ 1302 |

*Table continues on the next page...*

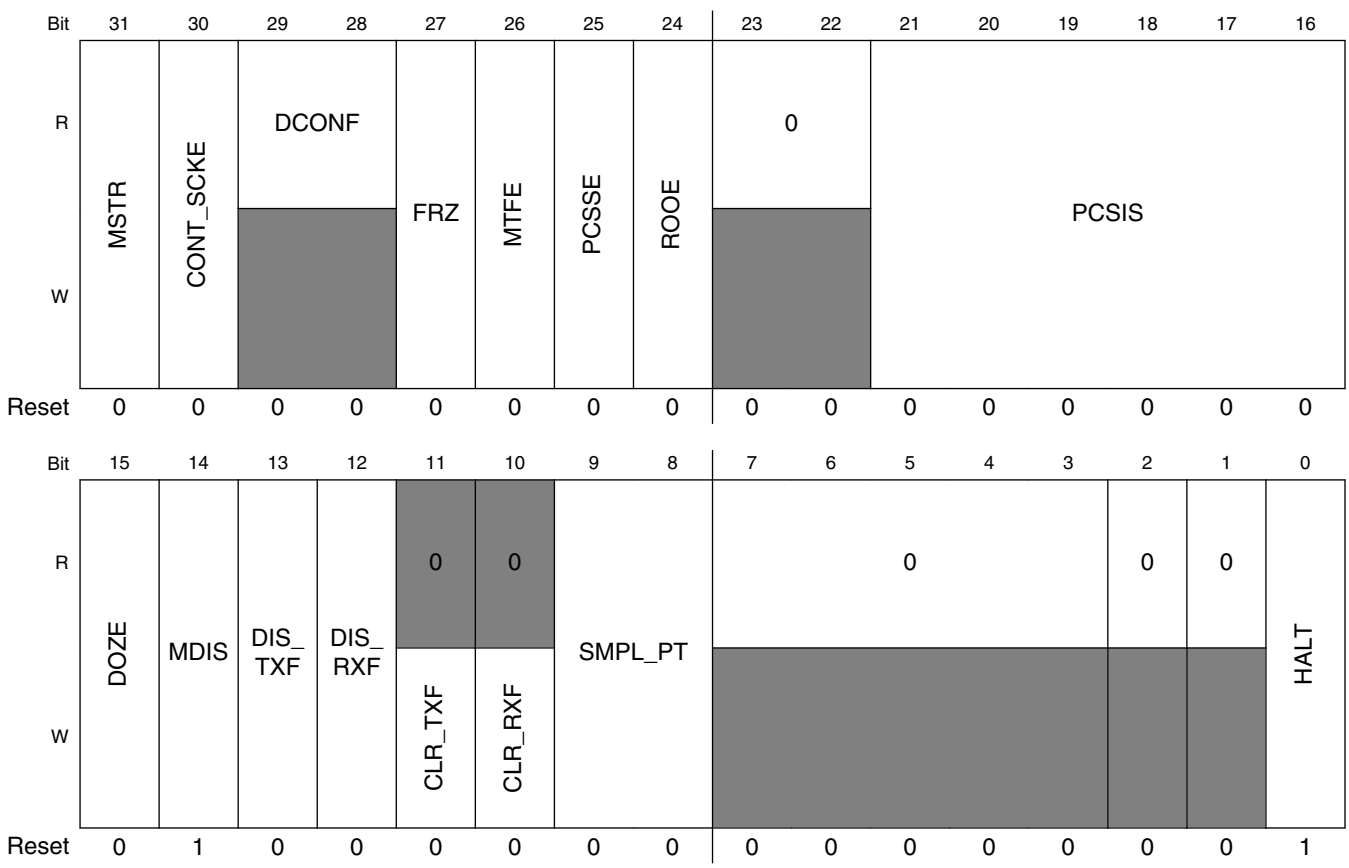
**SPI memory map (continued)**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 400A_C07C | Receive FIFO Registers (SPI2_RXFR0) | 32 | R | 0000_0000h | 49.3.11/ 1302 |
| 400A_C080 | Receive FIFO Registers (SPI2_RXFR1) | 32 | R | 0000_0000h | 49.3.11/ 1302 |
| 400A_C084 | Receive FIFO Registers (SPI2_RXFR2) | 32 | R | 0000_0000h | 49.3.11/ 1302 |
| 400A_C088 | Receive FIFO Registers (SPI2_RXFR3) | 32 | R | 0000_0000h | 49.3.11/ 1302 |

## 49.3.1 Module Configuration Register (SPIx_MCR)

Contains bits to configure various attributes associated with the module operations. The HALT and MDIS bits can be changed at any time, but the effect takes place only on the next frame boundary. Only the HALT and MDIS bits in the MCR can be changed, while the module is in the Running state.

Address: Base address + 0h offset



## SPIx_MCR field descriptions

| Field | Description |
|---|---|
| 31<br>MSTR | Master/Slave Mode Select<br><br>Configures the module for either Master mode or Slave mode.<br><br>0 The module is in Slave mode.<br>1 The module is in Master mode. |
| 30<br>CONT_SCKE | Continuous SCK Enable<br><br>Enables the Serial Communication Clock (SCK) to run continuously.<br><br>0 Continuous SCK disabled.<br>1 Continuous SCK enabled. |
| 29–28<br>DCONF | SPI Configuration.<br><br>Selects among the different configurations of the module.<br><br>00 SPI<br>01 Reserved<br>10 Reserved<br>11 Reserved |
| 27<br>FRZ | Freeze<br><br>Enables the SPI transfers to be stopped on the next frame boundary when the device enters Debug mode.<br><br>0 Do not halt serial transfers in Debug mode.<br>1 Halt serial transfers in Debug mode. |
| 26<br>MTFE | Modified Timing Format Enable<br><br>Enables a modified transfer format to be used.<br><br>0 Modified SPI transfer format disabled.<br>1 Modified SPI transfer format enabled. |
| 25<br>PCSSE | Peripheral Chip Select Strobe Enable<br><br>Enables the PCS5/ $\overline{\text{PCSS}}$ to operate as a PCS Strobe output signal.<br><br>0 PCS5/ $\overline{\text{PCSS}}$ is used as the Peripheral Chip Select[5] signal.<br>1 PCS5/ $\overline{\text{PCSS}}$ is used as an active-low PCS Strobe signal. |
| 24<br>ROOE | Receive FIFO Overflow Overwrite Enable<br><br>In the RX FIFO overflow condition, configures the module to ignore the incoming serial data or overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer, generating the overflow, is ignored or shifted into the shift register.<br><br>0 Incoming data is ignored.<br>1 Incoming data is shifted into the shift register. |
| 23–22<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 21–16<br>PCSIS | Peripheral Chip Select x Inactive State<br><br>Determines the inactive state of PCSx. |

*Table continues on the next page...*

## SPIx_MCR field descriptions (continued)

| Field | Description |
|---|---|
| | 0 The inactive state of PCSx is low.<br>1 The inactive state of PCSx is high. |
| 15<br>DOZE | Doze Enable<br><br>Provides support for an externally controlled Doze mode power-saving mechanism.<br><br>0 Doze mode has no effect on the module.<br>1 Doze mode disables the module. |
| 14<br>MDIS | Module Disable<br><br>Allows the clock to be stopped to the non-memory mapped logic in the module effectively putting it in a software-controlled power-saving state. The reset value of the MDIS bit is parameterized, with a default reset value of 0. When SPI is used in Slave Mode, it is recommended to leave this bit set to '0', since a slave doesn't have control over master transactions.<br><br>0 Enables the module clocks.<br>1 Allows external logic to disable the module clocks. |
| 13<br>DIS_TXF | Disable Transmit FIFO<br><br>When the TX FIFO is disabled, the transmit part of the module operates as a simplified double-buffered SPI. This bit can be written only when the MDIS bit is cleared.<br><br>0 TX FIFO is enabled.<br>1 TX FIFO is disabled. |
| 12<br>DIS_RXF | Disable Receive FIFO<br><br>When the RX FIFO is disabled, the receive part of the module operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared.<br><br>0 RX FIFO is enabled.<br>1 RX FIFO is disabled. |
| 11<br>CLR_TXF | Clear TX FIFO<br><br>Flushes the TX FIFO. Writing a 1 to CLR_TXF clears the TX FIFO Counter. The CLR_TXF bit is always read as zero.<br><br>0 Do not clear the TX FIFO counter.<br>1 Clear the TX FIFO counter. |
| 10<br>CLR_RXF | Flushes the RX FIFO. Writing a 1 to CLR_RXF clears the RX Counter. The CLR_RXF bit is always read as zero.<br><br>0 Do not clear the RX FIFO counter.<br>1 Clear the RX FIFO counter. |
| 9–8<br>SMPL_PT | Sample Point<br><br>Controls when the module master samples SIN in Modified Transfer Format. This field is valid only when CPHA bit in CTARn[CPHA] is 0.<br><br>00 0 system clocks between SCK edge and SIN sample<br>01 1 system clock between SCK edge and SIN sample<br>10 2 system clocks between SCK edge and SIN sample<br>11 Reserved |

*Table continues on the next page...*

**SPIx_MCR field descriptions (continued)**

| Field | Description |
|---|---|
| 7–3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>HALT | Halt<br><br>The HALT bit starts and stops SPI transfers. See Start and Stop of Module transfers<br><br>0 Start transfers.<br>1 Stop transfers. |

## 49.3.2 Transfer Count Register (SPIx_TCR)

TCR contains a counter that indicates the number of SPI transfers made. The transfer counter is intended to assist in queue management. Do not write the TCR when the module is in the Running state.

Address: Base address + 8h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SPI_TCNT | | | | | | | | | | | | | | | | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SPIx_TCR field descriptions**

| Field | Description |
|---|---|
| 31–16<br>SPI_TCNT | SPI Transfer Counter<br><br>Counts the number of SPI transfers the module makes. The SPI_TCNT field increments every time the last bit of an SPI frame is transmitted. A value written to SPI_TCNT presets the counter to that value. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI command. The Transfer Counter wraps around; incrementing the counter past 65535 resets the counter to zero. |
| 15–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

### 49.3.3 Clock and Transfer Attributes Register (In Master Mode) (SPIx_CTAR*n*)

CTAR registers are used to define different transfer attributes. Do not write to the CTAR registers while the module is in the Running state.

In Master mode, the CTAR registers define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays. In slave mode, a subset of the bitfields in CTAR0 are used to set the slave transfer attributes.

When the module is configured as an SPI master, the CTAS field in the command portion of the TX FIFO entry selects which of the CTAR registers is used. When the module is configured as an SPI bus slave, the CTAR0 register is used.

Address: Base address + Ch offset + (4d × i), where i=0d to 1d

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DBR | FMSZ | | | | CPOL | CPHA | LSBFE | PCSSCK | | PASC | | PDT | | PBR | |
| Reset | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CSSCK | | | | ASC | | | | DT | | | | BR | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SPIx_CTAR*n* field descriptions**

| Field | Description |
|---|---|
| 31<br>DBR | Double Baud Rate<br><br>Doubles the effective baud rate of the Serial Communications Clock (SCK). This field is used only in master mode. It effectively halves the Baud Rate division ratio, supporting faster frequencies, and odd division ratios for the Serial Communications Clock (SCK). When the DBR bit is set, the duty cycle of the Serial Communications Clock (SCK) depends on the value in the Baud Rate Prescaler and the Clock Phase bit as listed in the following table. See the BR field description for details on how to compute the baud rate. |

**Table 49-40. SPI SCK Duty Cycle**

| DBR | CPHA | PBR | SCK Duty Cycle |
|---|---|---|---|
| 0 | any | any | 50/50 |
| 1 | 0 | 00 | 50/50 |
| 1 | 0 | 01 | 33/66 |
| 1 | 0 | 10 | 40/60 |

*Table continues on the next page...*

## SPIx_CTAR*n* field descriptions (continued)

| Field | Description |
|---|---|
| | **Table 49-40. SPI SCK Duty Cycle (continued)**<br><br>DBR / CPHA / PBR / SCK Duty Cycle:<br>1 / 0 / 11 / 43/57<br>1 / 1 / 00 / 50/50<br>1 / 1 / 01 / 66/33<br>1 / 1 / 10 / 60/40<br>1 / 1 / 11 / 57/43<br><br>0 The baud rate is computed normally with a 50/50 duty cycle.<br>1 The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler. |
| 30–27<br>FMSZ | Frame Size<br><br>The number of bits transferred per frame is equal to the FMSZ field value plus 1. Regardless of the transmission mode, the minimum valid frame size value is 4. |
| 26<br>CPOL | Clock Polarity<br><br>Selects the inactive state of the Serial Communications Clock (SCK). This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock polarities. When the Continuous Selection Format is selected, switching between clock polarities without stopping the module can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge.<br><br>0 The inactive state value of SCK is low.<br>1 The inactive state value of SCK is high. |
| 25<br>CPHA | Clock Phase<br><br>Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode , the bit value is ignored and the transfers are done as if the CPHA bit is set to 1.<br><br>0 Data is captured on the leading edge of SCK and changed on the following edge.<br>1 Data is changed on the leading edge of SCK and captured on the following edge. |
| 24<br>LSBFE | LSB First<br><br>Specifies whether the LSB or MSB of the frame is transferred first.<br><br>0 Data is transferred MSB first.<br>1 Data is transferred LSB first. |
| 23–22<br>PCSSCK | PCS to SCK Delay Prescaler<br><br>Selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. See the CSSCK field description for information on how to compute the PCS to SCK Delay. Refer PCS to SCK Delay ($t_{CSC}$ ) for more details.<br><br>00 PCS to SCK Prescaler value is 1.<br>01 PCS to SCK Prescaler value is 3. |

*Table continues on the next page...*

## SPIx_CTAR*n* field descriptions (continued)

| Field | Description |
|---|---|
| | 10 PCS to SCK Prescaler value is 5.<br>11 PCS to SCK Prescaler value is 7. |
| 21–20<br>PASC | After SCK Delay Prescaler<br><br>Selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. See the ASC field description for information on how to compute the After SCK Delay. Refer After SCK Delay ($t_{ASC}$ ) for more details.<br><br>00 Delay after Transfer Prescaler value is 1.<br>01 Delay after Transfer Prescaler value is 3.<br>10 Delay after Transfer Prescaler value is 5.<br>11 Delay after Transfer Prescaler value is 7. |
| 19–18<br>PDT | Delay after Transfer Prescaler<br><br>Selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The PDT field is only used in master mode. See the DT field description for details on how to compute the Delay after Transfer. Refer Delay after Transfer ($t_{DT}$ ) for more details.<br><br>00 Delay after Transfer Prescaler value is 1.<br>01 Delay after Transfer Prescaler value is 3.<br>10 Delay after Transfer Prescaler value is 5.<br>11 Delay after Transfer Prescaler value is 7. |
| 17–16<br>PBR | Baud Rate Prescaler<br><br>Selects the prescaler value for the baud rate. This field is used only in master mode. The baud rate is the frequency of the SCK. The system clock is divided by the prescaler value before the baud rate selection takes place. See the BR field description for details on how to compute the baud rate.<br><br>00 Baud Rate Prescaler value is 2.<br>01 Baud Rate Prescaler value is 3.<br>10 Baud Rate Prescaler value is 5.<br>11 Baud Rate Prescaler value is 7. |
| 15–12<br>CSSCK | PCS to SCK Delay Scaler<br><br>Selects the scaler value for the PCS to SCK delay. This field is used only in master mode. The PCS to SCK Delay is the delay between the assertion of PCS and the first edge of the SCK. The delay is a multiple of the system clock period, and it is computed according to the following equation:<br><br>$t_{CSC} = (1/f_{SYS}) \times PCSSCK \times CSSCK$.<br><br>The following table lists the delay scaler values. |

### Table 49-39. Delay Scaler Encoding

| Field Value | Delay Scaler Value |
|---|---|
| 0000 | 2 |
| 0001 | 4 |
| 0010 | 8 |
| 0011 | 16 |
| 0100 | 32 |

*Table continues on the next page...*

## SPIx_CTAR*n* field descriptions (continued)

| Field | Description |
|---|---|
| | **Table 49-39. Delay Scaler Encoding (continued)**<br><br>\| Field Value \| Delay Scaler Value \|<br>\| --- \| --- \|<br>\| 0101 \| 64 \|<br>\| 0110 \| 128 \|<br>\| 0111 \| 256 \|<br>\| 1000 \| 512 \|<br>\| 1001 \| 1024 \|<br>\| 1010 \| 2048 \|<br>\| 1011 \| 4096 \|<br>\| 1100 \| 8192 \|<br>\| 1101 \| 16384 \|<br>\| 1110 \| 32768 \|<br>\| 1111 \| 65536 \|<br><br>Refer PCS to SCK Delay ($t_{CSC}$ ) for more details. |
| 11–8<br>ASC | After SCK Delay Scaler<br><br>Selects the scaler value for the After SCK Delay. This field is used only in master mode. The After SCK Delay is the delay between the last edge of SCK and the negation of PCS. The delay is a multiple of the system clock period, and it is computed according to the following equation:<br><br>$t_{ASC} = (1/f_{SYS}) \times PASC \times ASC$<br><br>See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. Refer After SCK Delay ($t_{ASC}$ ) for more details. |
| 7–4<br>DT | Delay After Transfer Scaler<br><br>Selects the Delay after Transfer Scaler. This field is used only in master mode. The Delay after Transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame.<br><br>In the Continuous Serial Communications Clock operation, the DT value is fixed to one SCK clock period, The Delay after Transfer is a multiple of the system clock period, and it is computed according to the following equation:<br><br>$t_{DT} = (1/f_{SYS}) \times PDT \times DT$<br><br>See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. |
| 3–0<br>BR | Baud Rate Scaler<br><br>Selects the scaler value for the baud rate. This field is used only in master mode. The prescaled system clock is divided by the Baud Rate Scaler to generate the frequency of the SCK. The baud rate is computed according to the following equation:<br><br>SCK baud rate = $(f_{SYS}/PBR) \times [(1+DBR)/BR]$<br><br>The following table lists the baud rate scaler values. |

*Table continues on the next page...*

SPIx_CTAR*n* field descriptions (continued)

| Field | Description |
|---|---|

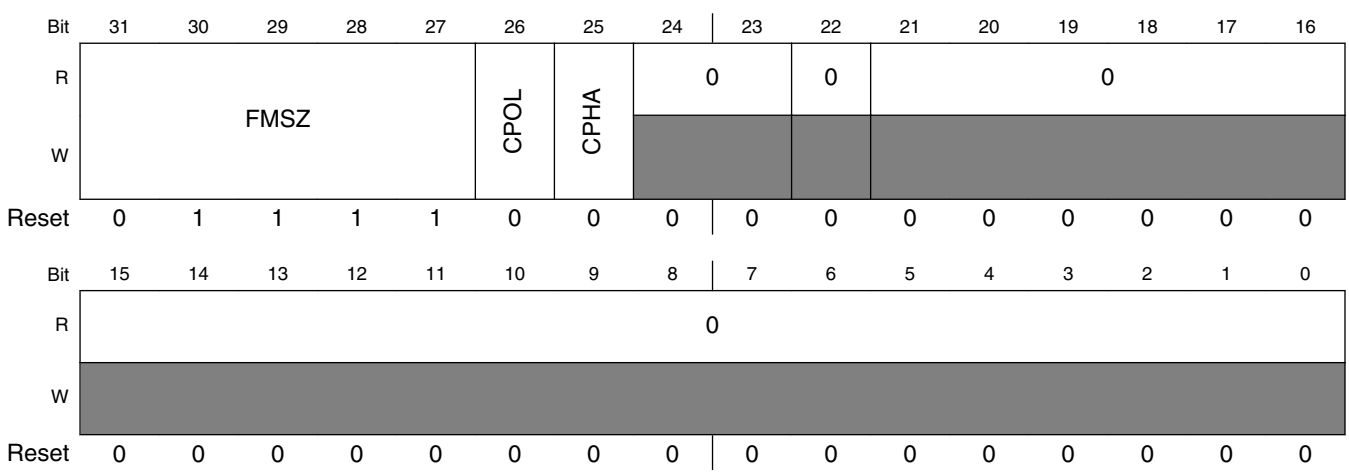
Table 49-38. Baud Rate Scaler

| CTARn[BR] | Baud Rate Scaler Value |
|---|---|
| 0000 | 2 |
| 0001 | 4 |
| 0010 | 6 |
| 0011 | 8 |
| 0100 | 16 |
| 0101 | 32 |
| 0110 | 64 |
| 0111 | 128 |
| 1000 | 256 |
| 1001 | 512 |
| 1010 | 1024 |
| 1011 | 2048 |
| 1100 | 4096 |
| 1101 | 8192 |
| 1110 | 16384 |
| 1111 | 32768 |

## 49.3.4 Clock and Transfer Attributes Register (In Slave Mode) (SPIx_CTAR*n*_SLAVE)

When the module is configured as an SPI bus slave, the CTAR0 register is used.

Address: Base address + Ch offset + (0d × i), where i=0d to 0d



## SPIx_CTAR*n*_SLAVE field descriptions

| Field | Description |
|---|---|
| 31–27<br>FMSZ | Frame Size<br><br>The number of bits transfered per frame is equal to the FMSZ field value plus 1. Note that the minimum valid value of frame size is 4. |
| 26<br>CPOL | Clock Polarity<br><br>Selects the inactive state of the Serial Communications Clock (SCK).<br><br>0 The inactive state value of SCK is low.<br>1 The inactive state value of SCK is high. |
| 25<br>CPHA | Clock Phase<br><br>Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode , the bit value is ignored and the transfers are done as the CPHA bit is set to 1.<br><br>0 Data is captured on the leading edge of SCK and changed on the following edge.<br>1 Data is changed on the leading edge of SCK and captured on the following edge. |
| 24–23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 21–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 49.3.5 Status Register (SPIx_SR)

SR contains status and flag bits. The bits reflect the status of the module and indicate the occurrence of events that can generate interrupt or DMA requests. Software can clear flag bits in the SR by writing a 1 to them. Writing a 0 to a flag bit has no effect. This register may not be writable in Module Disable mode due to the use of power saving mechanisms.

Address: Base address + 2Ch offset

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TCF | TXRXS | 0 | EOQF | TFUF | 0 | TFFF | 0 | 0 | 0 | 0 | 0 | RFOF | 0 | RFDF | 0 |
| W | w1c | w1c | | w1c | w1c | | w1c | | | | | | w1c | | w1c | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TXCTR | | | | TXNXTPTR | | | | RXCTR | | | | POPNXTPTR | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### SPIx_SR field descriptions

| Field | Description |
|---|---|
| 31<br>TCF | Transfer Complete Flag<br><br>Indicates that all bits in a frame have been shifted out. TCF remains set until it is cleared by writing a 1 to it.<br><br>0 Transfer not complete.<br>1 Transfer complete. |
| 30<br>TXRXS | TX and RX Status<br><br>Reflects the run status of the module.<br><br>0 Transmit and receive operations are disabled (The module is in Stopped state).<br>1 Transmit and receive operations are enabled (The module is in Running state). |
| 29<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

*Table continues on the next page...*

## SPIx_SR field descriptions (continued)

| Field | Description |
|---|---|
| 28<br>EOQF | End of Queue Flag<br><br>Indicates that the last entry in a queue has been transmitted when the module is in Master mode. The EOQF bit is set when the TX FIFO entry has the EOQ bit set in the command halfword and the end of the transfer is reached. The EOQF bit remains set until cleared by writing a 1 to it. When the EOQF bit is set, the TXRXS bit is automatically cleared.<br><br>0 EOQ is not set in the executing command.<br>1 EOQ is set in the executing SPI command. |
| 27<br>TFUF | Transmit FIFO Underflow Flag<br><br>Indicates an underflow condition in the TX FIFO. The transmit underflow condition is detected only for SPI blocks operating in Slave mode and SPI configuration. TFUF is set when the TX FIFO of the module operating in SPI Slave mode is empty and an external SPI master initiates a transfer. The TFUF bit remains set until cleared by writing 1 to it.<br><br>0 No TX FIFO underflow.<br>1 TX FIFO underflow has occurred. |
| 26<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 25<br>TFFF | Transmit FIFO Fill Flag<br><br>Provides a method for the module to request more entries to be added to the TX FIFO. The TFFF bit is set while the TX FIFO is not full. The TFFF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller to the TX FIFO full request.<br><br>0 TX FIFO is full.<br>1 TX FIFO is not full. |
| 24<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 21<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 20<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 19<br>RFOF | Receive FIFO Overflow Flag<br><br>Indicates an overflow condition in the RX FIFO. The field is set when the RX FIFO and shift register are full and a transfer is initiated. The bit remains set until it is cleared by writing a 1 to it.<br><br>0 No Rx FIFO overflow.<br>1 Rx FIFO overflow has occurred. |
| 18<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 17<br>RFDF | Receive FIFO Drain Flag |

*Table continues on the next page...*

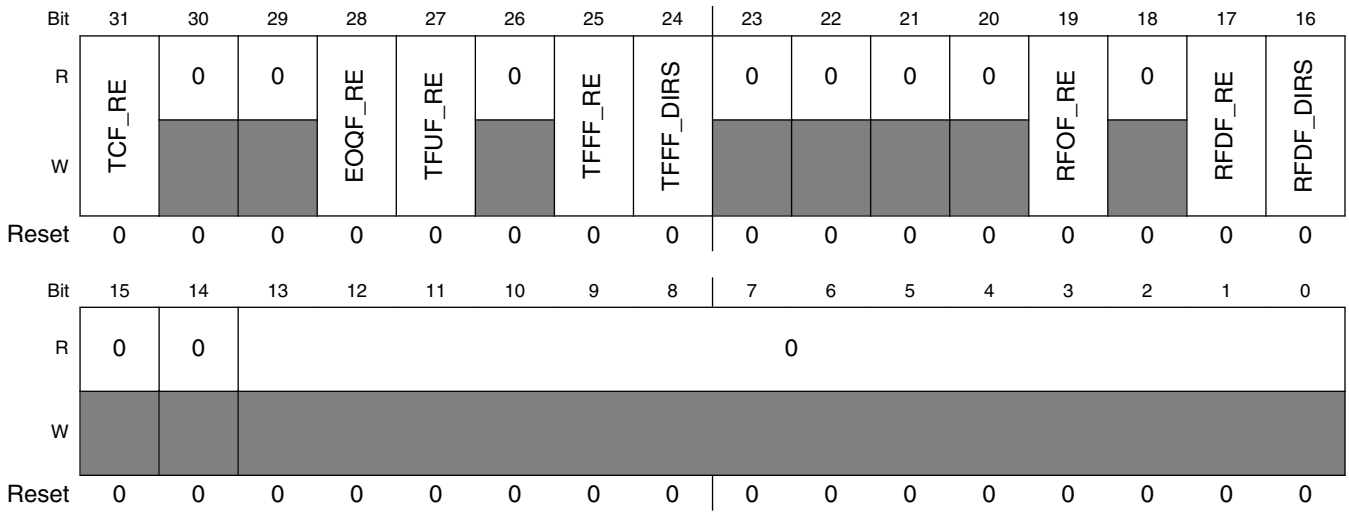
## SPIx_SR field descriptions (continued)

| Field | Description |
|---|---|
| | Provides a method for the module to request that entries be removed from the RX FIFO. The bit is set while the RX FIFO is not empty. The RFDF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller when the RX FIFO is empty.<br><br>0 RX FIFO is empty.<br>1 RX FIFO is not empty. |
| 16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–12<br>TXCTR | TX FIFO Counter<br><br>Indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the PUSHR is written. The TXCTR is decremented every time an SPI command is executed and the SPI data is transferred to the shift register. |
| 11–8<br>TXNXTPTR | Transmit Next Pointer<br><br>Indicates which TX FIFO entry is transmitted during the next transfer. The TXNXTPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register. |
| 7–4<br>RXCTR | RX FIFO Counter<br><br>Indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the POPR is read. The RXCTR is incremented every time data is transferred from the shift register to the RX FIFO. |
| 3–0<br>POPNXTPTR | Pop Next Pointer<br><br>Contains a pointer to the RX FIFO entry to be returned when the POPR is read. The POPNXTPTR is updated when the POPR is read. |

## 49.3.6 DMA/Interrupt Request Select and Enable Register (SPIx_RSER)

RSER controls DMA and interrupt requests. Do not write to the RSER while the module is in the Running state.

Address: Base address + 30h offset




**SPIx_RSER field descriptions**

| Field | Description |
|---|---|
| 31<br>TCF_RE | Transmission Complete Request Enable<br><br>Enables TCF flag in the SR to generate an interrupt request.<br><br>0 TCF interrupt requests are disabled.<br>1 TCF interrupt requests are enabled. |
| 30<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 29<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 28<br>EOQF_RE | Finished Request Enable<br><br>Enables the EOQF flag in the SR to generate an interrupt request.<br><br>0 EOQF interrupt requests are disabled.<br>1 EOQF interrupt requests are enabled. |
| 27<br>TFUF_RE | Transmit FIFO Underflow Request Enable<br><br>Enables the TFUF flag in the SR to generate an interrupt request.<br><br>0 TFUF interrupt requests are disabled.<br>1 TFUF interrupt requests are enabled. |

*Table continues on the next page...*

## SPIx_RSER field descriptions (continued)

| Field | Description |
|---|---|
| 26<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 25<br>TFFF_RE | Transmit FIFO Fill Request Enable<br><br>Enables the TFFF flag in the SR to generate a request. The TFFF_DIRS bit selects between generating an interrupt request or a DMA request.<br><br>0 TFFF interrupts or DMA requests are disabled.<br>1 TFFF interrupts or DMA requests are enabled. |
| 24<br>TFFF_DIRS | Transmit FIFO Fill DMA or Interrupt Request Select<br><br>Selects between generating a DMA request or an interrupt request. When SR[TFFF] and RSER[TFFF_RE] are set, this field selects between generating an interrupt request or a DMA request.<br><br>0 TFFF flag generates interrupt requests.<br>1 TFFF flag generates DMA requests. |
| 23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 21<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 20<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 19<br>RFOF_RE | Receive FIFO Overflow Request Enable<br><br>Enables the RFOF flag in the SR to generate an interrupt request.<br><br>0 RFOF interrupt requests are disabled.<br>1 RFOF interrupt requests are enabled. |
| 18<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 17<br>RFDF_RE | Receive FIFO Drain Request Enable<br><br>Enables the RFDF flag in the SR to generate a request. The RFDF_DIRS bit selects between generating an interrupt request or a DMA request.<br><br>0 RFDF interrupt or DMA requests are disabled.<br>1 RFDF interrupt or DMA requests are enabled. |
| 16<br>RFDF_DIRS | Receive FIFO Drain DMA or Interrupt Request Select<br><br>Selects between generating a DMA request or an interrupt request. When the RFDF flag bit in the SR is set, and the RFDF_RE bit in the RSER is set, the RFDF_DIRS bit selects between generating an interrupt request or a DMA request.<br><br>0 Interrupt request.<br>1 DMA request. |
| 15<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

*Table continues on the next page...*

**SPIx_RSER field descriptions (continued)**

| Field | Description |
|---|---|
| 14<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 13–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

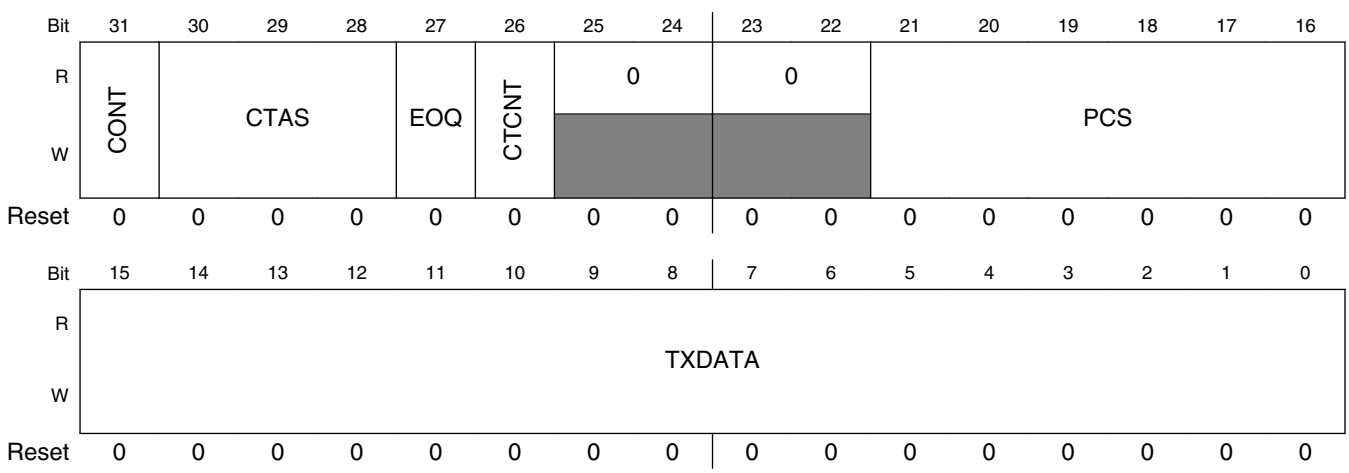
## 49.3.7 PUSH TX FIFO Register In Master Mode (SPIx_PUSHR)

PUSHR provides the means to write to the TX FIFO . Data written to this register is transferred to the TX FIFO 8- or 16-bit write accesses to the PUSHR transfer all 32 register bits to the TXFIFO.The register structure is different in Master and Slave modes. In Master mode, the register provides 16-bit command and data to the TX FIFO. In Slave mode, all 32 register bits can be used as data, supporting upto 32-bit SPI Frame operation.

A PUSHR Read Operation returns the topmost TX FIFO entry.

When the module is disabled, any writes to this register will not update the FIFO. Hence any reads performed during Module disable mode will return the last PUSHR write performed when Module was enabled.

Address: Base address + 34h offset




**SPIx_PUSHR field descriptions**

| Field | Description |
|---|---|
| 31<br>CONT | Continuous Peripheral Chip Select Enable<br><br>Selects a continuous selection format. The bit is used in SPI Master mode. The bit enables the selected PCS signals to remain asserted between transfers. |

*Table continues on the next page...*

## SPIx_PUSHR field descriptions (continued)

| Field | Description |
|---|---|
| | 0 Return PCSn signals to their inactive state between transfers.<br>1 Keep PCSn signals asserted between transfers. |
| 30–28<br>CTAS | Clock and Transfer Attributes Select<br><br>Selects which CTAR to use in master mode to specify the transfer attributes for the associated SPI frame. In SPI Slave mode, CTAR0 is used. See the chapter on chip configuration to determine how many CTARs this device has. You should not program a value in this field for a register that is not present.<br><br>000 CTAR0<br>001 CTAR1<br>010 Reserved<br>011 Reserved<br>100 Reserved<br>101 Reserved<br>110 Reserved<br>111 Reserved |
| 27<br>EOQ | End Of Queue<br><br>Host software uses this bit to signal to the module that the current SPI transfer is the last in a queue. At the end of the transfer, the EOQF bit in the SR is set.<br><br>0 The SPI data is not the last data to transfer.<br>1 The SPI data is the last data to transfer. |
| 26<br>CTCNT | Clear Transfer Counter<br><br>Clears the TCNT field in the TCR register. The TCNT field is cleared before the module starts transmitting the current SPI frame.<br><br>0 Do not clear the TCR[TCNT] field.<br>1 Clear the TCR[TCNT] field. |
| 25–24<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 23–22<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 21–16<br>PCS | Select which PCS signals are to be asserted for the transfer. Refer to the chip configuration chapter for the number of PCS signals used in this MCU.<br><br>0 Negate the PCS[x] signal.<br>1 Assert the PCS[x] signal. |
| 15–0<br>TXDATA | Transmit Data<br><br>Holds SPI data to be transferred according to the associated SPI command. |

## 49.3.8 PUSH TX FIFO Register In Slave Mode (SPIx_PUSHR_SLAVE)

PUSHR provides the means to write to the TX FIFO. Data written to this register is transferred to the TX FIFO. 8- or 16-bit write accesses to the PUSHR transfer all 32 register bits to the TXFIFO.The register structure is different in master and slave modes. In master mode the register provides 16-bit command and data to the TX FIFO. In slave mode, all 32 register bits can be used as data, supporting upto 32-bit SPI Frame operation.

Address: Base address + 34h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TXDATA | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SPIx_PUSHR_SLAVE field descriptions**

| Field | Description |
|---|---|
| 31–0<br>TXDATA | Transmit Data<br><br>Holds SPI data to be transferred according to the associated SPI command. |

## 49.3.9 POP RX FIFO Register (SPIx_POPR)

POPR is used to read the RX FIFO. Eight- or sixteen-bit read accesses to the POPR have the same effect on the RX FIFO as 32-bit read accesses. A write to this register will generate a Transfer Error.

Address: Base address + 38h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RXDATA | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

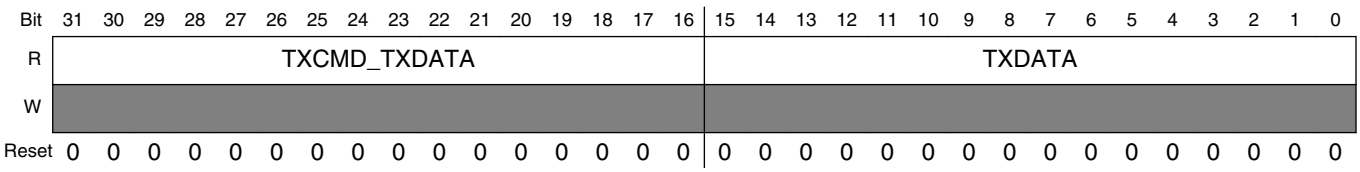
**SPIx_POPR field descriptions**

| Field | Description |
|---|---|
| 31–0<br>RXDATA | Received Data<br><br>Contains the SPI data from the RX FIFO entry to which the Pop Next Data Pointer points. |

## 49.3.10 Transmit FIFO Registers (SPIx_TXFR*n*)

TXFRn registers provide visibility into the TX FIFO for debugging purposes. Each register is an entry in the TX FIFO. The registers are read-only and cannot be modified. Reading the TXFRx registers does not alter the state of the TX FIFO.

Address: Base address + 3Ch offset + (4d × i), where i=0d to 3d



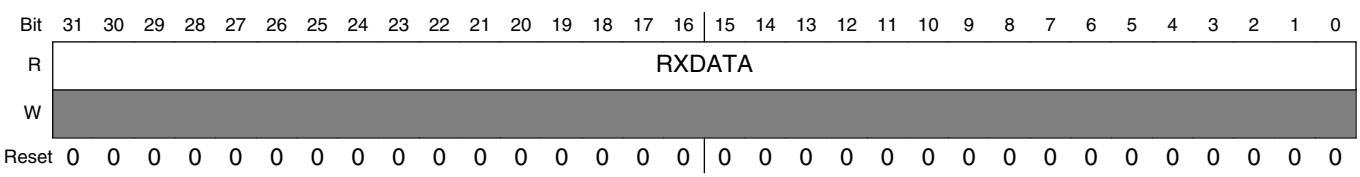

**SPIx_TXFR*n* field descriptions**

| Field | Description |
|---|---|
| 31–16<br>TXCMD_<br>TXDATA | Transmit Command or Transmit Data<br><br>In Master mode the TXCMD field contains the command that sets the transfer attributes for the SPI data.In Slave mode, the TXDATA contains 16 MSB bits of the SPI data to be shifted out. |
| 15–0<br>TXDATA | Transmit Data<br><br>Contains the SPI data to be shifted out. |

## 49.3.11 Receive FIFO Registers (SPIx_RXFR*n*)

RXFRn provide visibility into the RX FIFO for debugging purposes. Each register is an entry in the RX FIFO. The RXFR registers are read-only. Reading the RXFRx registers does not alter the state of the RX FIFO.

Address: Base address + 7Ch offset + (4d × i), where i=0d to 3d




**SPIx_RXFR*n* field descriptions**

| Field | Description |
|---|---|
| 31–0<br>RXDATA | Receive Data<br><br>Contains the received SPI data. |

**SPIx_RXFR*n* field descriptions (continued)**

| Field | Description |
|---|---|

## 49.4 Functional description

The Serial Peripheral Interface (SPI) block supports full-duplex, synchronous serial communications between MCUs and peripheral devices. The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes.

The SPI has the following configurations

- The SPI Configuration in which the module operates as a basic SPI or a queued SPI.

The DCONF field in the Module Configuration Register (MCR) determines the module Configuration. SPI configuration is selected when DCONF within SPIx_MCR is 0b00.

The CTARn registers hold clock and transfer attributes. The SPI configuration allows to select which CTAR to use on a frame by frame basis by setting a field in the SPI command.

See Clock and Transfer Attributes Register (In Master Mode) (SPI_CTAR*n*) for information on the fields of CTAR registers.

Typical master to slave connections are shown in the following figure. When a data transfer operation is performed, data is serially shifted a predetermined number of bit positions. Because the modules are linked, data is exchanged between the master and the slave. The data that was in the master shift register is now in the shift register of the slave, and vice versa. At the end of a transfer, the Transfer Control Flag(TCF) bit in the Shift Register(SR) is set to indicate a completed frame transfer.

**Figure 49-91. SPI serial protocol overview**

Generally, more than one slave device can be connected to the module master. Six Peripheral Chip Select (PCS) signals of the module masters can be used to select which of the slaves to communicate with. Refer to the chip configuration chapter for the number of PCS signals used in this MCU.

The SPI configuration share transfer protocol and timing properties which are described independently of the configuration in Transfer formats . The transfer rate and delay settings are described in Module baud rate and clock delay generation.

### 49.4.1 Start and Stop of module transfers

The module has two operating states: Stopped and Running. Both the states are independent of it's configuration. The default state of the module is Stopped. In the Stopped state, no serial transfers are initiated in Master mode and no transfers are responded to in Slave mode. The Stopped state is also a safe state for writing the various configuration registers of the module without causing undetermined results. In the Running state serial transfers take place.

The TXRXS bit in the SR indicates the state of module. The bit is set if the module is in Running state.

The module starts or transitions to Running when all of the following conditions are true:

- SR[EOQF] bit is clear
- MCU is not in the Debug mode or the MCR[FRZ] bit is clear
- MCR[HALT] bit is clear

The module stops or transitions from Running to Stopped after the current frame when any one of the following conditions exist:

- SR[EOQF] bit is set
- MCU in the Debug mode and the MCR[FRZ] bit is set
- MCR[HALT] bit is set

State transitions from Running to Stopped occur on the next frame boundary if a transfer is in progress, or immediately if no transfers are in progress.

### 49.4.2 Serial Peripheral Interface (SPI) configuration

The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The module is in SPI configuration when the DCONF field in the MCR is 0b00. The SPI frames can be 32 bits long. The host CPU or a DMA controller transfers the SPI data from the external to SPI RAM queues to a TX FIFO buffer. The received data is stored in entries in the RX FIFO buffer. The host CPU or the DMA controller transfers the received data from the RX FIFO to memory external to the module. The operation of FIFO buffers is described in Transmit First In First Out (TX FIFO) buffering mechanism, Transmit First In First Out (TX FIFO) buffering mechanism and Receive First In First Out (RX FIFO) buffering mechanism. The interrupt and DMA request conditions are described in Interrupts/DMA requests.

The SPI configuration supports two block-specific modes—Master mode and Slave mode.In Master mode the module initiates and controls the transfer according to the fields of the executing SPI Command. In Slave mode, the module responds only to transfers initiated by a bus master external to it and the SPI command field space is reserved.

#### 49.4.2.1 Master mode

In SPI Master, mode the module initiates the serial transfers by controlling the SCK and the PCS signals. The executing SPI Command determines which CTARs will be used to set the transfer attributes and which PCS signals to assert . The command field also contains various bits that help with queue management and transfer protocol . See PUSH TX FIFO Register In Master Mode (SPI_PUSHR) for details on the SPI command fields. The data in the executing TX FIFO entry is loaded into the shift register and shifted out on the Serial Out (SOUT) pin. In SPI Master mode, each SPI frame to be transmitted has a command associated with it, allowing for transfer attribute control on a frame by frame basis.

#### 49.4.2.2 Slave mode

In SPI Slave mode the module responds to transfers initiated by an SPI bus master. It does not initiate transfers. Certain transfer attributes such as clock polarity, clock phase, and frame size must be set for successful communication with an SPI master. The SPI Slave mode transfer attributes are set in the CTAR0 . The data is shifted out with MSB first. Shifting out of LSB is not supported in this mode.

#### 49.4.2.3 FIFO disable operation

The FIFO disable mechanisms allow SPI transfers without using the TX FIFO or RX FIFO. The module operates as a double-buffered simplified SPI when the FIFOs are disabled. The Transmit and Receive side of the FIFOs are disabled separately; setting the MCR[DIS_TXF] bit disables the TX FIFO, and setting the MCR[DIS_RXF] bit disables the RX FIFO.

The FIFO disable mechanisms are transparent to the user and to host software. Transmit data and commands are written to the PUSHR and received data is read from the POPR.

When the TX FIFO is disabled, the fields SR[TFFF], SR[TFUF] and SR[TXCTR] behave as if there is a one-entry FIFO but the contents of TXFRs, SR[TXNXTPTR] are undefined. Similarly, when the RX FIFO is disabled, the RFDF, RFOF, and RXCTR fields in the SR behave as if there is a one-entry FIFO, but the contents of the RXFR registers and POPNXTPTR are undefined.

#### 49.4.2.4 Transmit First In First Out (TX FIFO) buffering mechanism

The TX FIFO functions as a buffer of SPI data for transmission. The TX FIFO holds four words, each consisting of SPI data. The number of entries in the TX FIFO is device-specific. SPI data is added to the TX FIFO by writing to the Data Field of module PUSH FIFO Register (PUSHR). TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

The TX FIFO Counter field (TXCTR) in the module Status Register (SR) indicates the number of valid entries in the TX FIFO. The TXCTR is updated every time a 8- or 16-bit write takes place to the Data Field of SPI_PUSHR or SPI data is transferred into the shift register from the TX FIFO.

The TXNXTPTR field indicates the TX FIFO Entry that will be transmitted during the next transfer. The TXNXTPTR field is incremented every time SPI data is transferred from the TX FIFO to the shift register. The maximum value of the field is equal to the maximum implemented TXFR number and it rolls over after reaching the maximum.

##### 49.4.2.4.1 Filling the TX FIFO

Host software or other intelligent blocks can add (push) entries to the TX FIFO by writing to the PUSHR. When the TX FIFO is not full, the TX FIFO Fill Flag (TFFF) in the SR is set. The TFFF bit is cleared when TX FIFO is full and the DMA controller indicates that a write to PUSHR is complete. Writing a '1' to the TFFF bit also clears it. The TFFF can generate a DMA request or an interrupt request. See Transmit FIFO Fill Interrupt or DMA Request for details.

The module ignores attempts to push data to a full TX FIFO, and the state of the TX FIFO does not change and no error condition is indicated.

##### 49.4.2.4.2 Draining the TX FIFO

The TX FIFO entries are removed (drained) by shifting SPI data out through the shift register. Entries are transferred from the TX FIFO to the shift register and shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the

TX FIFO to the shift register, the TX FIFO Counter decrements by one. At the end of a transfer, the TCF bit in the SR is set to indicate the completion of a transfer. The TX FIFO is flushed by writing a '1' to the CLR_TXF bit in MCR.

If an external bus master initiates a transfer with a module slave while the slave's TX FIFO is empty, the Transmit FIFO Underflow Flag (TFUF) in the slave's SR is set. See Transmit FIFO Underflow Interrupt Request for details.

### 49.4.2.5 Receive First In First Out (RX FIFO) buffering mechanism

The RX FIFO functions as a buffer for data received on the SIN pin. The RX FIFO holds four received SPI data frames. The number of entries in the RX FIFO is device-specific. SPI data is added to the RX FIFO at the completion of a transfer when the received data in the shift register is transferred into the RX FIFO. SPI data are removed (popped) from the RX FIFO by reading the module POP RX FIFO Register (POPR). RX FIFO entries can only be removed from the RX FIFO by reading the POPR or by flushing the RX FIFO.

The RX FIFO Counter field (RXCTR) in the module's Status Register (SR) indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the POPR is read or SPI data is copied from the shift register to the RX FIFO.

The POPNXTPTR field in the SR points to the RX FIFO entry that is returned when the POPR is read. The POPNXTPTR contains the positive offset from RXFR0 in a number of 32-bit registers. For example, POPNXTPTR equal to two means that the RXFR2 contains the received SPI data that will be returned when the POPR is read. The POPNXTPTR field is incremented every time the POPR is read. The maximum value of the field is equal to the maximum implemented RXFR number and it rolls over after reaching the maximum.

#### 49.4.2.5.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register. While the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time an SPI frame is transferred to the RX FIFO, the RX FIFO Counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the RFOF bit in the SR is set indicating an overflow condition. Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

#### 49.4.2.5.2 Draining the RX FIFO

Host CPU or a DMA can remove (pop) entries from the RX FIFO by reading the module POP RX FIFO Register (POPR). A read of the POPR decrements the RX FIFO Counter by one. Attempts to pop data from an empty RX FIFO are ignored and the RX FIFO Counter remains unchanged. The data, read from the empty RX FIFO, is undetermined.

When the RX FIFO is not empty, the RX FIFO Drain Flag (RFDF) in the SR is set. The RFDF bit is cleared when the RX_FIFO is empty and the DMA controller indicates that a read from POPR is complete or by writing a 1 to it.

### 49.4.3 Module baud rate and clock delay generation

The SCK frequency and the delay values for serial transfer are generated by dividing the system clock frequency by a prescaler and a scaler with the option for doubling the baud rate. The following figure shows conceptually how the SCK signal is generated.

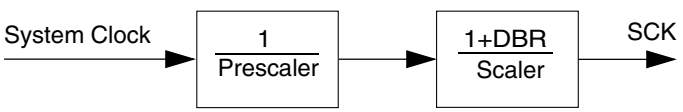



**Figure 49-92. Communications clock prescalers and scalers**

#### 49.4.3.1 Baud rate generator

The baud rate is the frequency of the SCK. The system clock is divided by a prescaler (PBR) and scaler (BR) to produce SCK with the possibility of halving the scaler division. The DBR, PBR, and BR fields in the CTARs select the frequency of SCK by the formula in the BR field description. The following table shows an example of how to compute the baud rate.

**Table 49-130. Baud rate computation example**

| $f_{SYS}$ | PBR | Prescaler | BR | Scaler | DBR | Baud rate |
|---|---|---|---|---|---|---|
| 100 MHz | 0b00 | 2 | 0b0000 | 2 | 0 | 25 Mb/s |
| 20 MHz | 0b00 | 2 | 0b0000 | 2 | 1 | 10 Mb/s |

**NOTE**

The clock frequencies mentioned in the preceding table are given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

### 49.4.3.2 PCS to SCK Delay ($t_{CSC}$)

The PCS to SCK delay is the length of time from assertion of the PCS signal to the first SCK edge. See Figure 49-94 for an illustration of the PCS to SCK delay. The PCSSCK and CSSCK fields in the CTAR*x* registers select the PCS to SCK delay by the formula in the CSSCK field description. The following table shows an example of how to compute the PCS to SCK delay.

**Table 49-131. PCS to SCK delay computation example**

| $f_{SYS}$ | PCSSCK | Prescaler | CSSCK | Scaler | PCS to SCK Delay |
|---|---|---|---|---|---|
| 100 MHz | 0b01 | 3 | 0b0100 | 32 | 0.96 µs |

**NOTE**

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

### 49.4.3.3 After SCK Delay ($t_{ASC}$)

The After SCK Delay is the length of time between the last edge of SCK and the negation of PCS. See Figure 49-94 and Figure 49-95 for illustrations of the After SCK delay. The PASC and ASC fields in the CTAR*x* registers select the After SCK Delay by the formula in the ASC field description. The following table shows an example of how to compute the After SCK delay.

**Table 49-132. After SCK Delay computation example**

| $f_{SYS}$ | PASC | Prescaler | ASC | Scaler | After SCK Delay |
|---|---|---|---|---|---|
| 100 MHz | 0b01 | 3 | 0b0100 | 32 | 0.96 µs |

**NOTE**

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

### 49.4.3.4 Delay after Transfer ($t_{DT}$)

The Delay after Transfer is the minimum time between negation of the PCS signal for a frame and the assertion of the PCS signal for the next frame. See Figure 49-94 for an illustration of the Delay after Transfer. The PDT and DT fields in the CTAR*x* registers select the Delay after Transfer by the formula in the DT field description. The following table shows an example of how to compute the Delay after Transfer.

**Table 49-133. Delay after Transfer computation example**

| $f_{SYS}$ | PDT | Prescaler | DT | Scaler | Delay after Transfer |
|---|---|---|---|---|---|
| 100 MHz | 0b01 | 3 | 0b1110 | 32768 | 0.98 ms |

**NOTE**

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

When in Non-Continuous Clock mode the $t_{DT}$ delay is configured according to the equation specified in the CTAR[DT] field description. When in Continuous Clock mode, the delay is fixed at 1 SCK period.

### 49.4.3.5 Peripheral Chip Select Strobe Enable ($\overline{PCSS}$)

The $\overline{PCSS}$ signal provides a delay to allow the PCS signals to settle after a transition occurs thereby avoiding glitches. When the Module is in Master mode and the PCSSE bit is set in the MCR, $\overline{PCSS}$ provides a signal for an external demultiplexer to decode the PCS[0] – PCS[4] signals into as many as 128 glitch-free PCS signals. The following figure shows the timing of the $\overline{PCSS}$ signal relative to PCS signals.

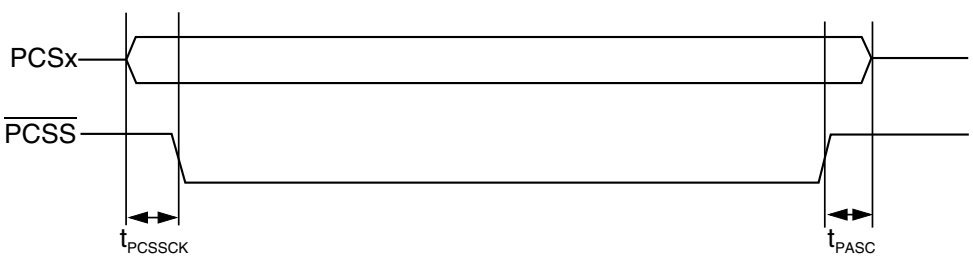



**Figure 49-93. Peripheral Chip Select Strobe timing**

The delay between the assertion of the PCS signals and the assertion of $\overline{PCSS}$ is selected by the PCSSCK field in the CTAR based on the following formula:

$$t_{PCSSCK} = \frac{1}{f_{SYS}} \times PCSSCK$$

At the end of the transfer, the delay between $\overline{\text{PCSS}}$ negation and PCS negation is selected by the PASC field in the CTAR based on the following formula:

$$t_{\text{PASC}} = \frac{1}{f_{\text{SYS}}} \times \text{PASC}$$

The following table shows an example of how to compute the $t_{pcssck}$ delay.

**Table 49-134. Peripheral Chip Select Strobe Assert computation example**

| $f_{SYS}$ | PCSSCK | Prescaler | Delay before Transfer |
|---|---|---|---|
| 100 MHz | 0b11 | 7 | 70.0 ns |

The following table shows an example of how to compute the $t_{pasc}$ delay.

**Table 49-135. Peripheral Chip Select Strobe Negate computation example**

| $f_{SYS}$ | PASC | Prescaler | Delay after Transfer |
|---|---|---|---|
| 100 MHz | 0b11 | 7 | 70.0 ns |

The $\overline{\text{PCSS}}$ signal is not supported when Continuous Serial Communication SCK mode is enabled.

**NOTE**

The clock frequency mentioned in the preceding tables is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

## 49.4.4 Transfer formats

The SPI serial communication is controlled by the Serial Communications Clock (SCK) signal and the PCS signals. The SCK signal provided by the master device synchronizes shifting and sampling of the data on the SIN and SOUT pins. The PCS signals serve as enable signals for the slave devices.

In Master mode, the CPOL and CPHA bits in the Clock and Transfer Attributes Registers (CTARn) select the polarity and phase of the serial clock, SCK.

- CPOL - Selects the idle state polarity of the SCK
- CPHA - Selects if the data on SOUT is valid before or on the first SCK edge

Even though the bus slave does not control the SCK signal, in Slave mode the values of CPOL and CPHA must be identical to the master device settings to ensure proper transmission. In SPI Slave mode, only CTAR0 is used.

The module supports four different transfer formats:

- Classic SPI with CPHA=0
- Classic SPI with CPHA=1
- Modified Transfer Format with CPHA = 0
- Modified Transfer Format with CPHA = 1

A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The module can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the MCR selects between Classic SPI Format and Modified Transfer Format.

In the SPI configurations, the module provides the option of keeping the PCS signals asserted between frames. See Continuous Selection Format for details.

#### 49.4.4.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in following figure is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SIN pins on the odd-numbered SCK edges and change the data on their SOUT pins on the even-numbered SCK edges.

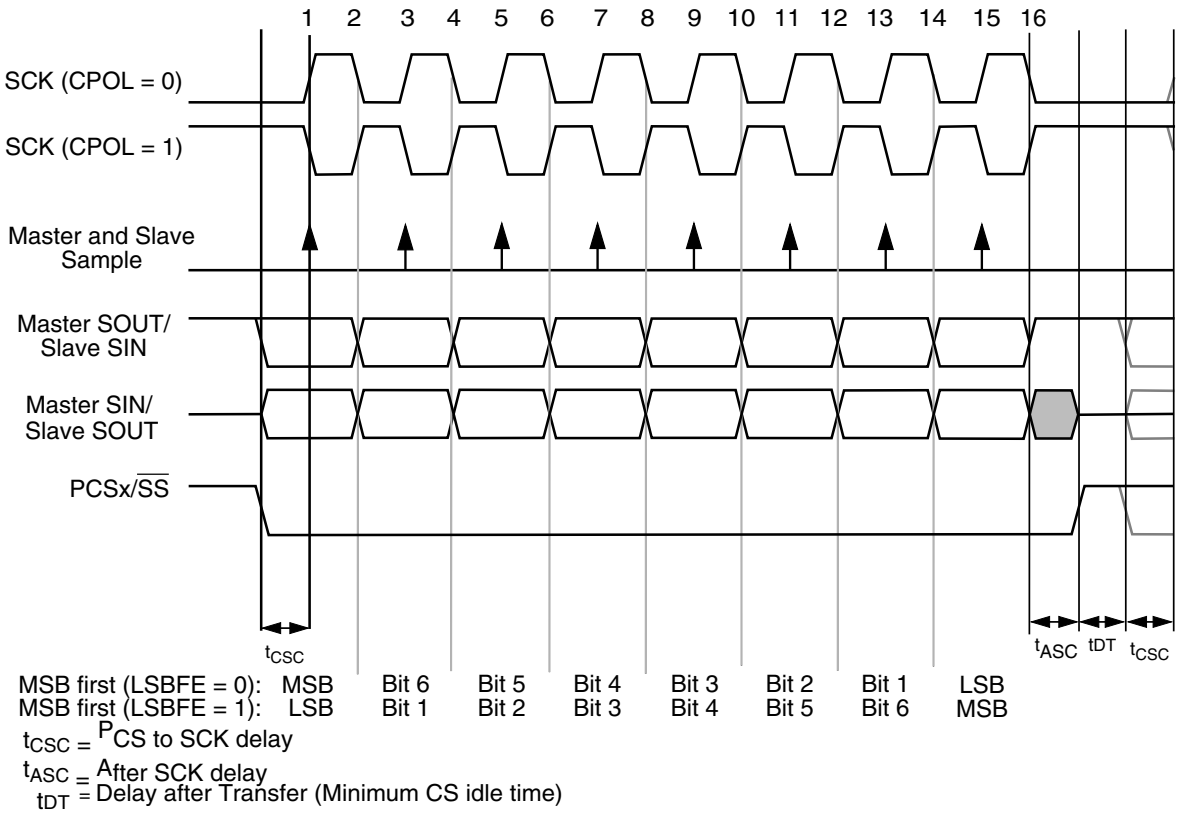



**Figure 49-94. Module transfer timing diagram (MTFE=0, CPHA=0, FMSZ=8)**

The master initiates the transfer by placing its first data bit on the SOUT pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT pin. After the $t_{ASC}$ delay elapses, the master outputs the first edge of SCK. The master and slave devices use this edge to sample the first input data bit on their serial data input signals. At the second edge of the SCK, the master and slave devices place their second data bit on their serial data output signals. For the rest of the frame the master and the slave sample their SIN pins on the odd-numbered clock edges and changes the data on their SOUT pins on the even-numbered clock edges. After the last clock edge occurs, a delay of $t_{ASC}$ is inserted before the master negates the PCS signals. A delay of $t_{DT}$ is inserted before a new frame transfer can be initiated by the master.

### 49.4.4.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in the following figure is used to communicate with peripheral SPI slave devices that require the first SCK edge before the first data bit becomes available on the slave SOUT pin. In this format, the master and slave devices change the data on their SOUT pins on the odd-numbered SCK edges and sample the data on their SIN pins on the even-numbered SCK edges .

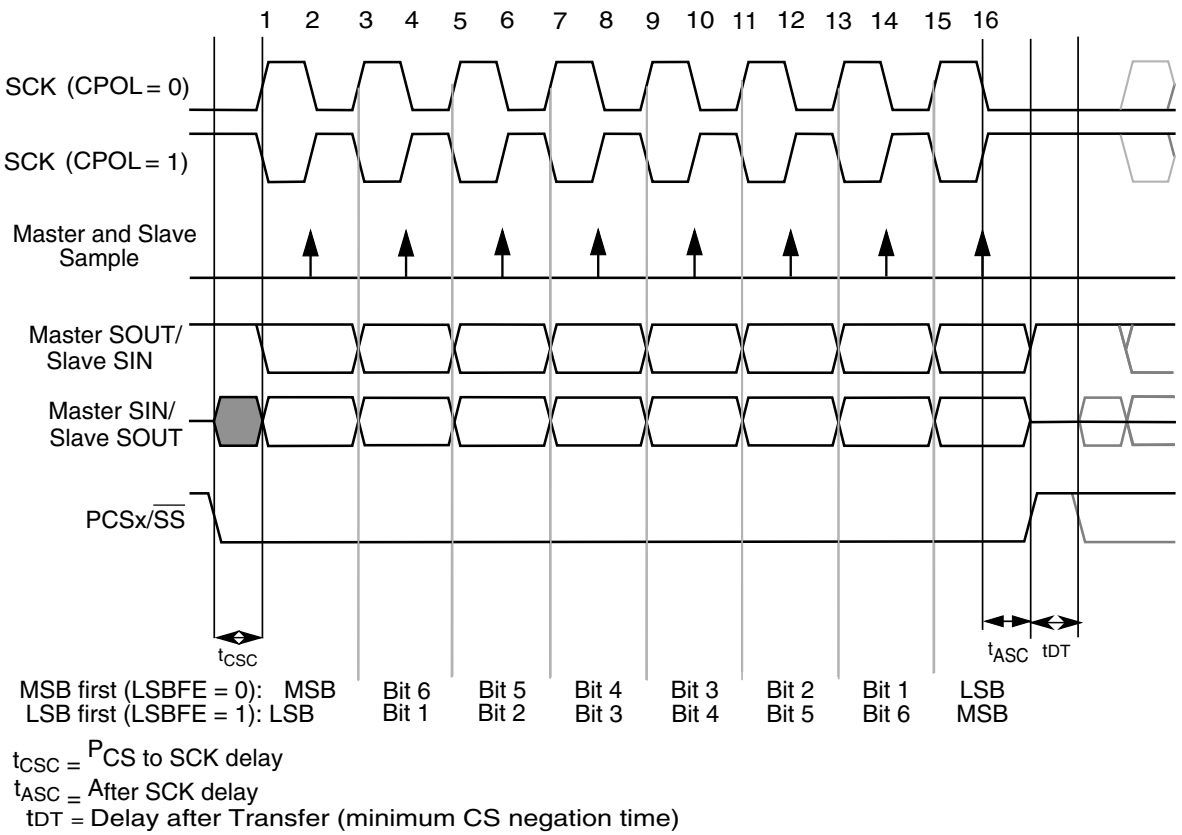



**Figure 49-95. Module transfer timing diagram (MTFE=0, CPHA=1, FMSZ=8)**

The master initiates the transfer by asserting the PCS signal to the slave. After the $t_{CSC}$ delay has elapsed, the master generates the first SCK edge and at the same time places valid data on the master SOUT pin .The slave responds to the first SCK edge by placing its first data bit on its slave SOUT pin.

At the second edge of the SCK the master and slave sample their SIN pins. For the rest of the frame the master and the slave change the data on their SOUT pins on the odd-numbered clock edges and sample their SIN pins on the even-numbered clock edges. After the last clock edge occurs, a delay of $t_{ASC}$ is inserted before the master negates the PCS signal. A delay of $t_{DT}$ is inserted before a new frame transfer can be initiated by the master.

### 49.4.4.3 Continuous Selection Format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The Continuous Selection Format provides the flexibility to handle the following case. The Continuous Selection Format is enabled for the SPI configuration by setting the CONT bit in the SPI command. The behavior of the PCS signals in the configurations is identical so only SPI configuration will be described.

When the CONT bit = 0, the module drives the asserted Chip Select signals to their idle states in between frames. The idle states of the Chip Select signals are selected by the PCSISn bits in the MCR. The following timing diagram is for two four-bit transfers with CPHA = 1 and CONT = 0.

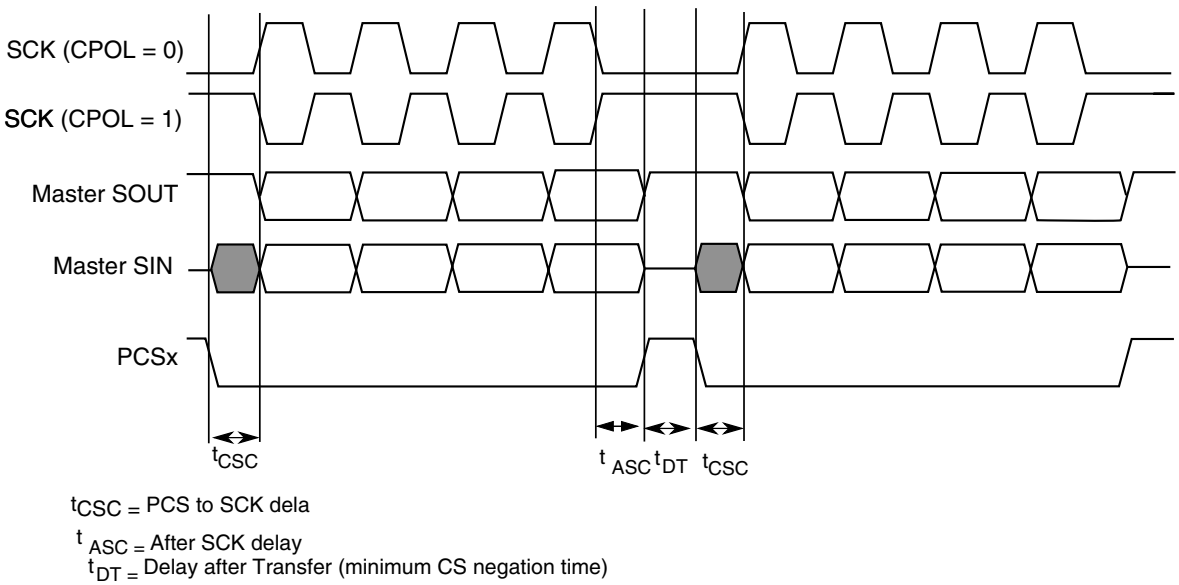



**Figure 49-96. Example of non-continuous format (CPHA=1, CONT=0)**

When the CONT bit = 1, the PCS signal remains asserted for the duration of the two transfers. The Delay between Transfers ($t_{DT}$) is not inserted between the transfers. The following figure shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 1.

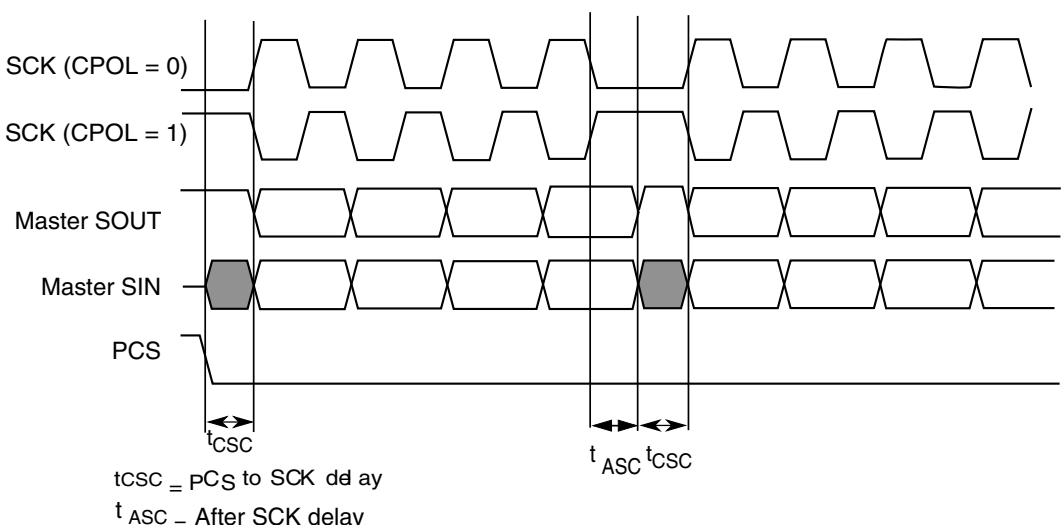



**Figure 49-97. Example of continuous transfer (CPHA=1, CONT=1)**

When using the module with continuous selection follow these rules:

- All transmit commands must have the same PCSn bits programming.
- The CTARs, selected by transmit commands, must be programmed with the same transfer attributes. Only FMSZ field can be programmed differently in these CTARs.
- When transmitting multiple frames in this mode, the user software must ensure that the last frame has the PUSHR[CONT] bit deasserted in Master mode and the user software must provide sufficient frames in the TX_FIFO to be sent out in Slave mode and the master deasserts the PCSn at end of transmission of the last frame.
- The PUSHR[CONT] / DSICR0[DCONT] bits must be deasserted before asserting MCR[HALT] bit in master mode. This will make sure that the PCSn signals are deasserted. Asserting MCR[HALT] bit during continuous transfer will cause the PCSn signals to remain asserted and hence Slave Device cannot transition from Running to Stopped state.

**NOTE**

User must fill the TX FIFO with the number of entries that will be concatenated together under one PCS assertion for both master and slave before the TX FIFO becomes empty.

When operating in Slave mode, ensure that when the last entry in the TX FIFO is completely transmitted, that is, the corresponding TCF flag is asserted and TXFIFO is empty, the slave is deselected for any further serial communication; otherwise, an underflow error occurs.

### 49.4.5 Continuous Serial Communications Clock

The module provides the option of generating a Continuous SCK signal for slave peripherals that require a continuous clock.

Continuous SCK is enabled by setting the CONT_SCKE bit in the MCR. Enabling this bit generates the Continuous SCK regardless of the MCR[HALT] bit status. Continuous SCK is valid in all configurations.

Continuous SCK is only supported for CPHA=1. Clearing CPHA is ignored if the CONT_SCKE bit is set. Continuous SCK is supported for Modified Transfer Format.

Clock and transfer attributes for the Continuous SCK mode are set according to the following rules:

- When the module is in SPI configuration, CTAR0 is used initially. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame is used.
- In all configurations, the currently selected CTAR remains in use until the start of a frame with a different CTAR specified, or the Continuous SCK mode is terminated.

It is recommended to keep the baud rate the same while using the Continuous SCK. Switching clock polarity between frames while using Continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the module is put into the External Stop mode or Module Disable mode.

Enabling Continuous SCK disables the PCS to SCK delay and the Delay after Transfer ($t_{DT}$) is fixed to one SCK cycle. The following figure is the timing diagram for Continuous SCK format with Continuous Selection disabled.

**NOTE**

In Continuous SCK mode, for the SPI transfer CTAR0 should always be used, and the TX FIFO must be cleared using the MCR[CLR_TXF] field before initiating transfer.

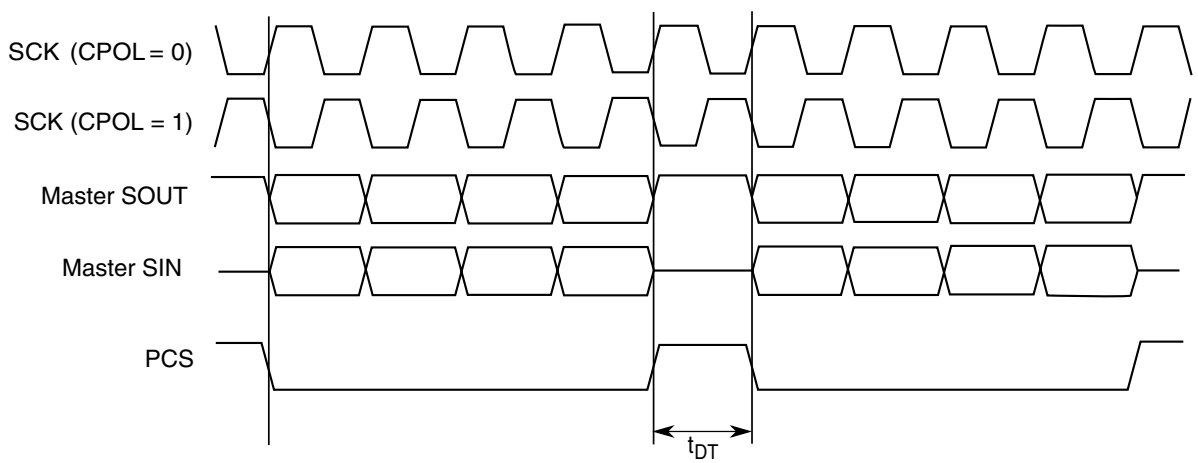



**Figure 49-98. Continuous SCK Timing Diagram (CONT=0)**

If the CONT bit in the TX FIFO entry is set, PCS remains asserted between the transfers. Under certain conditions, SCK can continue with PCS asserted, but with no data being shifted out of SOUT, that is, SOUT pulled high. This can cause the slave to receive incorrect data. Those conditions include:

- Continuous SCK with CONT bit set, but no data in the TX FIFO.

- Continuous SCK with CONT bit set and entering Stopped state (refer to Start and Stop of module transfers).
- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode.

The following figure shows timing diagram for Continuous SCK format with Continuous Selection enabled.

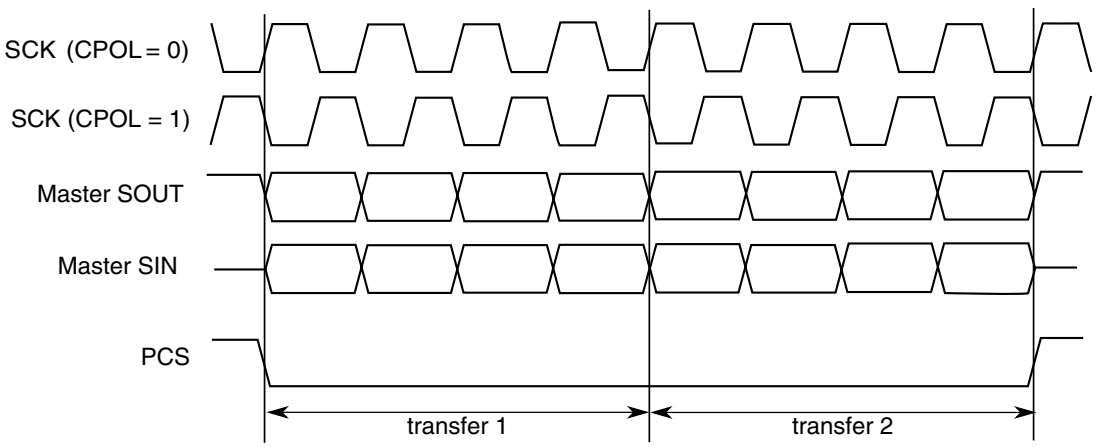



**Figure 49-99. Continuous SCK timing diagram (CONT=1)**

## 49.4.6 Slave Mode Operation Constraints

Slave mode logic shift register is buffered. This allows data streaming operation, when the SPI is permanently selected and data is shifted in with a constant rate.

The transmit data is transferred at second SCK clock edge of the each frame to the shift register if the $\overline{SS}$ signal is asserted and any time when transmit data is ready and $\overline{SS}$ signal is negated.

Received data is transferred to the receive buffer at last SCK edge of each frame, defined by frame size programmed to the CTAR0/1 register. Then the data from the buffer is transferred to the RXFIFO or DDR register.

If the $\overline{SS}$ negates before that last SCK edge, the data from shift register is lost.

## 49.4.7 Interrupts/DMA requests

The module has several conditions that can generate only interrupt requests and two conditions that can generate interrupt or DMA requests. The following table lists these conditions.

**Table 49-136. Interrupt and DMA request conditions**

| Condition | Flag | Interrupt | DMA |
|---|---|---|---|
| End of Queue (EOQ) | EOQF | Yes | - |
| TX FIFO Fill | TFFF | Yes | Yes |
| Transfer Complete | TCF | Yes | - |
| TX FIFO Underflow | TFUF | Yes | - |
| RX FIFO Drain | RFDF | Yes | Yes |
| RX FIFO Overflow | RFOF | Yes | - |

Each condition has a flag bit in the module Status Register (SR) and a Request Enable bit in the DMA/Interrupt Request Select and Enable Register (RSER). Certain flags (as shown in above table) generate interrupt requests or DMA requests depending on configuration of RSER register.

The module also provides a global interrupt request line, which is asserted when any of individual interrupt requests lines is asserted.

### 49.4.7.1 End of Queue Interrupt Request

The End of Queue Request indicates that the end of a transmit queue is reached. The End of Queue Request is generated when the EOQ bit in the executing SPI command is set and the EOQF_RE bit in the RSER is set.

**NOTE**

This interrupt request is generated when the last bit of the SPI frame with EOQ bit set is transmitted.

### 49.4.7.2 Transmit FIFO Fill Interrupt or DMA Request

The Transmit FIFO Fill Request indicates that the TX FIFO is not full. The Transmit FIFO Fill Request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the RSER is set. The TFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

**NOTE**

TFFF flag clears automatically when DMA is used to fill TX FIFO.

To clear TFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill TX FIFO:

1. Wait until TFFF = 1.
2. Write data to PUSHR using CPU.
3. Clear TFFF by writing a 1 to its location. If TX FIFO is not full, this flag will not clear.

#### 49.4.7.3 Transfer Complete Interrupt Request

The Transfer Complete Request indicates the end of the transfer of a serial frame. The Transfer Complete Request is generated at the end of each frame transfer when the TCF_RE bit is set in the RSER.

#### 49.4.7.4 Transmit FIFO Underflow Interrupt Request

The Transmit FIFO Underflow Request indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for the module operating in Slave mode and SPI configuration . The TFUF bit is set when the TX FIFO of a SPI is empty, and a transfer is initiated from an external SPI master. If the TFUF bit is set while the TFUF_RE bit in the RSER is set, an interrupt request is generated.

#### 49.4.7.5 Receive FIFO Drain Interrupt or DMA Request

The Receive FIFO Drain Request indicates that the RX FIFO is not empty. The Receive FIFO Drain Request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the RSER is set. The RFDF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

#### 49.4.7.6 Receive FIFO Overflow Interrupt Request

The Receive FIFO Overflow Request indicates that an overflow condition in the RX FIFO has occurred. A Receive FIFO Overflow request is generated when RX FIFO and shift register are full and a transfer is initiated. The RFOF_RE bit in the RSER must be set for the interrupt request to be generated.

Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

## 49.4.8 Power saving features

The module supports following power-saving strategies:

- External Stop mode
- Module Disable mode – Clock gating of non-memory mapped logic

### 49.4.8.1 Stop mode (External Stop mode)

The SPI supports the Stop mode protocol. When a request is made to enter External Stop mode, the SPI block acknowledges the request . If a serial transfer is in progress, the SPI waits until it reaches the frame boundary before it is ready to have its clocks shut off . While the clocks are shut off, the SPI memory-mapped logic is not accessible. This also puts the SPI in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. The states of the interrupt and DMA request signals cannot be changed while in External Stop mode.

### 49.4.8.2 Module Disable mode

Module Disable mode is a block-specific mode that the module can enter to save power. Host CPU can initiate the Module Disable mode by setting the MDIS bit in the MCR. The Module Disable mode can also be initiated by hardware. A power management block can initiate the Module Disable mode by asserting the DOZE mode signal while the DOZE bit in the MCR is set.

When the MDIS bit is set or the DOZE mode signal is asserted while the DOZE bit is set, the module negates Clock Enable signal at the next frame boundary. Once the Clock Enable signal is negated, it is said to have entered Module Disable Mode. This also puts the module in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state.If implemented, the Clock Enable signal can stop the clock to the non-memory mapped logic. When Clock Enable is negated, the module is in a dormant state, but the memory mapped registers are still accessible. Certain read or write operations have a different effect when the module is in the Module Disable mode. Reading the RX FIFO Pop Register does not change the state of the RX FIFO. Similarly, writing to the PUSHR

Register does not change the state of the TX FIFO. Clearing either of the FIFOs has no effect in the Module Disable mode. Changes to the DIS_TXF and DIS_RXF fields of the MCR have no effect in the Module Disable mode. In the Module Disable mode, all status bits and register flags in the module return the correct values when read, but writing to them has no effect. Writing to the TCR during Module Disable mode has no effect. Interrupt and DMA request signals cannot be cleared while in the Module Disable mode.

## 49.5 Initialization/application information

This section describes how to initialize the module.

### 49.5.1 How to manage queues

The queues are not part of the module, but it includes features in support of queue management. Queues are primarily supported in SPI configuration.

1. When module executes last command word from a queue, the EOQ bit in the command word is set to indicate it that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the SR is set.
3. The setting of the EOQF flag disables serial transmission and reception of data, putting the module in the Stopped state. The TXRXS bit is cleared to indicate the Stopped state.
4. The DMA can continue to fill TX FIFO until it is full or step 5 occurs.
5. Disable DMA transfers by disabling the DMA enable request for the DMA channel assigned to TX FIFO and RX FIFO. This is done by clearing the corresponding DMA enable request bits in the DMA Controller.
6. Ensure all received data in RX FIFO has been transferred to memory receive queue by reading the RXCNT in SR or by checking RFDF in the SR after each read operation of the POPR.
7. Modify DMA descriptor of TX and RX channels for new queues
8. Flush TX FIFO by writing a 1 to the CLR_TXF bit in the MCR. Flush RX FIFO by writing a '1' to the CLR_RXF bit in the MCR.
9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to SPI_TCNT field in the TCR.

10. Enable DMA channel by enabling the DMA enable request for the DMA channel assigned to the module TX FIFO, and RX FIFO by setting the corresponding DMA set enable request bit.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

### 49.5.2 Switching Master and Slave mode

When changing modes in the module, follow the steps below to guarantee proper operation.

1. Halt it by setting MCR[HALT].
2. Clear the transmit and receive FIFOs by writing a 1 to the CLR_TXF and CLR_RXF bits in MCR.
3. Set the appropriate mode in MCR[MSTR] and enable it by clearing MCR[HALT].

### 49.5.3 Initializing Module in Master/Slave Modes

Once the appropriate mode in MCR[MSTR] is configured, the module is enabled by clearing MCR[HALT]. It should be ensured that module Slave is enabled before enabling it's Master. This ensures the Slave is ready to be communicated with, before Master initializes communication.

### 49.5.4 Baud rate settings

The following table shows the baud rate that is generated based on the combination of the baud rate prescaler PBR and the baud rate scaler BR in the CTARs. The values calculated assume a 100 MHz system frequency and the double baud rate DBR bit is cleared.

**NOTE**

The clock frequency mentioned above is given as an example in this chapter. See the clocking chapter for the frequency used to drive this module in the device.

**Table 49-137. Baud rate values (bps)**

| | | Baud rate divider prescaler values | | | |
|---|---|---|---|---|---|
| | | 2 | 3 | 5 | 7 |
| Baud Rate Scaler Values | 2 | 25.0M | 16.7M | 10.0M | 7.14M |
| | 4 | 12.5M | 8.33M | 5.00M | 3.57M |
| | 6 | 8.33M | 5.56M | 3.33M | 2.38M |
| | 8 | 6.25M | 4.17M | 2.50M | 1.79M |
| | 16 | 3.12M | 2.08M | 1.25M | 893k |
| | 32 | 1.56M | 1.04M | 625k | 446k |
| | 64 | 781k | 521k | 312k | 223k |
| | 128 | 391k | 260k | 156k | 112k |
| | 256 | 195k | 130k | 78.1k | 55.8k |
| | 512 | 97.7k | 65.1k | 39.1k | 27.9k |
| | 1024 | 48.8k | 32.6k | 19.5k | 14.0k |
| | 2048 | 24.4k | 16.3k | 9.77k | 6.98k |
| | 4096 | 12.2k | 8.14k | 4.88k | 3.49k |
| | 8192 | 6.10k | 4.07k | 2.44k | 1.74k |
| | 16384 | 3.05k | 2.04k | 1.22k | 872 |
| | 32768 | 1.53k | 1.02k | 610 | 436 |

## 49.5.5 Delay settings

The following table shows the values for the Delay after Transfer ($t_{DT}$) and CS to SCK Delay ($T_{CSC}$) that can be generated based on the prescaler values and the scaler values set in the CTARs. The values calculated assume a 100 MHz system frequency.

**NOTE**

The clock frequency mentioned above is given as an example in this chapter. See the clocking chapter for the frequency used to drive this module in the device.

**Table 49-138. Delay values**

| | | Delay prescaler values | | | |
|---|---|---|---|---|---|
| | | 1 | 3 | 5 | 7 |
| Delay scaler values | 2 | 20.0 ns | 60.0 ns | 100.0 ns | 140.0 ns |
| | 4 | 40.0 ns | 120.0 ns | 200.0 ns | 280.0 ns |
| | 8 | 80.0 ns | 240.0 ns | 400.0 ns | 560.0 ns |
| | 16 | 160.0 ns | 480.0 ns | 800.0 ns | 1.1 µs |
| | 32 | 320.0 ns | 960.0 ns | 1.6 µs | 2.2 µs |
| | 64 | 640.0 ns | 1.9 µs | 3.2 µs | 4.5 µs |
| | 128 | 1.3 µs | 3.8 µs | 6.4 µs | 9.0 µs |
| | 256 | 2.6 µs | 7.7 µs | 12.8 µs | 17.9 µs |
| | 512 | 5.1 µs | 15.4 µs | 25.6 µs | 35.8 µs |
| | 1024 | 10.2 µs | 30.7 µs | 51.2 µs | 71.7 µs |
| | 2048 | 20.5 µs | 61.4 µs | 102.4 µs | 143.4 µs |
| | 4096 | 41.0 µs | 122.9 µs | 204.8 µs | 286.7 µs |
| | 8192 | 81.9 µs | 245.8 µs | 409.6 µs | 573.4 µs |
| | 16384 | 163.8 µs | 491.5 µs | 819.2 µs | 1.1 ms |
| | 32768 | 327.7 µs | 983.0 µs | 1.6 ms | 2.3 ms |
| | 65536 | 655.4 µs | 2.0 ms | 3.3 ms | 4.6 ms |

## 49.5.6 Calculation of FIFO pointer addresses

Complete visibility of the TX and RX FIFO contents is available through the FIFO registers, and valid entries can be identified through a memory-mapped pointer and counter for each FIFO. The pointer to the first-in entry in each FIFO is memory mapped. For the TX FIFO the first-in pointer is the Transmit Next Pointer (TXNXTPTR). For the RX FIFO the first-in pointer is the Pop Next Pointer (POPNXTPTR). The following figure illustrates the concept of first-in and last-in FIFO entries along with the FIFO Counter. The TX FIFO is chosen for the illustration, but the concepts carry over to the RX FIFO. See Transmit First In First Out (TX FIFO) buffering mechanism and Receive First In First Out (RX FIFO) buffering mechanism for details on the FIFO operation.

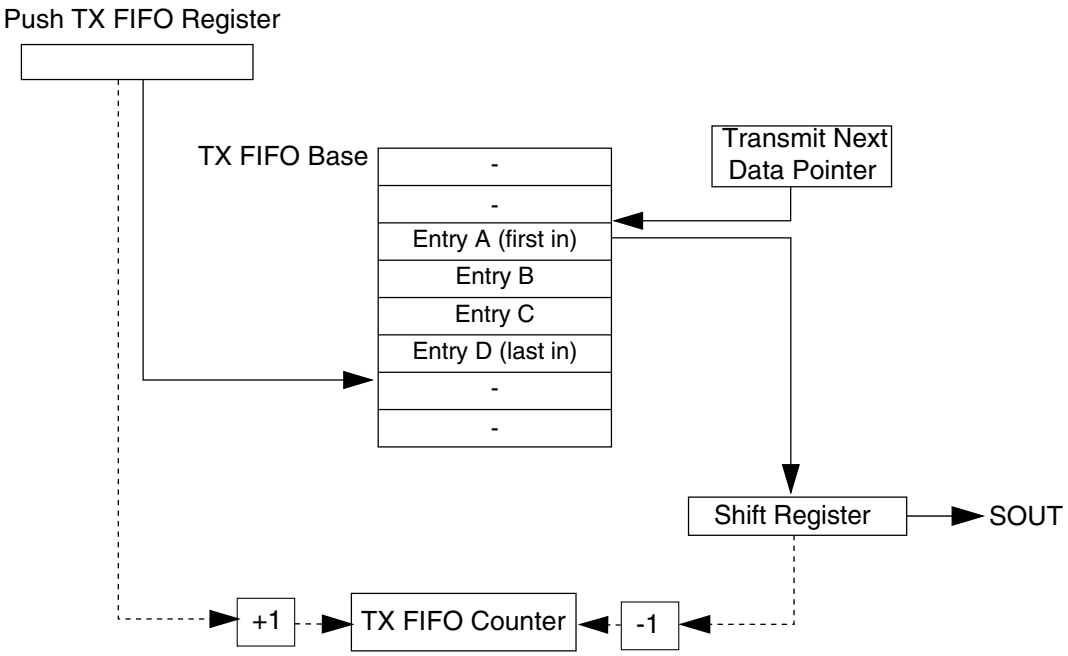



**Figure 49-100. TX FIFO pointers and counter**

### 49.5.6.1 Address Calculation for the First-in Entry and Last-in Entry in the TX FIFO

The memory address of the first-in entry in the TX FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{TXFIFOBase} + (4 \times \text{TXNXTPTR})$$

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{TXFIFOBase} + 4 \times (\text{TXCTR} + \text{TXNXTPTR} - 1) \bmod (\text{TXFIFOdepth})$$

TX FIFO Base - Base address of TX FIFO
TXCTR - TX FIFO Counter
TXNXTPTR - Transmit Next Pointer
TX FIFO Depth - Transmit FIFO depth, implementation specific

### 49.5.6.2 Address Calculation for the First-in Entry and Last-in Entry in the RX FIFO

The memory address of the first-in entry in the RX FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{RX FIFOBase} + \left(4 \times \text{POPNXTPTR}\right)$$

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{RX FIFO Base} + 4 \times \left(\text{RXCTR} + \text{POPNXTPTR} - 1\right) \text{mod} \left(\text{RXFIFOdepth}\right)$$

RX FIFO Base - Base address of RX FIFO
RXCTR - RX FIFO counter
POPNXTPTR - Pop Next Pointer
RX FIFO Depth - Receive FIFO depth, implementation specific

# Chapter 50
# Inter-Integrated Circuit (I2C)

## 50.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The inter-integrated circuit ($I^2C$, I2C, or IIC) module provides a method of communication between a number of devices. The interface is designed to operate up to 100 kbit/s with maximum bus loading and timing. The I2C device is capable of operating at higher baud rates, up to a maximum of clock/20, with reduced bus loading. The maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF. The I2C module also complies with the *System Management Bus (SMBus) Specification, version 2*.

### 50.1.1 Features

The I2C module has the following features:

- Compatible with *The $I^2C$-Bus Specification*
- Multimaster operation
- Software programmable for one of 64 different serial clock frequencies
- Software-selectable acknowledge bit
- Interrupt-driven byte-by-byte data transfer
- Arbitration-lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- START and STOP signal generation and detection
- Repeated START signal generation and detection
- Acknowledge bit generation and detection
- Bus busy detection
- General call recognition

- 10-bit address extension
- Support for *System Management Bus (SMBus) Specification, version 2*
- Programmable glitch input filter
- Low power mode wakeup on slave address match
- Range slave address support
- DMA support

### 50.1.2 Modes of operation

The I2C module's operation in various low power modes is as follows:

- Run mode: This is the basic mode of operation. To conserve power in this mode, disable the module.
- Wait mode: The module continues to operate when the core is in Wait mode and can provide a wakeup interrupt.
- Stop mode: The module is inactive in Stop mode for reduced power consumption, except that address matching is enabled in Stop mode. The STOP instruction does not affect the I2C module's register states.

### 50.1.3 Block diagram

The following figure is a functional block diagram of the I2C module.

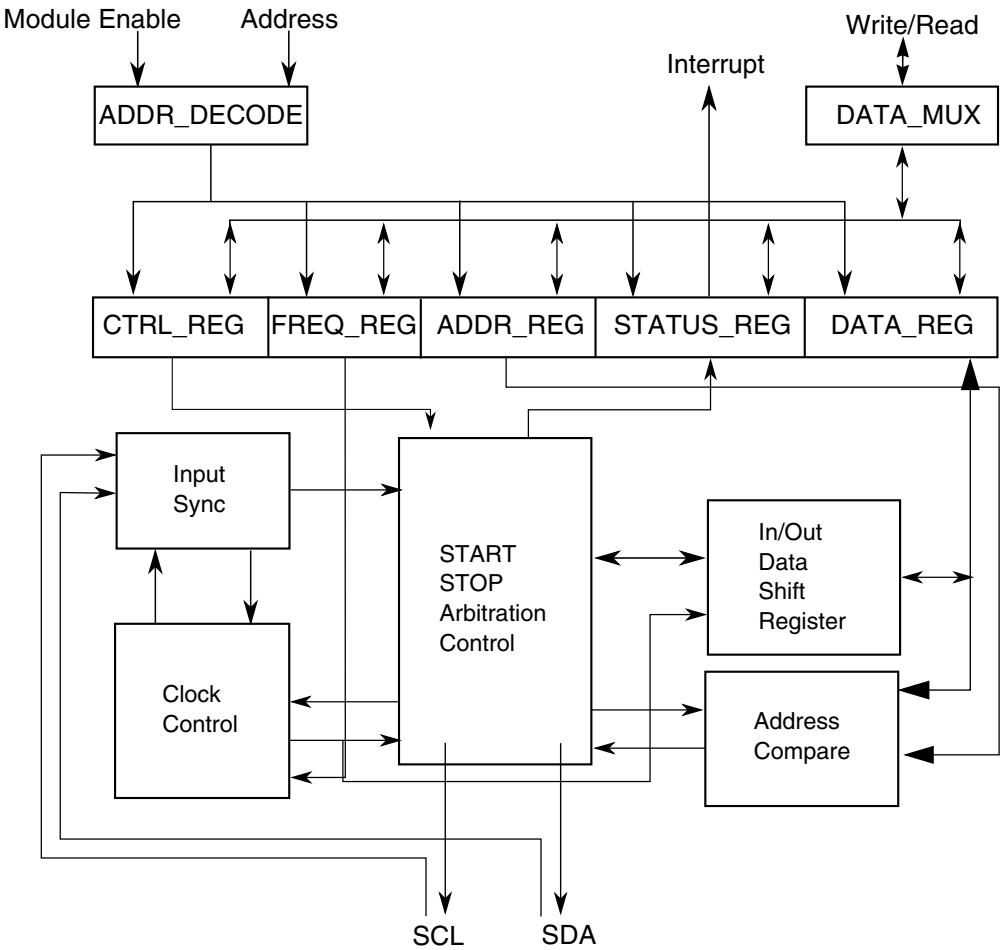



**Figure 50-1. I2C Functional block diagram**

## 50.2 I²C signal descriptions

The signal properties of I²C are shown in the following table.

**Table 50-1. I²C signal descriptions**

| Signal | Description | I/O |
|---|---|---|
| SCL | Bidirectional serial clock line of the I²C system. | I/O |
| SDA | Bidirectional serial data line of the I²C system. | I/O |

## 50.3 Memory map and register descriptions

This section describes in detail all I2C registers accessible to the end user.

## I2C memory map

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4006_6000 | I2C Address Register 1 (I2C0_A1) | 8 | R/W | 00h | 50.3.1/1333 |
| 4006_6001 | I2C Frequency Divider register (I2C0_F) | 8 | R/W | 00h | 50.3.2/1333 |
| 4006_6002 | I2C Control Register 1 (I2C0_C1) | 8 | R/W | 00h | 50.3.3/1335 |
| 4006_6003 | I2C Status register (I2C0_S) | 8 | R/W | 80h | 50.3.4/1336 |
| 4006_6004 | I2C Data I/O register (I2C0_D) | 8 | R/W | 00h | 50.3.5/1338 |
| 4006_6005 | I2C Control Register 2 (I2C0_C2) | 8 | R/W | 00h | 50.3.6/1339 |
| 4006_6006 | I2C Programmable Input Glitch Filter register (I2C0_FLT) | 8 | R/W | 00h | 50.3.7/1340 |
| 4006_6007 | I2C Range Address register (I2C0_RA) | 8 | R/W | 00h | 50.3.8/1341 |
| 4006_6008 | I2C SMBus Control and Status register (I2C0_SMB) | 8 | R/W | 00h | 50.3.9/1341 |
| 4006_6009 | I2C Address Register 2 (I2C0_A2) | 8 | R/W | C2h | 50.3.10/ 1343 |
| 4006_600A | I2C SCL Low Timeout Register High (I2C0_SLTH) | 8 | R/W | 00h | 50.3.11/ 1343 |
| 4006_600B | I2C SCL Low Timeout Register Low (I2C0_SLTL) | 8 | R/W | 00h | 50.3.12/ 1344 |
| 4006_7000 | I2C Address Register 1 (I2C1_A1) | 8 | R/W | 00h | 50.3.1/1333 |
| 4006_7001 | I2C Frequency Divider register (I2C1_F) | 8 | R/W | 00h | 50.3.2/1333 |
| 4006_7002 | I2C Control Register 1 (I2C1_C1) | 8 | R/W | 00h | 50.3.3/1335 |
| 4006_7003 | I2C Status register (I2C1_S) | 8 | R/W | 80h | 50.3.4/1336 |
| 4006_7004 | I2C Data I/O register (I2C1_D) | 8 | R/W | 00h | 50.3.5/1338 |
| 4006_7005 | I2C Control Register 2 (I2C1_C2) | 8 | R/W | 00h | 50.3.6/1339 |
| 4006_7006 | I2C Programmable Input Glitch Filter register (I2C1_FLT) | 8 | R/W | 00h | 50.3.7/1340 |
| 4006_7007 | I2C Range Address register (I2C1_RA) | 8 | R/W | 00h | 50.3.8/1341 |
| 4006_7008 | I2C SMBus Control and Status register (I2C1_SMB) | 8 | R/W | 00h | 50.3.9/1341 |
| 4006_7009 | I2C Address Register 2 (I2C1_A2) | 8 | R/W | C2h | 50.3.10/ 1343 |
| 4006_700A | I2C SCL Low Timeout Register High (I2C1_SLTH) | 8 | R/W | 00h | 50.3.11/ 1343 |
| 4006_700B | I2C SCL Low Timeout Register Low (I2C1_SLTL) | 8 | R/W | 00h | 50.3.12/ 1344 |
| 400E_6000 | I2C Address Register 1 (I2C2_A1) | 8 | R/W | 00h | 50.3.1/1333 |
| 400E_6001 | I2C Frequency Divider register (I2C2_F) | 8 | R/W | 00h | 50.3.2/1333 |
| 400E_6002 | I2C Control Register 1 (I2C2_C1) | 8 | R/W | 00h | 50.3.3/1335 |
| 400E_6003 | I2C Status register (I2C2_S) | 8 | R/W | 80h | 50.3.4/1336 |
| 400E_6004 | I2C Data I/O register (I2C2_D) | 8 | R/W | 00h | 50.3.5/1338 |
| 400E_6005 | I2C Control Register 2 (I2C2_C2) | 8 | R/W | 00h | 50.3.6/1339 |
| 400E_6006 | I2C Programmable Input Glitch Filter register (I2C2_FLT) | 8 | R/W | 00h | 50.3.7/1340 |
| 400E_6007 | I2C Range Address register (I2C2_RA) | 8 | R/W | 00h | 50.3.8/1341 |
| 400E_6008 | I2C SMBus Control and Status register (I2C2_SMB) | 8 | R/W | 00h | 50.3.9/1341 |

*Table continues on the next page...*

**I2C memory map (continued)**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 400E_6009 | I2C Address Register 2 (I2C2_A2) | 8 | R/W | C2h | 50.3.10/ 1343 |
| 400E_600A | I2C SCL Low Timeout Register High (I2C2_SLTH) | 8 | R/W | 00h | 50.3.11/ 1343 |
| 400E_600B | I2C SCL Low Timeout Register Low (I2C2_SLTL) | 8 | R/W | 00h | 50.3.12/ 1344 |

## 50.3.1 I2C Address Register 1 (I2Cx_A1)

This register contains the slave address to be used by the I2C module.

Address: Base address + 0h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | AD[7:1] | | | | | | | 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**I2Cx_A1 field descriptions**

| Field | Description |
|---|---|
| 7–1<br>AD[7:1] | Address<br><br>Contains the primary slave address used by the I2C module when it is addressed as a slave. This field is used in the 7-bit address scheme and the lower seven bits in the 10-bit address scheme. |
| 0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 50.3.2 I2C Frequency Divider register (I2Cx_F)

Address: Base address + 1h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | MULT | | ICR | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## I2Cx_F field descriptions

| Field | Description |
|---|---|
| 7–6<br>MULT | The MULT bits define the multiplier factor mul. This factor is used along with the SCL divider to generate the I2C baud rate.<br><br>00 mul = 1<br>01 mul = 2<br>10 mul = 4<br>11 Reserved |
| 5–0<br>ICR | ClockRate<br><br>Prescales the bus clock for bit rate selection. This field and the MULT field determine the I2C baud rate, the SDA hold time, the SCL start hold time, and the SCL stop hold time. For a list of values corresponding to each ICR setting, see I2C divider and hold values.<br><br>The SCL divider multiplied by multiplier factor (mul) determines the I2C baud rate.<br><br>`I2C baud rate = bus speed (Hz)/(mul × SCL divider)`<br><br>The SDA hold time is the delay from the falling edge of SCL (I2C clock) to the changing of SDA (I2C data).<br><br>`SDA hold time = bus period (s) × mul × SDA hold value`<br><br>The SCL start hold time is the delay from the falling edge of SDA (I2C data) while SCL is high (start condition) to the falling edge of SCL (I2C clock).<br><br>`SCL start hold time = bus period (s) × mul × SCL start hold value`<br><br>The SCL stop hold time is the delay from the rising edge of SCL (I2C clock) to the rising edge of SDA (I2C data) while SCL is high (stop condition).<br><br>`SCL stop hold time = bus period (s) × mul × SCL stop hold value`<br><br>For example, if the bus speed is 8 MHz, the following table shows the possible hold time values with different ICR and MULT selections to achieve an $I^2C$ baud rate of 100 kbit/s. |

| MULT | ICR | Hold times (µs) | | |
|---|---|---|---|---|
| | | SDA | SCL Start | SCL Stop |
| 2h | 00h | 3.500 | 3.000 | 5.500 |
| 1h | 07h | 2.500 | 4.000 | 5.250 |
| 1h | 0Bh | 2.250 | 4.000 | 5.250 |
| 0h | 14h | 2.125 | 4.250 | 5.125 |
| 0h | 18h | 1.125 | 4.750 | 5.125 |

## 50.3.3 I2C Control Register 1 (I2Cx_C1)

Address: Base address + 2h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | IICEN | IICIE | MST | TX | TXAK | 0 | WUEN | DMAEN |
| Write | | | | | | RSTA | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### I2Cx_C1 field descriptions

| Field | Description |
|---|---|
| 7<br>IICEN | I2C Enable<br><br>Enables I2C module operation.<br><br>0 Disabled<br>1 Enabled |
| 6<br>IICIE | I2C Interrupt Enable<br><br>Enables I2C interrupt requests.<br><br>0 Disabled<br>1 Enabled |
| 5<br>MST | Master Mode Select<br><br>When the MST bit is changed from a 0 to a 1, a START signal is generated on the bus and master mode is selected. When this bit changes from a 1 to a 0, a STOP signal is generated and the mode of operation changes from master to slave.<br><br>0 Slave mode<br>1 Master mode |
| 4<br>TX | Transmit Mode Select<br><br>Selects the direction of master and slave transfers. In master mode this bit must be set according to the type of transfer required. Therefore, for address cycles, this bit is always set. When addressed as a slave this bit must be set by software according to the SRW bit in the status register.<br><br>0 Receive<br>1 Transmit |
| 3<br>TXAK | Transmit Acknowledge Enable<br><br>Specifies the value driven onto the SDA during data acknowledge cycles for both master and slave receivers. The value of the FACK bit affects NACK/ACK generation.<br><br>**NOTE:** SCL is held low until TXAK is written.<br><br>0 An acknowledge signal is sent to the bus on the following receiving byte (if FACK is cleared) or the current receiving byte (if FACK is set).<br>1 No acknowledge signal is sent to the bus on the following receiving data byte (if FACK is cleared) or the current receiving data byte (if FACK is set). |
| 2<br>RSTA | Repeat START |

*Table continues on the next page...*

### I2Cx_C1 field descriptions (continued)

| Field | Description |
|---|---|
| | Writing a one to this bit generates a repeated START condition provided it is the current master. This bit will always be read as zero. Attempting a repeat at the wrong time results in loss of arbitration. |
| 1<br>WUEN | Wakeup Enable<br><br>The I2C module can wake the MCU from low power mode with no peripheral bus running when slave address matching occurs.<br><br>0 Normal operation. No interrupt generated when address matching in low power mode.<br>1 Enables the wakeup function in low power mode. |
| 0<br>DMAEN | DMA Enable<br><br>The DMAEN bit enables or disables the DMA function.<br><br>0 All DMA signalling disabled.<br>1 DMA transfer is enabled and the following conditions trigger the DMA request:<br>• While FACK = 0, a data byte is received, either address or data is transmitted. (ACK/NACK automatic)<br>• While FACK = 0, the first byte received matches the A1 register or is general call address.<br><br>If any address matching occurs, IAAS and TCF are set. If the direction of transfer is known from master to slave, then it is not required to check the SRW. With this assumption, DMA can also be used in this case. In other cases, if the master reads data from the slave, then it is required to rewrite the C1 register operation. With this assumption, DMA cannot be used.<br><br>When FACK = 1, an address or a data byte is transmitted. |

## 50.3.4 I2C Status register (I2Cx_S)

Address: Base address + 3h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | TCF | IAAS | BUSY | ARBL | RAM | SRW | IICIF | RXAK |
| Write | | | | w1c | | | w1c | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### I2Cx_S field descriptions

| Field | Description |
|---|---|
| 7<br>TCF | Transfer Complete Flag<br><br>This bit sets on the completion of a byte and acknowledge bit transfer. This bit is valid only during or immediately following a transfer to or from the I2C module. The TCF bit is cleared by reading the I2C data register in receive mode or by writing to the I2C data register in transmit mode.<br><br>0 Transfer in progress<br>1 Transfer complete |
| 6<br>IAAS | Addressed As A Slave<br><br>This bit is set by one of the following conditions: |

*Table continues on the next page...*

## I2Cx_S field descriptions (continued)

| Field | Description |
|---|---|
| | • The calling address matches the programmed slave primary address in the A1 register or range address in the RA register (which must be set to a nonzero value).<br>• GCAEN is set and a general call is received.<br>• SIICAEN is set and the calling address matches the second programmed slave address.<br>• ALERTEN is set and an SMBus alert response address is received<br>• RMEN is set and an address is received that is within the range between the values of the A1 and RA registers.<br><br>This bit sets before the ACK bit. The CPU must check the SRW bit and set TX/RX accordingly. Writing the C1 register with any value clears this bit.<br><br>0 Not addressed<br>1 Addressed as a slave |
| 5<br>BUSY | Bus Busy<br><br>Indicates the status of the bus regardless of slave or master mode. This bit is set when a START signal is detected and cleared when a STOP signal is detected.<br><br>0 Bus is idle<br>1 Bus is busy |
| 4<br>ARBL | Arbitration Lost<br><br>This bit is set by hardware when the arbitration procedure is lost. The ARBL bit must be cleared by software, by writing a one to it.<br><br>0 Standard bus operation.<br>1 Loss of arbitration. |
| 3<br>RAM | Range Address Match<br><br>This bit is set to 1 by any of the following conditions:<br>• Any nonzero calling address is received that matches the address in the RA register.<br>• The RMEN bit is set and the calling address is within the range of values of the A1 and RA registers.<br><br>**NOTE:** For the RAM bit to be set to 1 correctly, C1[IICIE] must be set to 1.<br><br>Writing the C1 register with any value clears this bit to 0.<br><br>0 Not addressed<br>1 Addressed as a slave |
| 2<br>SRW | Slave Read/Write<br><br>When addressed as a slave, SRW indicates the value of the R/$\overline{W}$ command bit of the calling address sent to the master.<br><br>0 Slave receive, master writing to slave<br>1 Slave transmit, master reading from slave |
| 1<br>IICIF | Interrupt Flag<br><br>This bit sets when an interrupt is pending. This bit must be cleared by software by writing a 1 to it, such as in the interrupt routine. One of the following events can set this bit:<br>• One byte transfer, including ACK/NACK bit, completes if FACK is 0. An ACK or NACK is sent on the bus by writing 0 or 1 to TXAK after this bit is set in receive mode.<br>• One byte transfer, excluding ACK/NACK bit, completes if FACK is 1. |

*Table continues on the next page...*

**I2Cx_S field descriptions (continued)**

| Field | Description |
|---|---|
| | • Match of slave address to calling address including primary slave address, range slave address, alert response address, second slave address, or general call address.<br>• Arbitration lost<br>• In SMBus mode, any timeouts except SCL and SDA high timeouts<br>• I2C bus stop or start detection if the SSIE bit in the Input Glitch Filter register is 1<br><br>**NOTE:** To clear the I2C bus stop or start detection interrupt: In the interrupt service routine, first clear the STOPF or STARTF bit in the Input Glitch Filter register by writing 1 to it, and then clear the IICIF bit. If this sequence is reversed, the IICIF bit is asserted again.<br><br>0 No interrupt pending<br>1 Interrupt pending |
| 0<br>RXAK | Receive Acknowledge<br><br>0 Acknowledge signal was received after the completion of one byte of data transmission on the bus<br>1 No acknowledge signal detected |

## 50.3.5 I2C Data I/O register (I2Cx_D)

Address: Base address + 4h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | DATA | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**I2Cx_D field descriptions**

| Field | Description |
|---|---|
| 7–0<br>DATA | Data<br><br>In master transmit mode, when data is written to this register, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates receiving of the next byte of data.<br><br>**NOTE:** When making the transition out of master receive mode, switch the I2C mode before reading the Data register to prevent an inadvertent initiation of a master receive data transfer.<br><br>In slave mode, the same functions are available after an address match occurs.<br><br>The C1[TX] bit must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For example, if the I2C module is configured for master transmit but a master receive is desired, reading the Data register does not initiate the receive.<br><br>Reading the Data register returns the last byte received while the I2C module is configured in master receive or slave receive mode. The Data register does not reflect every byte that is transmitted on the I2C bus, and neither can software verify that a byte has been written to the Data register correctly by reading it back.<br><br>In master transmit mode, the first byte of data written to the Data register following assertion of MST (start bit) or assertion of RSTA (repeated start bit) is used for the address transfer and must consist of the calling address (in bits 7-1) concatenated with the required R/$\overline{W}$ bit (in position bit 0). |

## 50.3.6 I2C Control Register 2 (I2Cx_C2)

Address: Base address + 5h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | GCAEN | ADEXT | HDRS | SBRC | RMEN | AD[10:8] | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### I2Cx_C2 field descriptions

| Field | Description |
|---|---|
| 7<br>GCAEN | General Call Address Enable<br><br>Enables general call address.<br><br>0 Disabled<br>1 Enabled |
| 6<br>ADEXT | Address Extension<br><br>Controls the number of bits used for the slave address.<br><br>0 7-bit address scheme<br>1 10-bit address scheme |
| 5<br>HDRS | High Drive Select<br><br>Controls the drive capability of the I2C pads.<br><br>0 Normal drive mode<br>1 High drive mode |
| 4<br>SBRC | Slave Baud Rate Control<br><br>Enables independent slave mode baud rate at maximum frequency, which forces clock stretching on SCL in very fast I2C modes. To a slave, an example of a "very fast" mode is when the master transfers at 40 kbit/s but the slave can capture the master's data at only 10 kbit/s.<br><br>0 The slave baud rate follows the master baud rate and clock stretching may occur<br>1 Slave baud rate is independent of the master baud rate |
| 3<br>RMEN | Range Address Matching Enable<br><br>This bit controls slave address matching for addresses between the values of the A1 and RA registers. When this bit is set, a slave address match occurs for any address greater than the value of the A1 register and less than or equal to the value of the RA register.<br><br>0 Range mode disabled. No address match occurs for an address within the range of values of the A1 and RA registers.<br>1 Range mode enabled. Address matching occurs when a slave receives an address within the range of values of the A1 and RA registers. |
| 2–0<br>AD[10:8] | Slave Address<br><br>Contains the upper three bits of the slave address in the 10-bit address scheme. This field is valid only while the ADEXT bit is set. |

## 50.3.7 I2C Programmable Input Glitch Filter register (I2Cx_FLT)

Address: Base address + 6h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | SHEN | STOPF | SSIE | STARTF | FLT | | | |
| Write | | w1c | | w1c | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### I2Cx_FLT field descriptions

| Field | Description |
|---|---|
| 7<br>SHEN | Stop Hold Enable<br><br>Set this bit to hold off entry to stop mode when any data transmission or reception is occurring.<br>The following scenario explains the holdoff functionality:<br>1. The I2C module is configured for a basic transfer, and the SHEN bit is set to 1.<br>2. A transfer begins.<br>3. The MCU signals the I2C module to enter stop mode.<br>4. The byte currently being transferred, including both address and data, completes its transfer.<br>5. The I2C slave or master acknowledges that the in-transfer byte completed its transfer and acknowledges the request to enter stop mode.<br>6. After receiving the I2C module's acknowledgment of the request to enter stop mode, the MCU determines whether to shut off the I2C module's clock.<br><br>If the SHEN bit is set to 1 and the I2C module is in an idle or disabled state when the MCU signals to enter stop mode, the module immediately acknowledges the request to enter stop mode.<br><br>If SHEN is cleared to 0 and the overall data transmission or reception that was suspended by stop mode entry was incomplete: To resume the overall transmission or reception after the MCU exits stop mode, software must reinitialize the transfer by resending the address of the slave.<br><br>If the I2C Control Register 1's IICIE bit was set to 1 before the MCU entered stop mode, system software will receive the interrupt triggered by the I2C Status Register's TCF bit after the MCU wakes from the stop mode.<br><br>0 Stop holdoff is disabled. The MCU's entry to stop mode is not gated.<br>1 Stop holdoff is enabled. |
| 6<br>STOPF | I2C Bus Stop Detect Flag<br><br>Hardware sets this bit when the I2C bus's stop status is detected. The STOPF bit must be cleared by writing 1 to it.<br><br>0 No stop happens on I2C bus<br>1 Stop detected on I2C bus |
| 5<br>SSIE | I2C Bus Stop or Start Interrupt Enable<br><br>This bit enables the interrupt for I2C bus stop or start detection.<br><br>**NOTE:** To clear the I2C bus stop or start detection interrupt: In the interrupt service routine, first clear the STOPF or STARTF bit by writing 1 to it, and then clear the IICIF bit in the status register. If this sequence is reversed, the IICIF bit is asserted again.<br><br>0 Stop or start detection interrupt is disabled<br>1 Stop or start detection interrupt is enabled |

*Table continues on the next page...*

**I2Cx_FLT field descriptions (continued)**

| Field | Description |
|---|---|
| 4<br>STARTF | I2C Bus Start Detect Flag<br><br>Hardware sets this bit when the I2C bus's start status is detected. The STARTF bit must be cleared by writing 1 to it.<br><br>0 No start happens on I2C bus<br>1 Start detected on I2C bus |
| 3–0<br>FLT | I2C Programmable Filter Factor<br><br>Controls the width of the glitch, in terms of bus clock cycles, that the filter must absorb. For any glitch whose size is less than or equal to this width setting, the filter does not allow the glitch to pass.<br><br>0h No filter/bypass<br>1-Fh Filter glitches up to width of *n* bus clock cycles, where *n*=1-15d |

## 50.3.8 I2C Range Address register (I2Cx_RA)

Address: Base address + 7h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | RAD | | | | | | | 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**I2Cx_RA field descriptions**

| Field | Description |
|---|---|
| 7–1<br>RAD | Range Slave Address<br><br>This field contains the slave address to be used by the I2C module. The field is used in the 7-bit address scheme. Any nonzero write enables this register. This register's use is similar to that of the A1 register, but in addition this register can be considered a maximum boundary in range matching mode. |
| 0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 50.3.9 I2C SMBus Control and Status register (I2Cx_SMB)

**NOTE**

When the SCL and SDA signals are held high for a length of time greater than the high timeout period, the SHTF1 flag sets. Before reaching this threshold, while the system is detecting how long these signals are being held high, a master assumes that the bus is free. However, the SHTF1 bit rises in the bus transmission process with the idle bus state.

## NOTE

When the TCKSEL bit is set, there is no need to monitor the SHTF1 bit because the bus speed is too high to match the protocol of SMBus.

Address: Base address + 8h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | FACK | ALERTEN | SIICAEN | TCKSEL | SLTF | SHTF1 | SHTF2 | SHTF2IE |
| Write | | | | | w1c | | w1c | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### I2Cx_SMB field descriptions

| Field | Description |
|---|---|
| 7<br>FACK | Fast NACK/ACK Enable<br><br>For SMBus packet error checking, the CPU must be able to issue an ACK or NACK according to the result of receiving data byte.<br><br>0 An ACK or NACK is sent on the following receiving data byte<br>1 Writing 0 to TXAK after receiving a data byte generates an ACK. Writing 1 to TXAK after receiving a data byte generates a NACK. |
| 6<br>ALERTEN | SMBus Alert Response Address Enable<br><br>Enables or disables SMBus alert response address matching.<br><br>**NOTE:** After the host responds to a device that used the alert response address, you must use software to put the device's address on the bus. The alert protocol is described in the SMBus specification.<br><br>0 SMBus alert response address matching is disabled<br>1 SMBus alert response address matching is enabled |
| 5<br>SIICAEN | Second I2C Address Enable<br><br>Enables or disables SMBus device default address.<br><br>0 I2C address register 2 matching is disabled<br>1 I2C address register 2 matching is enabled |
| 4<br>TCKSEL | Timeout Counter Clock Select<br><br>Selects the clock source of the timeout counter.<br><br>0 Timeout counter counts at the frequency of the bus clock / 64<br>1 Timeout counter counts at the frequency of the bus clock |
| 3<br>SLTF | SCL Low Timeout Flag<br><br>This bit is set when the SLT register (consisting of the SLTH and SLTL registers) is loaded with a non-zero value (LoValue) and an SCL low timeout occurs. Software clears this bit by writing a logic 1 to it.<br><br>**NOTE:** The low timeout function is disabled when the SLT register's value is zero.<br><br>0 No low timeout occurs<br>1 Low timeout occurs |

*Table continues on the next page...*

**I2Cx_SMB field descriptions (continued)**

| Field | Description |
|---|---|
| 2<br>SHTF1 | SCL High Timeout Flag 1<br><br>This read-only bit sets when SCL and SDA are held high more than clock × LoValue / 512, which indicates the bus is free. This bit is cleared automatically.<br><br>0 No SCL high and SDA high timeout occurs<br>1 SCL high and SDA high timeout occurs |
| 1<br>SHTF2 | SCL High Timeout Flag 2<br><br>This bit sets when SCL is held high and SDA is held low more than clock × LoValue/512. Software clears this bit by writing a 1 to it.<br><br>0 No SCL high and SDA low timeout occurs<br>1 SCL high and SDA low timeout occurs |
| 0<br>SHTF2IE | SHTF2 Interrupt Enable<br><br>Enables SCL high and SDA low timeout interrupt.<br><br>0 SHTF2 interrupt is disabled<br>1 SHTF2 interrupt is enabled |

## 50.3.10 I2C Address Register 2 (I2Cx_A2)

Address: Base address + 9h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | SAD | | | | | | | 0 |
| Reset | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |

**I2Cx_A2 field descriptions**

| Field | Description |
|---|---|
| 7–1<br>SAD | SMBus Address<br><br>Contains the slave address used by the SMBus. This field is used on the device default address or other related addresses. |
| 0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 50.3.11 I2C SCL Low Timeout Register High (I2Cx_SLTH)

Address: Base address + Ah offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | SSLT[15:8] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**I2Cx_SLTH field descriptions**

| Field | Description |
|---|---|
| 7–0<br>SSLT[15:8] | Most significant byte of SCL low timeout value that determines the timeout period of SCL low. |

## 50.3.12 I2C SCL Low Timeout Register Low (I2Cx_SLTL)

Address: Base address + Bh offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | SSLT[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**I2Cx_SLTL field descriptions**

| Field | Description |
|---|---|
| 7–0<br>SSLT[7:0] | Least significant byte of SCL low timeout value that determines the timeout period of SCL low. |

# 50.4 Functional description

This section provides a comprehensive functional description of the I2C module.

## 50.4.1 I2C protocol

The I2C bus system uses a serial data line (SDA) and a serial clock line (SCL) for data transfers. All devices connected to it must have open drain or open collector outputs. A logic AND function is exercised on both lines with external pull-up resistors. The value of these resistors depends on the system.

Normally, a standard instance of communication is composed of four parts:

1. START signal
2. Slave address transmission
3. Data transfer
4. STOP signal

The STOP signal should not be confused with the CPU STOP instruction. The following figure illustrates I2C bus system communication.

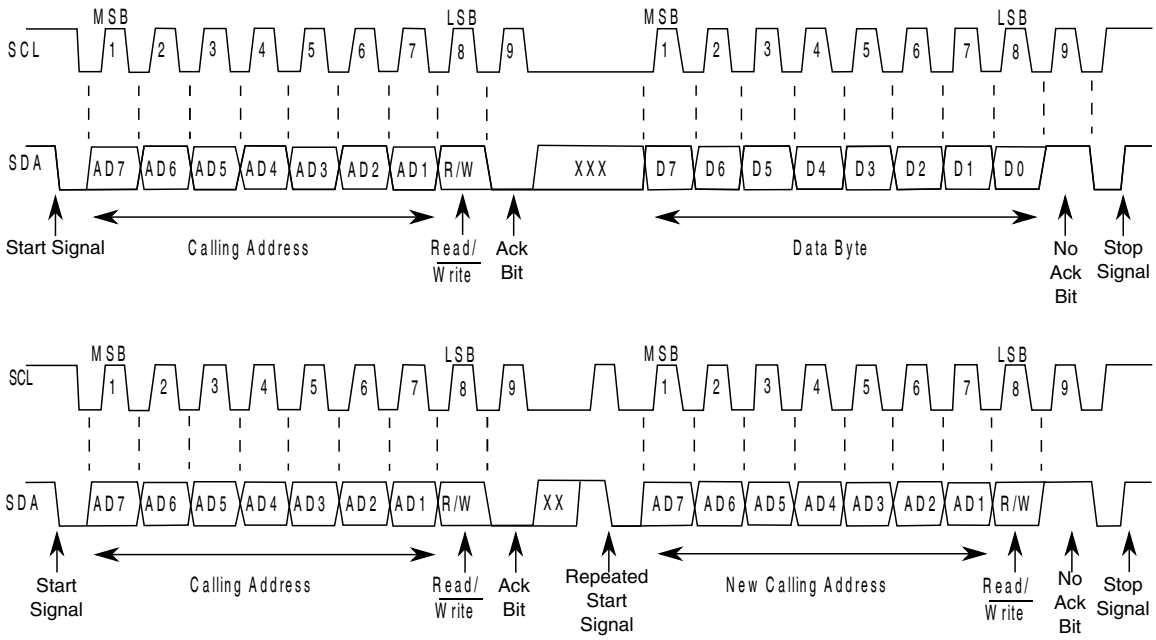



**Figure 50-50. I2C bus transmission signals**

### 50.4.1.1 START signal

The bus is free when no master device is engaging the bus (both SCL and SDA are high). When the bus is free, a master may initiate communication by sending a START signal. A START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer—each data transfer might contain several bytes of data—and brings all slaves out of their idle states.

### 50.4.1.2 Slave address transmission

Immediately after the START signal, the first byte of a data transfer is the slave address transmitted by the master. This address is a 7-bit calling address followed by an R/$\overline{W}$ bit. The R/$\overline{W}$ bit tells the slave the desired direction of data transfer.

- 1 = Read transfer: The slave transmits data to the master
- 0 = Write transfer: The master transmits data to the slave

Only the slave with a calling address that matches the one transmitted by the master responds by sending an acknowledge bit. The slave sends the acknowledge bit by pulling SDA low at the ninth clock.

No two slaves in the system can have the same address. If the I2C module is the master, it must not transmit an address that is equal to its own slave address. The I2C module cannot be master and slave at the same time. However, if arbitration is lost during an address cycle, the I2C module reverts to slave mode and operates correctly even if it is being addressed by another master.

#### 50.4.1.3 Data transfers

When successful slave addressing is achieved, data transfer can proceed on a byte-by-byte basis in the direction specified by the R/$\overline{W}$ bit sent by the calling master.

All transfers that follow an address cycle are referred to as data transfers, even if they carry subaddress information for the slave device.

Each data byte is 8 bits long. Data may be changed only while SCL is low. Data must be held stable while SCL is high. There is one clock pulse on SCL for each data bit, and the MSB is transferred first. Each data byte is followed by a ninth (acknowledge) bit, which is signaled from the receiving device by pulling SDA low at the ninth clock. In summary, one complete data transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master in the ninth bit, the slave must leave SDA high. The master interprets the failed acknowledgement as an unsuccessful data transfer.

If the master receiver does not acknowledge the slave transmitter after a data byte transmission, the slave interprets it as an end to data transfer and releases the SDA line.

In the case of a failed acknowledgement by either the slave or master, the data transfer is aborted and the master does one of two things:

- Relinquishes the bus by generating a STOP signal.
- Commences a new call by generating a repeated START signal.

#### 50.4.1.4 STOP signal

The master can terminate the communication by generating a STOP signal to free the bus. A STOP signal is defined as a low-to-high transition of SDA while SCL is asserted.

The master can generate a STOP signal even if the slave has generated an acknowledgement, at which point the slave must release the bus.

### 50.4.1.5 Repeated START signal

The master may generate a START signal followed by a calling command without generating a STOP signal first. This action is called a repeated START. The master uses a repeated START to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.

### 50.4.1.6 Arbitration procedure

The I2C bus is a true multimaster bus that allows more than one master to be connected on it.

If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock. The bus clock's low period is equal to the longest clock low period, and the high period is equal to the shortest one among the masters.

The relative priority of the contending masters is determined by a data arbitration procedure. A bus master loses arbitration if it transmits logic level 1 while another master transmits logic level 0. The losing masters immediately switch to slave receive mode and stop driving SDA output. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, hardware sets a status bit to indicate the loss of arbitration.

### 50.4.1.7 Clock synchronization

Because wire AND logic is performed on SCL, a high-to-low transition on SCL affects all devices connected on the bus. The devices start counting their low period and, after a device's clock has gone low, that device holds SCL low until the clock reaches its high state. However, the change of low to high in this device clock might not change the state of SCL if another device clock is still within its low period. Therefore, the synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time; see the following diagram. When all applicable devices have counted off their low period, the synchronized clock SCL is released and pulled high. Afterward there is no difference between the device clocks and the state of SCL, and all devices start counting their high periods. The first device to complete its high period pulls SCL low again.

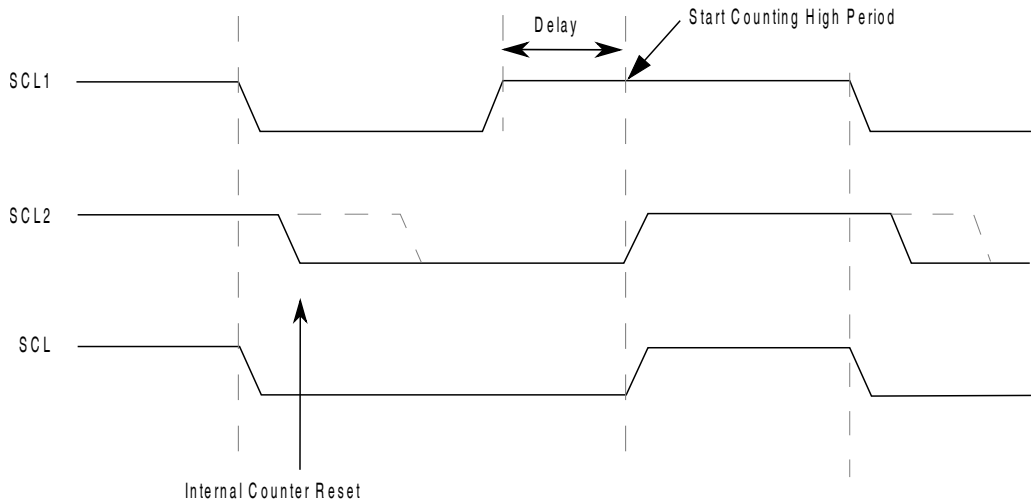



**Figure 50-51. I2C clock synchronization**

### 50.4.1.8 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfers. A slave device may hold SCL low after completing a single byte transfer (9 bits). In this case, it halts the bus clock and forces the master clock into wait states until the slave releases SCL.

### 50.4.1.9 Clock stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master drives SCL low, a slave can drive SCL low for the required period and then release it. If the slave's SCL low period is greater than the master's SCL low period, the resulting SCL bus signal's low period is stretched. In other words, the SCL bus signal's low period is increased to be the same length as the slave's SCL low period.

### 50.4.1.10 I2C divider and hold values

**NOTE**

For some cases on some devices, the SCL divider value may vary by +/-2 or +/-4 when ICR’s value ranges from 00h to 0Fh. These potentially varying SCL divider values are highlighted in the following table. For the actual SCL divider values for your device, see the chip-specific details about the I2C module.

## Table 50-54. I2C divider and hold values

| ICR (hex) | SCL divider | SDA hold value | SCL hold (start) value | SCL hold (stop) value | ICR (hex) | SCL divider (clocks) | SDA hold (clocks) | SCL hold (start) value | SCL hold (stop) value |
|---|---|---|---|---|---|---|---|---|---|
| 00 | 20 | 7 | 6 | 11 | 20 | 160 | 17 | 78 | 81 |
| 01 | 22 | 7 | 7 | 12 | 21 | 192 | 17 | 94 | 97 |
| 02 | 24 | 8 | 8 | 13 | 22 | 224 | 33 | 110 | 113 |
| 03 | 26 | 8 | 9 | 14 | 23 | 256 | 33 | 126 | 129 |
| 04 | 28 | 9 | 10 | 15 | 24 | 288 | 49 | 142 | 145 |
| 05 | 30 | 9 | 11 | 16 | 25 | 320 | 49 | 158 | 161 |
| 06 | 34 | 10 | 13 | 18 | 26 | 384 | 65 | 190 | 193 |
| 07 | 40 | 10 | 16 | 21 | 27 | 480 | 65 | 238 | 241 |
| 08 | 28 | 7 | 10 | 15 | 28 | 320 | 33 | 158 | 161 |
| 09 | 32 | 7 | 12 | 17 | 29 | 384 | 33 | 190 | 193 |
| 0A | 36 | 9 | 14 | 19 | 2A | 448 | 65 | 222 | 225 |
| 0B | 40 | 9 | 16 | 21 | 2B | 512 | 65 | 254 | 257 |
| 0C | 44 | 11 | 18 | 23 | 2C | 576 | 97 | 286 | 289 |
| 0D | 48 | 11 | 20 | 25 | 2D | 640 | 97 | 318 | 321 |
| 0E | 56 | 13 | 24 | 29 | 2E | 768 | 129 | 382 | 385 |
| 0F | 68 | 13 | 30 | 35 | 2F | 960 | 129 | 478 | 481 |
| 10 | 48 | 9 | 18 | 25 | 30 | 640 | 65 | 318 | 321 |
| 11 | 56 | 9 | 22 | 29 | 31 | 768 | 65 | 382 | 385 |
| 12 | 64 | 13 | 26 | 33 | 32 | 896 | 129 | 446 | 449 |
| 13 | 72 | 13 | 30 | 37 | 33 | 1024 | 129 | 510 | 513 |
| 14 | 80 | 17 | 34 | 41 | 34 | 1152 | 193 | 574 | 577 |
| 15 | 88 | 17 | 38 | 45 | 35 | 1280 | 193 | 638 | 641 |
| 16 | 104 | 21 | 46 | 53 | 36 | 1536 | 257 | 766 | 769 |
| 17 | 128 | 21 | 58 | 65 | 37 | 1920 | 257 | 958 | 961 |
| 18 | 80 | 9 | 38 | 41 | 38 | 1280 | 129 | 638 | 641 |
| 19 | 96 | 9 | 46 | 49 | 39 | 1536 | 129 | 766 | 769 |
| 1A | 112 | 17 | 54 | 57 | 3A | 1792 | 257 | 894 | 897 |
| 1B | 128 | 17 | 62 | 65 | 3B | 2048 | 257 | 1022 | 1025 |
| 1C | 144 | 25 | 70 | 73 | 3C | 2304 | 385 | 1150 | 1153 |
| 1D | 160 | 25 | 78 | 81 | 3D | 2560 | 385 | 1278 | 1281 |
| 1E | 192 | 33 | 94 | 97 | 3E | 3072 | 513 | 1534 | 1537 |
| 1F | 240 | 33 | 118 | 121 | 3F | 3840 | 513 | 1918 | 1921 |

## 50.4.2 10-bit address

For 10-bit addressing, 0x11110 is used for the first 5 bits of the first address byte. Various combinations of read/write formats are possible within a transfer that includes 10-bit addressing.

### 50.4.2.1 Master-transmitter addresses a slave-receiver

The transfer direction is not changed. When a 10-bit address follows a START condition, each slave compares the first 7 bits of the first byte of the slave address (11110XX) with its own address and tests whether the eighth bit (R/$\overline{W}$ direction bit) is 0. It is possible that more than one device finds a match and generates an acknowledge (A1). Each slave that finds a match compares the 8 bits of the second byte of the slave address with its own address, but only one slave finds a match and generates an acknowledge (A2). The matching slave remains addressed by the master until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different slave address.

**Table 50-55. Master-transmitter addresses slave-receiver with a 10-bit address**

| S | Slave address first 7 bits 11110 + AD10 + AD9 | R/$\overline{W}$ 0 | A1 | Slave address second byte AD[8:1] | A2 | Data | A | ... | Data | A/A | P |
|---|---|---|---|---|---|---|---|---|---|---|---|

After the master-transmitter has sent the first byte of the 10-bit address, the slave-receiver sees an I2C interrupt. User software must ensure that for this interrupt, the contents of the Data register are ignored and not treated as valid data.

### 50.4.2.2 Master-receiver addresses a slave-transmitter

The transfer direction is changed after the second R/$\overline{W}$ bit. Up to and including acknowledge bit A2, the procedure is the same as that described for a master-transmitter addressing a slave-receiver. After the repeated START condition (Sr), a matching slave remembers that it was addressed before. This slave then checks whether the first seven bits of the first byte of the slave address following Sr are the same as they were after the START condition (S), and it tests whether the eighth (R/$\overline{W}$) bit is 1. If there is a match, the slave considers that it has been addressed as a transmitter and generates acknowledge A3. The slave-transmitter remains addressed until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different slave address.

After a repeated START condition (Sr), all other slave devices also compare the first seven bits of the first byte of the slave address with their own addresses and test the eighth (R/$\overline{W}$) bit. However, none of them are addressed because R/$\overline{W}$ = 1 (for 10-bit devices), or the 11110XX slave address (for 7-bit devices) does not match.

**Table 50-56. Master-receiver addresses a slave-transmitter with a 10-bit address**

| S | Slave address first 7 bits 11110 + AD10 + AD9 | R/$\overline{W}$ 0 | A1 | Slave address second byte AD[8:1] | A2 | Sr | Slave address first 7 bits 11110 + AD10 + AD9 | R/$\overline{W}$ 1 | A3 | Data | A | ... | Data | A | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

After the master-receiver has sent the first byte of the 10-bit address, the slave-transmitter sees an I2C interrupt. User software must ensure that for this interrupt, the contents of the Data register are ignored and not treated as valid data.

## 50.4.3 Address matching

All received addresses can be requested in 7-bit or 10-bit address format.

- AD[7:1] in Address Register 1, which contains the I2C primary slave address, always participates in the address matching process. It provides a 7-bit address.
- If the ADEXT bit is set, AD[10:8] in Control Register 2 participates in the address matching process. It extends the I2C primary slave address to a 10-bit address.

Additional conditions that affect address matching include:

- If the GCAEN bit is set, general call participates the address matching process.
- If the ALERTEN bit is set, alert response participates the address matching process.
- If the SIICAEN bit is set, Address Register 2 participates in the address matching process.
- If the Range Address register is programmed to a nonzero value, the range address itself participates in the address matching process.
- If the RMEN bit is set, any address within the range of values of Address Register 1 and the Range Address register participates in the address matching process. The Range Address register must be programmed to a value greater than the value of Address Register 1.

When the I2C module responds to one of these addresses, it acts as a slave-receiver and the IAAS bit is set after the address cycle. Software must read the Data register after the first byte transfer to determine that the address is matched.

### 50.4.4 System management bus specification

SMBus provides a control bus for system and power management related tasks. A system can use SMBus to pass messages to and from devices instead of tripping individual control lines. Removing the individual control lines reduces pin count. Accepting messages ensures future expandability. With the system management bus, a device can provide manufacturer information, tell the system what its model/part number is, save its state for a suspend event, report different types of errors, accept control parameters, and return its status.

#### 50.4.4.1 Timeouts

The $T_{TIMEOUT,MIN}$ parameter allows a master or slave to conclude that a defective device is holding the clock low indefinitely or a master is intentionally trying to drive devices off the bus. The slave device must release the bus (stop driving the bus and let SCL and SDA float high) when it detects any single clock held low longer than $T_{TIMEOUT,MIN}$. Devices that have detected this condition must reset their communication and be able to receive a new START condition within the timeframe of $T_{TIMEOUT,MAX}$.

SMBus defines a clock low timeout, $T_{TIMEOUT}$, of 35 ms, specifies $T_{LOW:SEXT}$ as the cumulative clock low extend time for a slave device, and specifies $T_{LOW:MEXT}$ as the cumulative clock low extend time for a master device.

##### 50.4.4.1.1 SCL low timeout

If the SCL line is held low by a slave device on the bus, no further communication is possible. Furthermore, the master cannot force the SCL line high to correct the error condition. To solve this problem, the SMBus protocol specifies that devices participating in a transfer must detect any clock cycle held low longer than a timeout value condition. Devices that have detected the timeout condition must reset the communication. When the I2C module is an active master, if it detects that SMBCLK low has exceeded the value of $T_{TIMEOUT,MIN}$, it must generate a stop condition within or after the current data byte in the transfer process. When the I2C module is a slave, if it detects the $T_{TIMEOUT,MIN}$ condition, it resets its communication and is then able to receive a new START condition.

##### 50.4.4.1.2 SCL high timeout

When the I2C module has determined that the SMBCLK and SMBDAT signals have been high for at least $T_{HIGH:MAX}$, it assumes that the bus is idle.

A HIGH timeout occurs after a START condition appears on the bus but before a STOP condition appears on the bus. Any master detecting this scenario can assume the bus is free when either of the following occurs:

- SHTF1 rises.
- The BUSY bit is high and SHTF1 is high.

When the SMBDAT signal is low and the SMBCLK signal is high for a period of time, another kind of timeout occurs. The time period must be defined in software. SHTF2 is used as the flag when the time limit is reached. This flag is also an interrupt resource, so it triggers IICIF.

#### 50.4.4.1.3 CSMBCLK TIMEOUT MEXT and CSMBCLK TIMEOUT SEXT

The following figure illustrates the definition of the timeout intervals $T_{LOW:SEXT}$ and $T_{LOW:MEXT}$. When in master mode, the I2C module must not cumulatively extend its clock cycles for a period greater than $T_{LOW:MEXT}$ within a byte, where each byte is defined as START-to-ACK, ACK-to-ACK, or ACK-to-STOP. When CSMBCLK TIMEOUT MEXT occurs, SMBus MEXT rises and also triggers the SLTF.

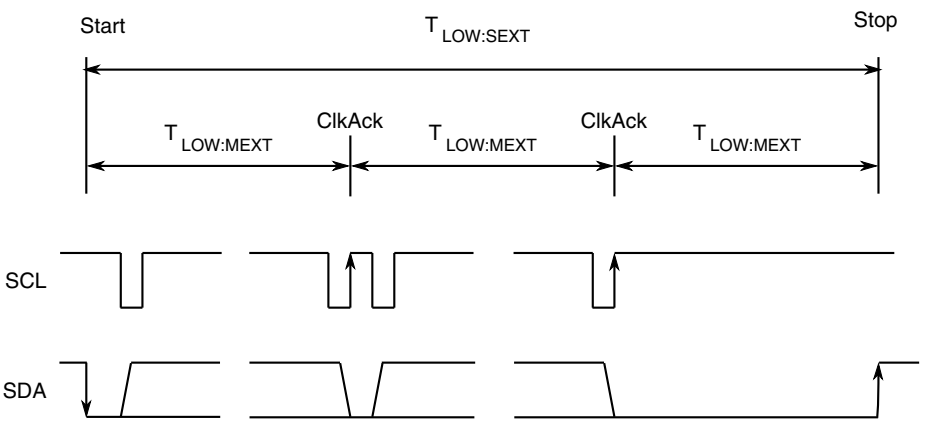



**Figure 50-52. Timeout measurement intervals**

A master is allowed to abort the transaction in progress to any slave that violates the $T_{LOW:SEXT}$ or $T_{TIMEOUT,MIN}$ specifications. To abort the transaction, the master issues a STOP condition at the conclusion of the byte transfer in progress. When a slave, the I2C module must not cumulatively extend its clock cycles for a period greater than $T_{LOW:SEXT}$ during any message from the initial START to the STOP. When CSMBCLK TIMEOUT SEXT occurs, SEXT rises and also triggers SLTF.

**NOTE**

CSMBCLK TIMEOUT SEXT and CSMBCLK TIMEOUT MEXT are optional functions that are implemented in the second step.

#### 50.4.4.2 FAST ACK and NACK

To improve reliability and communication robustness, implementation of packet error checking (PEC) by SMBus devices is optional for SMBus devices but required for devices participating in and only during the address resolution protocol (ARP) process. The PEC is a CRC-8 error checking byte, calculated on all the message bytes. The PEC is appended to the message by the device that supplied the last data byte. If the PEC is present but not correct, a NACK is issued by the receiver. Otherwise an ACK is issued. To calculate the CRC-8 by software, this module can hold the SCL line low after receiving the eighth SCL (8th bit) if this byte is a data byte. So software can determine whether an ACK or NACK should be sent to the bus by setting or clearing the TXAK bit if the FACK (fast ACK/NACK enable) bit is enabled.

SMBus requires a device always to acknowledge its own address, as a mechanism to detect the presence of a removable device (such as a battery or docking station) on the bus. In addition to indicating a slave device busy condition, SMBus uses the NACK mechanism to indicate the reception of an invalid command or invalid data. Because such a condition may occur on the last byte of the transfer, SMBus devices are required to have the ability to generate the not acknowledge after the transfer of each byte and before the completion of the transaction. This requirement is important because SMBus does not provide any other resend signaling. This difference in the use of the NACK signaling has implications on the specific implementation of the SMBus port, especially in devices that handle critical system data such as the SMBus host and the SBS components.

**NOTE**

In the last byte of master receive slave transmit mode, the master must send a NACK to the bus, so FACK must be switched off before the last byte transmits.

### 50.4.5 Resets

The I2C module is disabled after a reset. The I2C module cannot cause a core reset.

### 50.4.6 Interrupts

The I2C module generates an interrupt when any of the events in the following table occur, provided that the IICIE bit is set. The interrupt is driven by the IICIF bit (of the I2C Status Register) and masked with the IICIE bit (of the I2C Control Register 1). The IICIF bit must be cleared (by software) by writing 1 to it in the interrupt routine. The

SMBus timeouts interrupt is driven by SLTF and masked with the IICIE bit. The SLTF bit must be cleared by software by writing 1 to it in the interrupt routine. You can determine the interrupt type by reading the Status Register.

**NOTE**

In master receive mode, the FACK bit must be set to zero before the last byte transfer.

**Table 50-57. Interrupt summary**

| Interrupt source | Status | Flag | Local enable |
|---|---|---|---|
| Complete 1-byte transfer | TCF | IICIF | IICIE |
| Match of received calling address | IAAS | IICIF | IICIE |
| Arbitration lost | ARBL | IICIF | IICIE |
| $I^2C$ bus stop detection | STOPF | IICIF | IICIE & SSIE |
| $I^2C$ bus start detection | STARTF | IICIF | IICIE & SSIE |
| SMBus SCL low timeout | SLTF | IICIF | IICIE |
| SMBus SCL high SDA low timeout | SHTF2 | IICIF | IICIE & SHTF2IE |
| Wakeup from stop or wait mode | IAAS | IICIF | IICIE & WUEN |

### 50.4.6.1 Byte transfer interrupt

The Transfer Complete Flag (TCF) bit is set at the falling edge of the ninth clock to indicate the completion of a byte and acknowledgement transfer. When FACK is enabled, TCF is then set at the falling edge of eighth clock to indicate the completion of byte.

### 50.4.6.2 Address detect interrupt

When the calling address matches the programmed slave address (I2C Address Register) or when the GCAEN bit is set and a general call is received, the IAAS bit in the Status Register is set. The CPU is interrupted, provided the IICIE bit is set. The CPU must check the SRW bit and set its Tx mode accordingly.

### 50.4.6.3 Stop Detect Interrupt

When the stop status is detected on the $I^2C$ bus, the STOPF bit is set to 1. The CPU is interrupted, provided the IICIE and STOPIE bits are both set to 1.

### 50.4.6.4 Exit from low-power/stop modes

The slave receive input detect circuit and address matching feature are still active on low power modes (wait and stop). An asynchronous input matching slave address or general call address brings the CPU out of low power/stop mode if the interrupt is not masked. Therefore, TCF and IAAS both can trigger this interrupt.

### 50.4.6.5 Arbitration lost interrupt

The I2C is a true multimaster bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, the relative priority of the contending masters is determined by a data arbitration procedure. The I2C module asserts the arbitration-lost interrupt when it loses the data arbitration process and the ARBL bit in the Status Register is set.

Arbitration is lost in the following circumstances:

1. SDA is sampled as low when the master drives high during an address or data transmit cycle.
2. SDA is sampled as low when the master drives high during the acknowledge bit of a data receive cycle.
3. A START cycle is attempted when the bus is busy.
4. A repeated START cycle is requested in slave mode.
5. A STOP condition is detected when the master did not request it.

The ARBL bit must be cleared (by software) by writing 1 to it.

### 50.4.6.6 Timeout interrupt in SMBus

When the IICIE bit is set, the I2C module asserts a timeout interrupt (outputs SLTF and SHTF2) upon detection of any of the mentioned timeout conditions, with one exception. The SCL high and SDA high TIMEOUT mechanism must not be used to influence the timeout interrupt output, because this timeout indicates an idle condition on the bus. SHTF1 rises when it matches the SCL high and SDA high TIMEOUT and falls automatically just to indicate the bus status. The SHTF2's timeout period is the same as that of SHTF1, which is short compared to that of SLTF, so another control bit, SHTF2IE, is added to enable or disable it.

## 50.4.7 Programmable input glitch filter

An I2C glitch filter has been added outside legacy I2C modules but within the I2C package. This filter can absorb glitches on the I2C clock and data lines for the I2C module. The width of the glitch to absorb can be specified in terms of the number of (half) bus clock cycles. A single Programmable Input Glitch Filter control register is provided. Effectively, any down-up-down or up-down-up transition on the data line that occurs within the number of clock cycles programmed in this register is ignored by the I2C module. The programmer must specify the size of the glitch (in terms of bus clock cycles) for the filter to absorb and not pass.

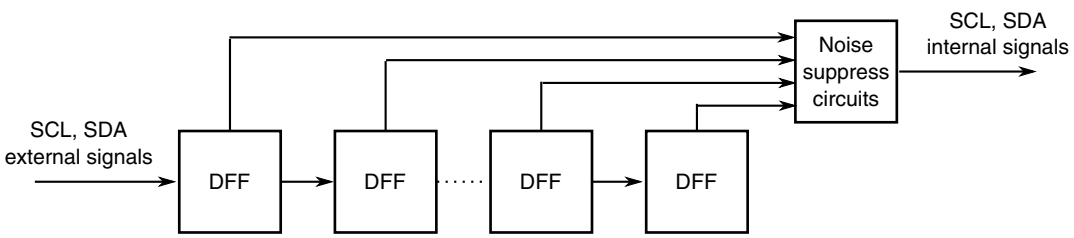



**Figure 50-53. Programmable input glitch filter diagram**

## 50.4.8 Address matching wakeup

When a primary, range, or general call address match occurs when the I2C module is in slave receive mode, the MCU wakes from a low power mode where no peripheral bus is running.

After the address matching IAAS bit is set, an interrupt is sent at the end of address matching to wake the core.

**NOTE**

During the wakeup process, if an external master continues to send data to the slave, the baud rate under Stop mode must be less than 50 kbps. To avoid the slower baud rate under Stop mode, the master can add a short delay in firmware to wait until the wakeup process is complete and then send data.

**NOTE**

Wakeup caused by an address match is not supported for SMBus mode.

### 50.4.9 DMA support

If the DMAEN bit is cleared and the IICIE bit is set, an interrupt condition generates an interrupt request. If the DMAEN bit is set and the IICIE bit is set, an interrupt condition generates a DMA request instead. DMA requests are generated by the transfer complete flag (TCF).

If the DMAEN bit is set, the only arbitration lost is to another I2C module (error), and SCL low timeouts (error) generate CPU interrupts. All other events initiate a DMA transfer.

**NOTE**

Before the last byte of master receive mode, TXAK must be set to send a NACK after the last byte's transfer. Therefore, the DMA must be disabled before the last byte's transfer.

**NOTE**

In 10-bit address mode transmission, the addresses to send occupy 2-3 bytes. During this transfer period, the DMA must be disabled because the C1 register is written to send a repeat start or to change the transfer direction.

## 50.5 Initialization/application information

Module Initialization (Slave)

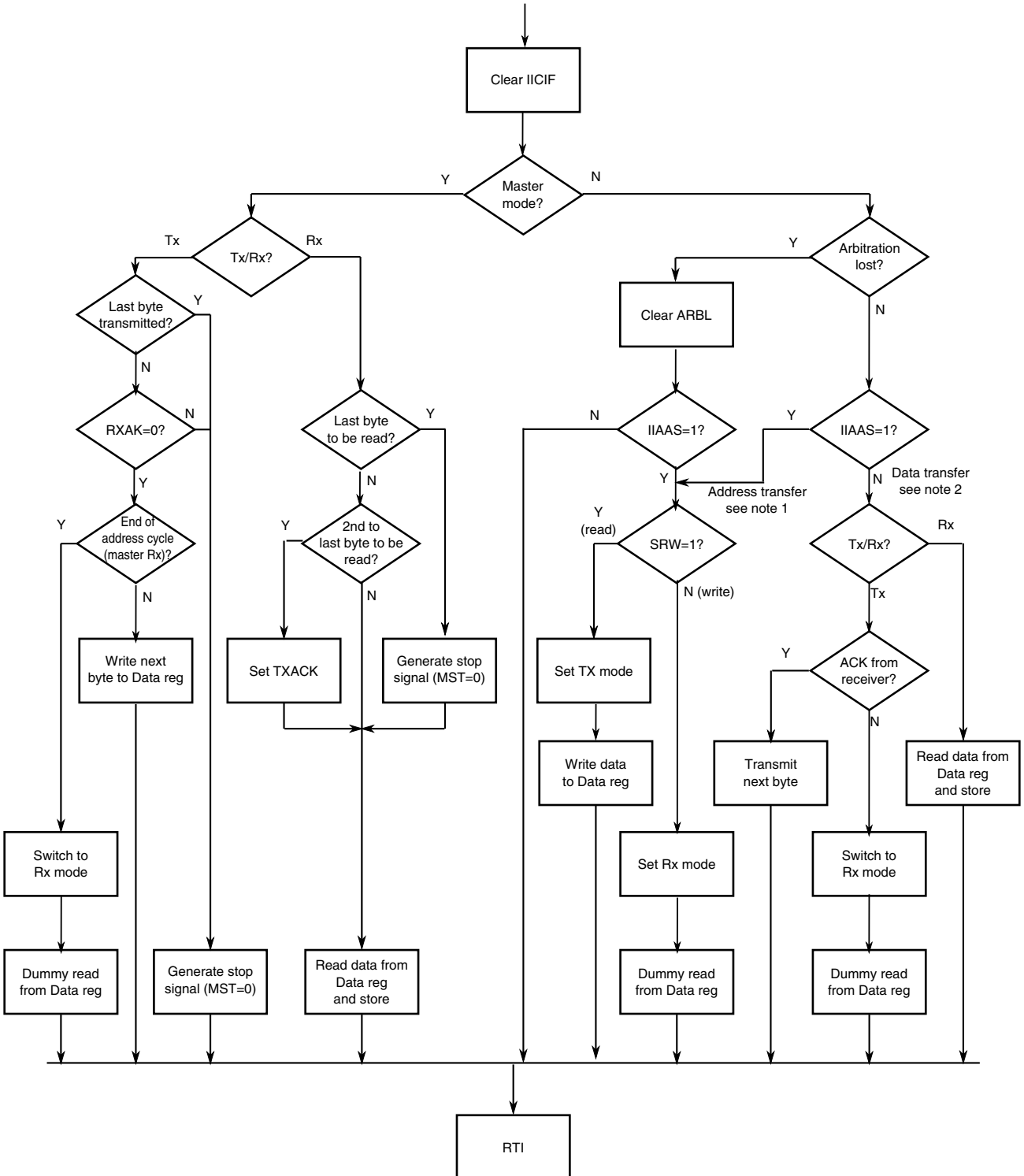
1. Write: Control Register 2
   - to enable or disable general call
   - to select 10-bit or 7-bit addressing mode
2. Write: Address Register 1 to set the slave address
3. Write: Control Register 1 to enable the I2C module and interrupts
4. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
5. Initialize RAM variables used to achieve the routine shown in the following figure

Module Initialization (Master)

1. Write: Frequency Divider register to set the I2C baud rate (see example in description of ICR)
2. Write: Control Register 1 to enable the I2C module and interrupts
3. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
4. Initialize RAM variables used to achieve the routine shown in the following figure
5. Write: Control Register 1 to enable TX

6. Write: Control Register 1 to enable MST (master mode)
7. Write: Data register with the address of the target slave (the LSB of this byte determines whether the communication is master receive or transmit)

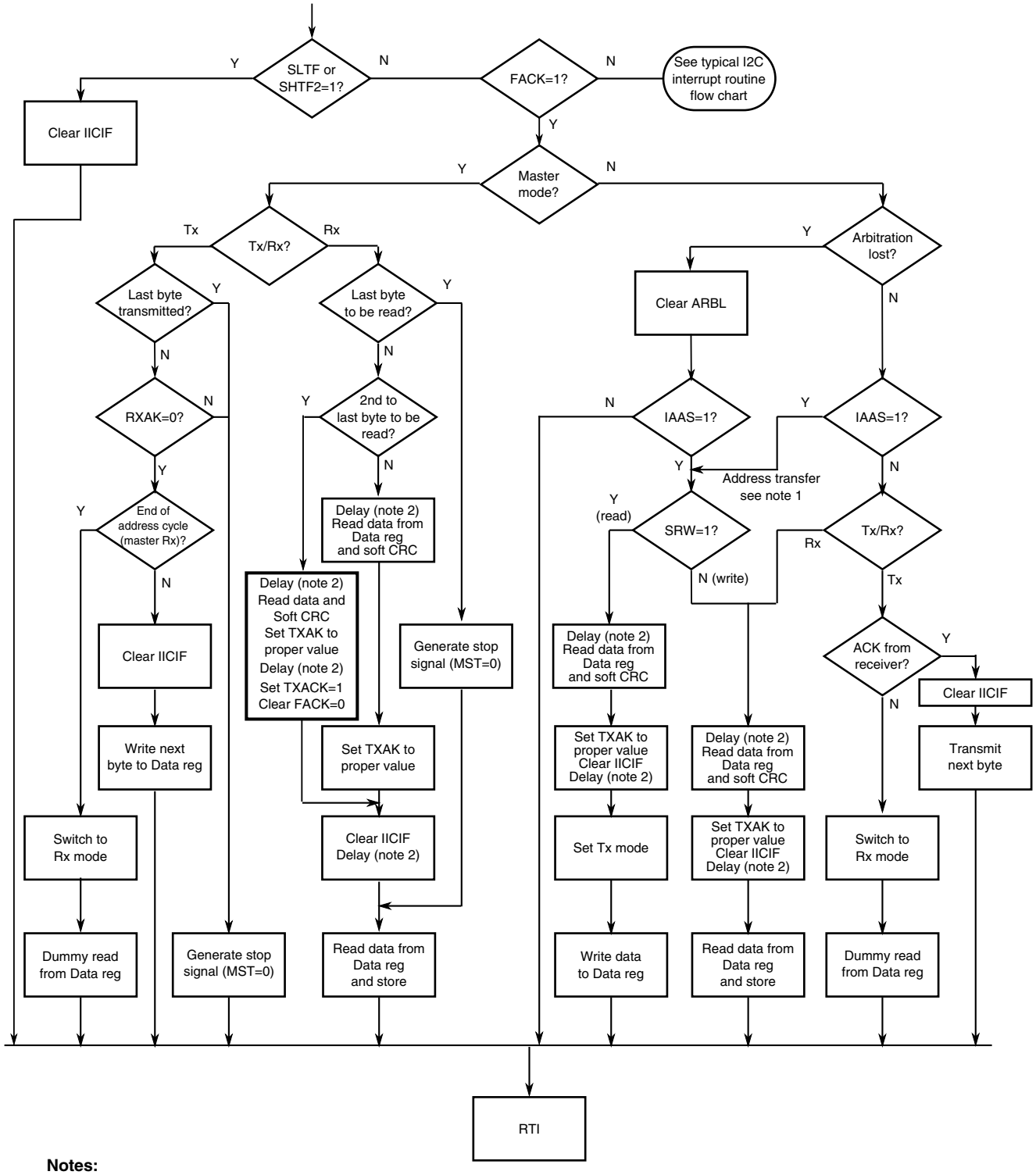
The routine shown in the following figure encompasses both master and slave I2C operations. For slave operation, an incoming I2C message that contains the proper address begins I2C communication. For master operation, communication must be initiated by writing the Data register.




**Notes:**

1. If general call is enabled, check to determine if the received address is a general call address (0x00). If the received address is a general call address, the general call must be handled by user software.
2. When 10-bit addressing addresses a slave, the slave sees an interrupt following the first byte of the extended address. Ensure that for this interrupt, the contents of the Data register are ignored and not treated as a valid data transfer.

**Figure 50-54. Typical I2C interrupt routine**




**Notes:**

1. If general call or SIICAEN is enabled, check to determine if the received address is a general call address (0x00) or an SMBus device default address. In either case, they must be handled by user software.
2. In receive mode, one bit time delay may be needed before the first and second data reading.

**Figure 50-55. Typical I2C SMBus interrupt routine**

# Chapter 51
# Universal Asynchronous Receiver/Transmitter (UART)

## 51.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The UART allows asynchronous serial communication with peripheral devices and CPUs.

### 51.1.1 Features

The UART includes the following features:

- Full-duplex operation
- Standard mark/space non-return-to-zero (NRZ) format
- Selectable IrDA 1.4 return-to-zero-inverted (RZI) format with programmable pulse width
- 13-bit baud rate selection with /32 fractional divide, based on the module clock frequency
- Programmable 8-bit or 9-bit data format
- Separately enabled transmitter and receiver
- Programmable transmitter output polarity
- Programmable receive input polarity
- Upto 14-bit break character transmission.

- 11-bit break character detection option
- Independent FIFO structure for transmit and receive
- Two receiver wakeup methods:
  - Idle line wakeup
  - Address mark wakeup
- Address match feature in the receiver to reduce address mark wakeup ISR overhead
- Ability to select MSB or LSB to be first bit on wire
- Hardware flow control support for request to send (RTS) and clear to send (CTS) signals
- Support for ISO 7816 protocol to interface with SIM cards and smart cards
  - Support for T=0 and T=1 protocols
  - Automatic retransmission of NACK'd packets with programmable retry threshold
  - Support for 11 and 12 ETU transfers
  - Detection of initial packet and automated transfer parameter programming
  - Interrupt-driven operation with seven ISO-7816 specific interrupts:
    - Wait time violated
    - Character wait time violated
    - Block wait time violated
    - Initial frame detected
    - Transmit error threshold exceeded
    - Receive error threshold exceeded
    - Guard time violated
- Interrupt-driven operation with 12 flags, not specific to ISO-7816 support
  - Transmitter data buffer at or below watermark
  - Transmission complete
  - Receiver data buffer at or above watermark
  - Idle receiver input

  - Receiver data buffer overrun
  - Receiver data buffer underflow
  - Transmit data buffer overflow
  - Noise error
  - Framing error
  - Parity error
  - Active edge on receive pin
  - LIN break detect
- Receiver framing error detection
- Hardware parity generation and checking
- 1/16 bit-time noise detection
- DMA interface

## 51.1.2 Modes of operation

The UART functions in the same way in all the normal modes.

It has the following low power modes:

- Wait mode
- Stop mode

### 51.1.2.1 Run mode

This is the normal mode of operation.

### 51.1.2.2 Wait mode

UART operation in the Wait mode depends on the state of the C1[UARTSWAI] field.

- If C1[UARTSWAI] is cleared, and the CPU is in Wait mode, the UART operates normally.
- If C1[UARTSWAI] is set, and the CPU is in Wait mode, the UART clock generation ceases and the UART module enters a power conservation state.

C1[UARTSWAI] does not initiate any power down or power up procedures for the ISO-7816 smartcard interface.

Setting C1[UARTSWAI] does not affect the state of the C2[RE] or C2[TE].

If C1[UARTSWAI] is set, any ongoing transmission or reception stops at the Wait mode entry. The transmission or reception resumes when either an internal or external interrupt brings the CPU out of Wait mode. Bringing the CPU out of Wait mode by reset aborts any ongoing transmission or reception and resets the UART.

#### 51.1.2.3 Stop mode

The UART is inactive during Stop mode for reduced power consumption. The STOP instruction does not affect the UART register states, but the UART module clock is disabled. The UART operation resumes after an external interrupt brings the CPU out of Stop mode. Bringing the CPU out of Stop mode by reset aborts any ongoing transmission or reception and resets the UART. Entering or leaving Stop mode does not initiate any power down or power up procedures for the ISO-7816 smartcard interface.

## 51.2 UART signal descriptions

The UART signals are shown in the following table.

**Table 51-1. UART signal descriptions**

| Signal | Description | I/O |
|---|---|---|
| CTS | Clear to send | I |
| $\overline{\text{RTS}}$ | Request to send | O |
| RXD | Receive data | I |
| TXD | Transmit data | O |

### 51.2.1 Detailed signal descriptions

The detailed signal descriptions of the UART are shown in the following table.

**Table 51-2. UART—Detailed signal descriptions**

| Signal | I/O | Description | |
|---|---|---|---|
| CTS | I | Clear to send. Indicates whether the UART can start transmitting data when flow control is enabled. | |
| | | **State meaning** | Asserted—Data transmission can start.<br>Negated—Data transmission cannot start. |
| | | **Timing** | Assertion—When transmitting device's RTS asserts.<br>Negation—When transmitting device's RTS deasserts. |
| RTS | O | Request to send. When driven by the receiver, indicates whether the UART is ready to receive data. When driven by the transmitter, can enable an external transceiver during transmission. | |
| | | **State meaning** | Asserted—When driven by the receiver, ready to receive data. When driven by the transmitter, enable the external transmitter.<br>Negated—When driven by the receiver, not ready to receive data. When driven by the transmitter, disable the external transmitter. |
| | | **Timing** | Assertion—Can occur at any time; can assert asynchronously to the other input signals.<br>Negation—Can occur at any time; can deassert asynchronously to the other input signals. |
| RXD | I | Receive data. Serial data input to receiver. | |
| | | **State meaning** | Whether RXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings. |
| | | **Timing** | Sampled at a frequency determined by the module clock divided by the baud rate. |
| TXD | O | Transmit data. Serial data output from transmitter. | |
| | | **State meaning** | Whether TXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings. |
| | | **Timing** | Driven at the beginning or within a bit time according to the bit encoding method along with other configuration settings. Otherwise, transmissions are independent of reception timing. |

## 51.3 Memory map and registers

This section provides a detailed description of all memory and registers.

Accessing reserved addresses within the memory map results in a transfer error. None of the contents of the implemented addresses are modified as a result of that access.

Only byte accesses are supported.

## UART memory map

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4006_A000 | UART Baud Rate Registers: High (UART0_BDH) | 8 | R/W | 00h | 51.3.1/1375 |
| 4006_A001 | UART Baud Rate Registers: Low (UART0_BDL) | 8 | R/W | 04h | 51.3.2/1376 |
| 4006_A002 | UART Control Register 1 (UART0_C1) | 8 | R/W | 00h | 51.3.3/1377 |
| 4006_A003 | UART Control Register 2 (UART0_C2) | 8 | R/W | 00h | 51.3.4/1378 |
| 4006_A004 | UART Status Register 1 (UART0_S1) | 8 | R | C0h | 51.3.5/1380 |
| 4006_A005 | UART Status Register 2 (UART0_S2) | 8 | R/W | 00h | 51.3.6/1383 |
| 4006_A006 | UART Control Register 3 (UART0_C3) | 8 | R/W | 00h | 51.3.7/1385 |
| 4006_A007 | UART Data Register (UART0_D) | 8 | R/W | 00h | 51.3.8/1386 |
| 4006_A008 | UART Match Address Registers 1 (UART0_MA1) | 8 | R/W | 00h | 51.3.9/1387 |
| 4006_A009 | UART Match Address Registers 2 (UART0_MA2) | 8 | R/W | 00h | 51.3.10/ 1388 |
| 4006_A00A | UART Control Register 4 (UART0_C4) | 8 | R/W | 00h | 51.3.11/ 1388 |
| 4006_A00B | UART Control Register 5 (UART0_C5) | 8 | R/W | 00h | 51.3.12/ 1389 |
| 4006_A00C | UART Extended Data Register (UART0_ED) | 8 | R | 00h | 51.3.13/ 1390 |
| 4006_A00D | UART Modem Register (UART0_MODEM) | 8 | R/W | 00h | 51.3.14/ 1391 |
| 4006_A00E | UART Infrared Register (UART0_IR) | 8 | R/W | 00h | 51.3.15/ 1392 |
| 4006_A010 | UART FIFO Parameters (UART0_PFIFO) | 8 | R/W | See section | 51.3.16/ 1393 |
| 4006_A011 | UART FIFO Control Register (UART0_CFIFO) | 8 | R/W | 00h | 51.3.17/ 1394 |
| 4006_A012 | UART FIFO Status Register (UART0_SFIFO) | 8 | R/W | C0h | 51.3.18/ 1395 |
| 4006_A013 | UART FIFO Transmit Watermark (UART0_TWFIFO) | 8 | R/W | 00h | 51.3.19/ 1396 |
| 4006_A014 | UART FIFO Transmit Count (UART0_TCFIFO) | 8 | R | 00h | 51.3.20/ 1397 |
| 4006_A015 | UART FIFO Receive Watermark (UART0_RWFIFO) | 8 | R/W | 01h | 51.3.21/ 1397 |
| 4006_A016 | UART FIFO Receive Count (UART0_RCFIFO) | 8 | R | 00h | 51.3.22/ 1398 |
| 4006_A018 | UART 7816 Control Register (UART0_C7816) | 8 | R/W | 00h | 51.3.23/ 1398 |
| 4006_A019 | UART 7816 Interrupt Enable Register (UART0_IE7816) | 8 | R/W | 00h | 51.3.24/ 1400 |
| 4006_A01A | UART 7816 Interrupt Status Register (UART0_IS7816) | 8 | R/W | 00h | 51.3.25/ 1401 |
| 4006_A01B | UART 7816 Wait Parameter Register (UART0_WP7816T0) | 8 | R/W | 0Ah | 51.3.26/ 1402 |

*Table continues on the next page...*

## UART memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4006_A01B | UART 7816 Wait Parameter Register (UART0_WP7816T1) | 8 | R/W | 0Ah | 51.3.27/ 1403 |
| 4006_A01C | UART 7816 Wait N Register (UART0_WN7816) | 8 | R/W | 00h | 51.3.28/ 1403 |
| 4006_A01D | UART 7816 Wait FD Register (UART0_WF7816) | 8 | R/W | 01h | 51.3.29/ 1404 |
| 4006_A01E | UART 7816 Error Threshold Register (UART0_ET7816) | 8 | R/W | 00h | 51.3.30/ 1404 |
| 4006_A01F | UART 7816 Transmit Length Register (UART0_TL7816) | 8 | R/W | 00h | 51.3.31/ 1405 |
| 4006_B000 | UART Baud Rate Registers: High (UART1_BDH) | 8 | R/W | 00h | 51.3.1/1375 |
| 4006_B001 | UART Baud Rate Registers: Low (UART1_BDL) | 8 | R/W | 04h | 51.3.2/1376 |
| 4006_B002 | UART Control Register 1 (UART1_C1) | 8 | R/W | 00h | 51.3.3/1377 |
| 4006_B003 | UART Control Register 2 (UART1_C2) | 8 | R/W | 00h | 51.3.4/1378 |
| 4006_B004 | UART Status Register 1 (UART1_S1) | 8 | R | C0h | 51.3.5/1380 |
| 4006_B005 | UART Status Register 2 (UART1_S2) | 8 | R/W | 00h | 51.3.6/1383 |
| 4006_B006 | UART Control Register 3 (UART1_C3) | 8 | R/W | 00h | 51.3.7/1385 |
| 4006_B007 | UART Data Register (UART1_D) | 8 | R/W | 00h | 51.3.8/1386 |
| 4006_B008 | UART Match Address Registers 1 (UART1_MA1) | 8 | R/W | 00h | 51.3.9/1387 |
| 4006_B009 | UART Match Address Registers 2 (UART1_MA2) | 8 | R/W | 00h | 51.3.10/ 1388 |
| 4006_B00A | UART Control Register 4 (UART1_C4) | 8 | R/W | 00h | 51.3.11/ 1388 |
| 4006_B00B | UART Control Register 5 (UART1_C5) | 8 | R/W | 00h | 51.3.12/ 1389 |
| 4006_B00C | UART Extended Data Register (UART1_ED) | 8 | R | 00h | 51.3.13/ 1390 |
| 4006_B00D | UART Modem Register (UART1_MODEM) | 8 | R/W | 00h | 51.3.14/ 1391 |
| 4006_B00E | UART Infrared Register (UART1_IR) | 8 | R/W | 00h | 51.3.15/ 1392 |
| 4006_B010 | UART FIFO Parameters (UART1_PFIFO) | 8 | R/W | See section | 51.3.16/ 1393 |
| 4006_B011 | UART FIFO Control Register (UART1_CFIFO) | 8 | R/W | 00h | 51.3.17/ 1394 |
| 4006_B012 | UART FIFO Status Register (UART1_SFIFO) | 8 | R/W | C0h | 51.3.18/ 1395 |
| 4006_B013 | UART FIFO Transmit Watermark (UART1_TWFIFO) | 8 | R/W | 00h | 51.3.19/ 1396 |
| 4006_B014 | UART FIFO Transmit Count (UART1_TCFIFO) | 8 | R | 00h | 51.3.20/ 1397 |
| 4006_B015 | UART FIFO Receive Watermark (UART1_RWFIFO) | 8 | R/W | 01h | 51.3.21/ 1397 |

*Table continues on the next page...*

## UART memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4006_B016 | UART FIFO Receive Count (UART1_RCFIFO) | 8 | R | 00h | 51.3.22/ 1398 |
| 4006_B018 | UART 7816 Control Register (UART1_C7816) | 8 | R/W | 00h | 51.3.23/ 1398 |
| 4006_B019 | UART 7816 Interrupt Enable Register (UART1_IE7816) | 8 | R/W | 00h | 51.3.24/ 1400 |
| 4006_B01A | UART 7816 Interrupt Status Register (UART1_IS7816) | 8 | R/W | 00h | 51.3.25/ 1401 |
| 4006_B01B | UART 7816 Wait Parameter Register (UART1_WP7816T0) | 8 | R/W | 0Ah | 51.3.26/ 1402 |
| 4006_B01B | UART 7816 Wait Parameter Register (UART1_WP7816T1) | 8 | R/W | 0Ah | 51.3.27/ 1403 |
| 4006_B01C | UART 7816 Wait N Register (UART1_WN7816) | 8 | R/W | 00h | 51.3.28/ 1403 |
| 4006_B01D | UART 7816 Wait FD Register (UART1_WF7816) | 8 | R/W | 01h | 51.3.29/ 1404 |
| 4006_B01E | UART 7816 Error Threshold Register (UART1_ET7816) | 8 | R/W | 00h | 51.3.30/ 1404 |
| 4006_B01F | UART 7816 Transmit Length Register (UART1_TL7816) | 8 | R/W | 00h | 51.3.31/ 1405 |
| 4006_C000 | UART Baud Rate Registers: High (UART2_BDH) | 8 | R/W | 00h | 51.3.1/1375 |
| 4006_C001 | UART Baud Rate Registers: Low (UART2_BDL) | 8 | R/W | 04h | 51.3.2/1376 |
| 4006_C002 | UART Control Register 1 (UART2_C1) | 8 | R/W | 00h | 51.3.3/1377 |
| 4006_C003 | UART Control Register 2 (UART2_C2) | 8 | R/W | 00h | 51.3.4/1378 |
| 4006_C004 | UART Status Register 1 (UART2_S1) | 8 | R | C0h | 51.3.5/1380 |
| 4006_C005 | UART Status Register 2 (UART2_S2) | 8 | R/W | 00h | 51.3.6/1383 |
| 4006_C006 | UART Control Register 3 (UART2_C3) | 8 | R/W | 00h | 51.3.7/1385 |
| 4006_C007 | UART Data Register (UART2_D) | 8 | R/W | 00h | 51.3.8/1386 |
| 4006_C008 | UART Match Address Registers 1 (UART2_MA1) | 8 | R/W | 00h | 51.3.9/1387 |
| 4006_C009 | UART Match Address Registers 2 (UART2_MA2) | 8 | R/W | 00h | 51.3.10/ 1388 |
| 4006_C00A | UART Control Register 4 (UART2_C4) | 8 | R/W | 00h | 51.3.11/ 1388 |
| 4006_C00B | UART Control Register 5 (UART2_C5) | 8 | R/W | 00h | 51.3.12/ 1389 |
| 4006_C00C | UART Extended Data Register (UART2_ED) | 8 | R | 00h | 51.3.13/ 1390 |
| 4006_C00D | UART Modem Register (UART2_MODEM) | 8 | R/W | 00h | 51.3.14/ 1391 |
| 4006_C00E | UART Infrared Register (UART2_IR) | 8 | R/W | 00h | 51.3.15/ 1392 |
| 4006_C010 | UART FIFO Parameters (UART2_PFIFO) | 8 | R/W | See section | 51.3.16/ 1393 |

*Table continues on the next page...*

## UART memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4006_C011 | UART FIFO Control Register (UART2_CFIFO) | 8 | R/W | 00h | 51.3.17/ 1394 |
| 4006_C012 | UART FIFO Status Register (UART2_SFIFO) | 8 | R/W | C0h | 51.3.18/ 1395 |
| 4006_C013 | UART FIFO Transmit Watermark (UART2_TWFIFO) | 8 | R/W | 00h | 51.3.19/ 1396 |
| 4006_C014 | UART FIFO Transmit Count (UART2_TCFIFO) | 8 | R | 00h | 51.3.20/ 1397 |
| 4006_C015 | UART FIFO Receive Watermark (UART2_RWFIFO) | 8 | R/W | 01h | 51.3.21/ 1397 |
| 4006_C016 | UART FIFO Receive Count (UART2_RCFIFO) | 8 | R | 00h | 51.3.22/ 1398 |
| 4006_C018 | UART 7816 Control Register (UART2_C7816) | 8 | R/W | 00h | 51.3.23/ 1398 |
| 4006_C019 | UART 7816 Interrupt Enable Register (UART2_IE7816) | 8 | R/W | 00h | 51.3.24/ 1400 |
| 4006_C01A | UART 7816 Interrupt Status Register (UART2_IS7816) | 8 | R/W | 00h | 51.3.25/ 1401 |
| 4006_C01B | UART 7816 Wait Parameter Register (UART2_WP7816T0) | 8 | R/W | 0Ah | 51.3.26/ 1402 |
| 4006_C01B | UART 7816 Wait Parameter Register (UART2_WP7816T1) | 8 | R/W | 0Ah | 51.3.27/ 1403 |
| 4006_C01C | UART 7816 Wait N Register (UART2_WN7816) | 8 | R/W | 00h | 51.3.28/ 1403 |
| 4006_C01D | UART 7816 Wait FD Register (UART2_WF7816) | 8 | R/W | 01h | 51.3.29/ 1404 |
| 4006_C01E | UART 7816 Error Threshold Register (UART2_ET7816) | 8 | R/W | 00h | 51.3.30/ 1404 |
| 4006_C01F | UART 7816 Transmit Length Register (UART2_TL7816) | 8 | R/W | 00h | 51.3.31/ 1405 |
| 4006_D000 | UART Baud Rate Registers: High (UART3_BDH) | 8 | R/W | 00h | 51.3.1/1375 |
| 4006_D001 | UART Baud Rate Registers: Low (UART3_BDL) | 8 | R/W | 04h | 51.3.2/1376 |
| 4006_D002 | UART Control Register 1 (UART3_C1) | 8 | R/W | 00h | 51.3.3/1377 |
| 4006_D003 | UART Control Register 2 (UART3_C2) | 8 | R/W | 00h | 51.3.4/1378 |
| 4006_D004 | UART Status Register 1 (UART3_S1) | 8 | R | C0h | 51.3.5/1380 |
| 4006_D005 | UART Status Register 2 (UART3_S2) | 8 | R/W | 00h | 51.3.6/1383 |
| 4006_D006 | UART Control Register 3 (UART3_C3) | 8 | R/W | 00h | 51.3.7/1385 |
| 4006_D007 | UART Data Register (UART3_D) | 8 | R/W | 00h | 51.3.8/1386 |
| 4006_D008 | UART Match Address Registers 1 (UART3_MA1) | 8 | R/W | 00h | 51.3.9/1387 |
| 4006_D009 | UART Match Address Registers 2 (UART3_MA2) | 8 | R/W | 00h | 51.3.10/ 1388 |
| 4006_D00A | UART Control Register 4 (UART3_C4) | 8 | R/W | 00h | 51.3.11/ 1388 |

*Table continues on the next page...*

## UART memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4006_D00B | UART Control Register 5 (UART3_C5) | 8 | R/W | 00h | 51.3.12/ 1389 |
| 4006_D00C | UART Extended Data Register (UART3_ED) | 8 | R | 00h | 51.3.13/ 1390 |
| 4006_D00D | UART Modem Register (UART3_MODEM) | 8 | R/W | 00h | 51.3.14/ 1391 |
| 4006_D00E | UART Infrared Register (UART3_IR) | 8 | R/W | 00h | 51.3.15/ 1392 |
| 4006_D010 | UART FIFO Parameters (UART3_PFIFO) | 8 | R/W | See section | 51.3.16/ 1393 |
| 4006_D011 | UART FIFO Control Register (UART3_CFIFO) | 8 | R/W | 00h | 51.3.17/ 1394 |
| 4006_D012 | UART FIFO Status Register (UART3_SFIFO) | 8 | R/W | C0h | 51.3.18/ 1395 |
| 4006_D013 | UART FIFO Transmit Watermark (UART3_TWFIFO) | 8 | R/W | 00h | 51.3.19/ 1396 |
| 4006_D014 | UART FIFO Transmit Count (UART3_TCFIFO) | 8 | R | 00h | 51.3.20/ 1397 |
| 4006_D015 | UART FIFO Receive Watermark (UART3_RWFIFO) | 8 | R/W | 01h | 51.3.21/ 1397 |
| 4006_D016 | UART FIFO Receive Count (UART3_RCFIFO) | 8 | R | 00h | 51.3.22/ 1398 |
| 4006_D018 | UART 7816 Control Register (UART3_C7816) | 8 | R/W | 00h | 51.3.23/ 1398 |
| 4006_D019 | UART 7816 Interrupt Enable Register (UART3_IE7816) | 8 | R/W | 00h | 51.3.24/ 1400 |
| 4006_D01A | UART 7816 Interrupt Status Register (UART3_IS7816) | 8 | R/W | 00h | 51.3.25/ 1401 |
| 4006_D01B | UART 7816 Wait Parameter Register (UART3_WP7816T0) | 8 | R/W | 0Ah | 51.3.26/ 1402 |
| 4006_D01B | UART 7816 Wait Parameter Register (UART3_WP7816T1) | 8 | R/W | 0Ah | 51.3.27/ 1403 |
| 4006_D01C | UART 7816 Wait N Register (UART3_WN7816) | 8 | R/W | 00h | 51.3.28/ 1403 |
| 4006_D01D | UART 7816 Wait FD Register (UART3_WF7816) | 8 | R/W | 01h | 51.3.29/ 1404 |
| 4006_D01E | UART 7816 Error Threshold Register (UART3_ET7816) | 8 | R/W | 00h | 51.3.30/ 1404 |
| 4006_D01F | UART 7816 Transmit Length Register (UART3_TL7816) | 8 | R/W | 00h | 51.3.31/ 1405 |
| 400E_A000 | UART Baud Rate Registers: High (UART4_BDH) | 8 | R/W | 00h | 51.3.1/1375 |
| 400E_A001 | UART Baud Rate Registers: Low (UART4_BDL) | 8 | R/W | 04h | 51.3.2/1376 |
| 400E_A002 | UART Control Register 1 (UART4_C1) | 8 | R/W | 00h | 51.3.3/1377 |
| 400E_A003 | UART Control Register 2 (UART4_C2) | 8 | R/W | 00h | 51.3.4/1378 |

*Table continues on the next page...*

## UART memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 400E_A004 | UART Status Register 1 (UART4_S1) | 8 | R | C0h | 51.3.5/1380 |
| 400E_A005 | UART Status Register 2 (UART4_S2) | 8 | R/W | 00h | 51.3.6/1383 |
| 400E_A006 | UART Control Register 3 (UART4_C3) | 8 | R/W | 00h | 51.3.7/1385 |
| 400E_A007 | UART Data Register (UART4_D) | 8 | R/W | 00h | 51.3.8/1386 |
| 400E_A008 | UART Match Address Registers 1 (UART4_MA1) | 8 | R/W | 00h | 51.3.9/1387 |
| 400E_A009 | UART Match Address Registers 2 (UART4_MA2) | 8 | R/W | 00h | 51.3.10/ 1388 |
| 400E_A00A | UART Control Register 4 (UART4_C4) | 8 | R/W | 00h | 51.3.11/ 1388 |
| 400E_A00B | UART Control Register 5 (UART4_C5) | 8 | R/W | 00h | 51.3.12/ 1389 |
| 400E_A00C | UART Extended Data Register (UART4_ED) | 8 | R | 00h | 51.3.13/ 1390 |
| 400E_A00D | UART Modem Register (UART4_MODEM) | 8 | R/W | 00h | 51.3.14/ 1391 |
| 400E_A00E | UART Infrared Register (UART4_IR) | 8 | R/W | 00h | 51.3.15/ 1392 |
| 400E_A010 | UART FIFO Parameters (UART4_PFIFO) | 8 | R/W | See section | 51.3.16/ 1393 |
| 400E_A011 | UART FIFO Control Register (UART4_CFIFO) | 8 | R/W | 00h | 51.3.17/ 1394 |
| 400E_A012 | UART FIFO Status Register (UART4_SFIFO) | 8 | R/W | C0h | 51.3.18/ 1395 |
| 400E_A013 | UART FIFO Transmit Watermark (UART4_TWFIFO) | 8 | R/W | 00h | 51.3.19/ 1396 |
| 400E_A014 | UART FIFO Transmit Count (UART4_TCFIFO) | 8 | R | 00h | 51.3.20/ 1397 |
| 400E_A015 | UART FIFO Receive Watermark (UART4_RWFIFO) | 8 | R/W | 01h | 51.3.21/ 1397 |
| 400E_A016 | UART FIFO Receive Count (UART4_RCFIFO) | 8 | R | 00h | 51.3.22/ 1398 |
| 400E_A018 | UART 7816 Control Register (UART4_C7816) | 8 | R/W | 00h | 51.3.23/ 1398 |
| 400E_A019 | UART 7816 Interrupt Enable Register (UART4_IE7816) | 8 | R/W | 00h | 51.3.24/ 1400 |
| 400E_A01A | UART 7816 Interrupt Status Register (UART4_IS7816) | 8 | R/W | 00h | 51.3.25/ 1401 |
| 400E_A01B | UART 7816 Wait Parameter Register (UART4_WP7816T0) | 8 | R/W | 0Ah | 51.3.26/ 1402 |
| 400E_A01B | UART 7816 Wait Parameter Register (UART4_WP7816T1) | 8 | R/W | 0Ah | 51.3.27/ 1403 |
| 400E_A01C | UART 7816 Wait N Register (UART4_WN7816) | 8 | R/W | 00h | 51.3.28/ 1403 |

*Table continues on the next page...*

## UART memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 400E_A01D | UART 7816 Wait FD Register (UART4_WF7816) | 8 | R/W | 01h | 51.3.29/ 1404 |
| 400E_A01E | UART 7816 Error Threshold Register (UART4_ET7816) | 8 | R/W | 00h | 51.3.30/ 1404 |
| 400E_A01F | UART 7816 Transmit Length Register (UART4_TL7816) | 8 | R/W | 00h | 51.3.31/ 1405 |
| 400E_B000 | UART Baud Rate Registers: High (UART5_BDH) | 8 | R/W | 00h | 51.3.1/1375 |
| 400E_B001 | UART Baud Rate Registers: Low (UART5_BDL) | 8 | R/W | 04h | 51.3.2/1376 |
| 400E_B002 | UART Control Register 1 (UART5_C1) | 8 | R/W | 00h | 51.3.3/1377 |
| 400E_B003 | UART Control Register 2 (UART5_C2) | 8 | R/W | 00h | 51.3.4/1378 |
| 400E_B004 | UART Status Register 1 (UART5_S1) | 8 | R | C0h | 51.3.5/1380 |
| 400E_B005 | UART Status Register 2 (UART5_S2) | 8 | R/W | 00h | 51.3.6/1383 |
| 400E_B006 | UART Control Register 3 (UART5_C3) | 8 | R/W | 00h | 51.3.7/1385 |
| 400E_B007 | UART Data Register (UART5_D) | 8 | R/W | 00h | 51.3.8/1386 |
| 400E_B008 | UART Match Address Registers 1 (UART5_MA1) | 8 | R/W | 00h | 51.3.9/1387 |
| 400E_B009 | UART Match Address Registers 2 (UART5_MA2) | 8 | R/W | 00h | 51.3.10/ 1388 |
| 400E_B00A | UART Control Register 4 (UART5_C4) | 8 | R/W | 00h | 51.3.11/ 1388 |
| 400E_B00B | UART Control Register 5 (UART5_C5) | 8 | R/W | 00h | 51.3.12/ 1389 |
| 400E_B00C | UART Extended Data Register (UART5_ED) | 8 | R | 00h | 51.3.13/ 1390 |
| 400E_B00D | UART Modem Register (UART5_MODEM) | 8 | R/W | 00h | 51.3.14/ 1391 |
| 400E_B00E | UART Infrared Register (UART5_IR) | 8 | R/W | 00h | 51.3.15/ 1392 |
| 400E_B010 | UART FIFO Parameters (UART5_PFIFO) | 8 | R/W | See section | 51.3.16/ 1393 |
| 400E_B011 | UART FIFO Control Register (UART5_CFIFO) | 8 | R/W | 00h | 51.3.17/ 1394 |
| 400E_B012 | UART FIFO Status Register (UART5_SFIFO) | 8 | R/W | C0h | 51.3.18/ 1395 |
| 400E_B013 | UART FIFO Transmit Watermark (UART5_TWFIFO) | 8 | R/W | 00h | 51.3.19/ 1396 |
| 400E_B014 | UART FIFO Transmit Count (UART5_TCFIFO) | 8 | R | 00h | 51.3.20/ 1397 |
| 400E_B015 | UART FIFO Receive Watermark (UART5_RWFIFO) | 8 | R/W | 01h | 51.3.21/ 1397 |
| 400E_B016 | UART FIFO Receive Count (UART5_RCFIFO) | 8 | R | 00h | 51.3.22/ 1398 |
| 400E_B018 | UART 7816 Control Register (UART5_C7816) | 8 | R/W | 00h | 51.3.23/ 1398 |

*Table continues on the next page...*

**UART memory map (continued)**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 400E_B019 | UART 7816 Interrupt Enable Register (UART5_IE7816) | 8 | R/W | 00h | 51.3.24/ 1400 |
| 400E_B01A | UART 7816 Interrupt Status Register (UART5_IS7816) | 8 | R/W | 00h | 51.3.25/ 1401 |
| 400E_B01B | UART 7816 Wait Parameter Register (UART5_WP7816T0) | 8 | R/W | 0Ah | 51.3.26/ 1402 |
| 400E_B01B | UART 7816 Wait Parameter Register (UART5_WP7816T1) | 8 | R/W | 0Ah | 51.3.27/ 1403 |
| 400E_B01C | UART 7816 Wait N Register (UART5_WN7816) | 8 | R/W | 00h | 51.3.28/ 1403 |
| 400E_B01D | UART 7816 Wait FD Register (UART5_WF7816) | 8 | R/W | 01h | 51.3.29/ 1404 |
| 400E_B01E | UART 7816 Error Threshold Register (UART5_ET7816) | 8 | R/W | 00h | 51.3.30/ 1404 |
| 400E_B01F | UART 7816 Transmit Length Register (UART5_TL7816) | 8 | R/W | 00h | 51.3.31/ 1405 |

## 51.3.1 UART Baud Rate Registers: High (UARTx_BDH)

This register, along with the BDL register, controls the prescale divisor for UART baud rate generation. To update the 13-bit baud rate setting (SBR[12:0]), first write to BDH to buffer the high half of the new value and then write to BDL. The working value in BDH does not change until BDL is written.

BDL is reset to a nonzero value, but after reset, the baud rate generator remains disabled until the first time the receiver or transmitter is enabled, that is, when C2[RE] or C2[TE] is set.

Address: Base address + 0h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | LBKDIE | RXEDGIE | 0 | SBR | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_BDH field descriptions**

| Field | Description |
|---|---|
| 7<br>LBKDIE | LIN Break Detect Interrupt Enable<br><br>Enables the LIN break detect flag, LBKDIF, to generate interrupt requests based on the state of LBKDDMAS. |

*Table continues on the next page...*

**UARTx_BDH field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 LBKDIF interrupt requests disabled.<br>1 LBKDIF interrupt requests enabled. |
| 6<br>RXEDGIE | RxD Input Active Edge Interrupt Enable<br><br>Enables the receive input active edge, RXEDGIF, to generate interrupt requests.<br><br>0 Hardware interrupts from RXEDGIF disabled using polling.<br>1 RXEDGIF interrupt request enabled. |
| 5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4–0<br>SBR | UART Baud Rate Bits<br><br>The baud rate for the UART is determined by the 13 SBR fields. See Baud rate generation for details.<br><br>**NOTE:** • The baud rate generator is disabled until C2[TE] or C2[RE] is set for the first time after reset.The baud rate generator is disabled when SBR = 0.<br>• Writing to BDH has no effect without writing to BDL, because writing to BDH puts the data in a temporary location until BDL is written. |

## 51.3.2 UART Baud Rate Registers: Low (UARTx_BDL)

This register, along with the BDH register, controls the prescale divisor for UART baud rate generation. To update the 13-bit baud rate setting, SBR[12:0], first write to BDH to buffer the high half of the new value and then write to BDL. The working value in BDH does not change until BDL is written. BDL is reset to a nonzero value, but after reset, the baud rate generator remains disabled until the first time the receiver or transmitter is enabled, that is, when C2[RE] or C2[TE] is set.

Address: Base address + 1h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | SBR | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**UARTx_BDL field descriptions**

| Field | Description |
|---|---|
| 7–0<br>SBR | UART Baud Rate Bits<br><br>The baud rate for the UART is determined by the 13 SBR fields. See Baud rate generation for details.<br><br>**NOTE:** • The baud rate generator is disabled until C2[TE] or C2[RE] is set for the first time after reset.The baud rate generator is disabled when SBR = 0.<br>• Writing to BDH has no effect without writing to BDL, because writing to BDH puts the data in a temporary location until BDL is written.<br>• When the 1/32 narrow pulse width is selected for infrared (IrDA), the baud rate fields must be even, the least significant bit is 0. See MODEM register for more details. |

## 51.3.3 UART Control Register 1 (UARTx_C1)

This read/write register controls various optional features of the UART system.

Address: Base address + 2h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | LOOPS | UARTSWAI | RSRC | M | WAKE | ILT | PE | PT |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_C1 field descriptions**

| Field | Description |
|---|---|
| 7<br>LOOPS | Loop Mode Select<br><br>When LOOPS is set, the RxD pin is disconnected from the UART and the transmitter output is internally connected to the receiver input. The transmitter and the receiver must be enabled to use the loop function.<br><br>0 Normal operation.<br>1 Loop mode where transmitter output is internally connected to receiver input. The receiver input is determined by RSRC. |
| 6<br>UARTSWAI | UART Stops in Wait Mode<br><br>0 UART clock continues to run in Wait mode.<br>1 UART clock freezes while CPU is in Wait mode. |
| 5<br>RSRC | Receiver Source Select<br><br>This field has no meaning or effect unless the LOOPS field is set. When LOOPS is set, the RSRC field determines the source for the receiver shift register input.<br><br>0 Selects internal loop back mode. The receiver input is internally connected to transmitter output.<br>1 Single wire UART mode where the receiver input is connected to the transmit pin input signal. |
| 4<br>M | 9-bit or 8-bit Mode Select<br><br>This field must be set when C7816[ISO_7816E] is set/enabled.<br><br>0 Normal—start + 8 data bits (MSB/LSB first as determined by MSBF) + stop.<br>1 Use—start + 9 data bits (MSB/LSB first as determined by MSBF) + stop. |
| 3<br>WAKE | Receiver Wakeup Method Select<br><br>Determines which condition wakes the UART:<br>• Address mark in the most significant bit position of a received data character, or<br>• An idle condition on the receive pin input signal.<br><br>0 Idle line wakeup.<br>1 Address mark wakeup. |
| 2<br>ILT | Idle Line Type Select<br><br>Determines when the receiver starts counting logic 1s as idle character bits. The count begins either after a valid start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit can cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions. |

*Table continues on the next page...*

**UARTx_C1 field descriptions (continued)**

| Field | Description |
|---|---|
| | **NOTE:** • In case the UART is programmed with ILT = 1, a logic of 1'b0 is automatically shifted after a received stop bit, therefore resetting the idle count.<br>• In case the UART is programmed for IDLE line wakeup (RWU = 1 and WAKE = 0), ILT has no effect on when the receiver starts counting logic 1s as idle character bits. In idle line wakeup, an idle character is recognized at anytime the receiver sees 10, 11, or 12 1s depending on the M, PE, and C4[M10] fields.<br><br>0 Idle character bit count starts after start bit.<br>1 Idle character bit count starts after stop bit. |
| 1<br>PE | Parity Enable<br><br>Enables the parity function. When parity is enabled, parity function inserts a parity bit in the bit position immediately preceding the stop bit. This field must be set when C7816[ISO_7816E] is set/enabled.<br><br>0 Parity function disabled.<br>1 Parity function enabled. |
| 0<br>PT | Parity Type<br><br>Determines whether the UART generates and checks for even parity or odd parity. With even parity, an even number of 1s clears the parity bit and an odd number of 1s sets the parity bit. With odd parity, an odd number of 1s clears the parity bit and an even number of 1s sets the parity bit. This field must be cleared when C7816[ISO_7816E] is set/enabled.<br><br>0 Even parity.<br>1 Odd parity. |

## 51.3.4 UART Control Register 2 (UARTx_C2)

This register can be read or written at any time.

Address: Base address + 3h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_C2 field descriptions**

| Field | Description |
|---|---|
| 7<br>TIE | Transmitter Interrupt or DMA Transfer Enable.<br><br>Enables S1[TDRE] to generate interrupt requests or DMA transfer requests, based on the state of C5[TDMAS].<br><br>**NOTE:** If C2[TIE] and C5[TDMAS] are both set, then TCIE must be cleared, and D[D] must not be written unless servicing a DMA request.<br><br>0 TDRE interrupt and DMA transfer requests disabled.<br>1 TDRE interrupt or DMA transfer requests enabled. |

*Table continues on the next page...*

## UARTx_C2 field descriptions (continued)

| Field | Description |
|---|---|
| 6<br>TCIE | Transmission Complete Interrupt Enable<br><br>Enables the transmission complete flag, S1[TC], to generate interrupt requests .<br><br>0 TC interrupt requests disabled.<br>1 TC interrupt requests enabled. |
| 5<br>RIE | Receiver Full Interrupt or DMA Transfer Enable<br><br>Enables S1[RDRF] to generate interrupt requests or DMA transfer requests, based on the state of C5[RDMAS].<br><br>0 RDRF interrupt and DMA transfer requests disabled.<br>1 RDRF interrupt or DMA transfer requests enabled. |
| 4<br>ILIE | Idle Line Interrupt Enable<br><br>Enables the idle line flag, S1[IDLE], to generate interrupt requests<br><br>0 IDLE interrupt requests disabled.<br>1 IDLE interrupt requests enabled. |
| 3<br>TE | Transmitter Enable<br><br>Enables the UART transmitter. TE can be used to queue an idle preamble by clearing and then setting TE. When C7816[ISO_7816E] is set/enabled and C7816[TTYPE] = 1, this field is automatically cleared after the requested block has been transmitted. This condition is detected when TL7816[TLEN] = 0 and four additional characters are transmitted.<br><br>0 Transmitter off.<br>1 Transmitter on. |
| 2<br>RE | Receiver Enable<br><br>Enables the UART receiver.<br><br>0 Receiver off.<br>1 Receiver on. |
| 1<br>RWU | Receiver Wakeup Control<br><br>This field can be set to place the UART receiver in a standby state. RWU automatically clears when an RWU event occurs, that is, an IDLE event when C1[WAKE] is clear or an address match when C1[WAKE] is set. This field must be cleared when C7816[ISO_7816E] is set.<br><br>**NOTE:** RWU must be set only with C1[WAKE] = 0 (wakeup on idle) if the channel is currently not idle. This can be determined by S2[RAF]. If the flag is set to wake up an IDLE event and the channel is already idle, it is possible that the UART will discard data. This is because the data must be received or a LIN break detected after an IDLE is detected before IDLE is allowed to reasserted.<br><br>0 Normal operation.<br>1 RWU enables the wakeup function and inhibits further receiver interrupt requests. Normally, hardware wakes the receiver by automatically clearing RWU. |
| 0<br>SBK | Send Break<br><br>Toggling SBK sends one break character from the following: See Transmitting break characters for the number of logic 0s for the different configurations. Toggling implies clearing the SBK field before the break character has finished transmitting. As long as SBK is set, the transmitter continues to send complete |

*Table continues on the next page...*

**UARTx_C2 field descriptions (continued)**

| Field | Description |
|---|---|
| | break characters (10, 11, or 12 bits, or 13 or 14 bits). Ensure that C2[TE] is asserted atleast 1 clock before assertion of this bit.<br>• 10, 11, or 12 logic 0s if S2[BRK13] is cleared<br>• 13 or 14 logic 0s if S2[BRK13] is set.<br><br>This field must be cleared when C7816[ISO_7816E] is set.<br><br>0 Normal transmitter operation.<br>1 Queue break characters to be sent. |

## 51.3.5 UART Status Register 1 (UARTx_S1)

The S1 register provides inputs to the MCU for generation of UART interrupts or DMA requests. This register can also be polled by the MCU to check the status of its fields. To clear a flag, the status register should be read followed by a read or write to D register, depending on the interrupt flag type. Other instructions can be executed between the two steps as long the handling of I/O is not compromised, but the order of operations is important for flag clearing. When a flag is configured to trigger a DMA request, assertion of the associated DMA done signal from the DMA controller clears the flag.

**NOTE**

- If the condition that results in the assertion of the flag, interrupt, or DMA request is not resolved prior to clearing the flag, the flag, and interrupt/DMA request, reasserts. For example, if the DMA or interrupt service routine fails to write sufficient data to the transmit buffer to raise it above the watermark level, the flag reasserts and generates another interrupt or DMA request.
- Reading an empty data register to clear one of the flags of the S1 register causes the FIFO pointers to become misaligned. A receive FIFO flush reinitializes the pointers. A better way to prevent this situation is to always leave one byte in FIFO and this byte will be read eventually in clearing the flag bit.

Address: Base address + 4h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| Write | | | | | | | | |
| Reset | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

## UARTx_S1 field descriptions

| Field | Description |
|---|---|
| 7<br>TDRE | Transmit Data Register Empty Flag<br><br>TDRE will set when the number of datawords in the transmit buffer (D and C3[T8])is equal to or less than the number indicated by TWFIFO[TXWATER]. A character that is in the process of being transmitted is not included in the count. To clear TDRE, read S1 when TDRE is set and then write to the UART data register (D). For more efficient interrupt servicing, all data except the final value to be written to the buffer must be written to D/C3[T8]. Then S1 can be read before writing the final data value, resulting in the clearing of the TRDE flag. This is more efficient because the TDRE reasserts until the watermark has been exceeded. So, attempting to clear the TDRE with every write will be ineffective until sufficient data has been written.<br><br>0 The amount of data in the transmit buffer is greater than the value indicated by TWFIFO[TXWATER].<br>1 The amount of data in the transmit buffer is less than or equal to the value indicated by TWFIFO[TXWATER] at some point in time since the flag has been cleared. |
| 6<br>TC | Transmit Complete Flag<br><br>TC is set when the transmit buffer is empty and no data, preamble, or break character is being transmitted. When TC is set, the transmit data output signal becomes idle (logic 1). TC is cleared by reading S1 with TC set and then doing one of the following: When C7816[ISO_7816E] is set/enabled, this field is set after any NACK signal has been received, but prior to any corresponding guard times expiring.<br>• Writing to D to transmit new data.<br>• Queuing a preamble by clearing and then setting C2[TE].<br>• Queuing a break character by writing 1 to SBK in C2.<br><br>0 Transmitter active (sending data, a preamble, or a break).<br>1 Transmitter idle (transmission activity complete). |
| 5<br>RDRF | Receive Data Register Full Flag<br><br>RDRF is set when the number of datawords in the receive buffer is equal to or more than the number indicated by RWFIFO[RXWATER]. A dataword that is in the process of being received is not included in the count. To clear RDRF, read S1 when RDRF is set and then read D. For more efficient interrupt and DMA operation, read all data except the final value from the buffer, using D/C3[T8]/ED. Then read S1 and the final data value, resulting in the clearing of the RDRF flag. Even if RDRF is set, data will continue to be received until an overrun condition occurs.RDRF is prevented from setting while S2[LBKDE] is set. Additionally, when S2[LBKDE] is set, the received datawords are stored in the receive buffer but over-write each other.<br><br>0 The number of datawords in the receive buffer is less than the number indicated by RXWATER.<br>1 The number of datawords in the receive buffer is equal to or greater than the number indicated by RXWATER at some point in time since this flag was last cleared. |
| 4<br>IDLE | Idle Line Flag<br>After the IDLE flag is cleared, a frame must be received (although not necessarily stored in the data buffer, for example if C2[RWU] is set), or a LIN break character must set the S2[LBKDIF] flag before an idle condition can set the IDLE flag. To clear IDLE, read UART status S1 with IDLE set and then read D. IDLE is set when either of the following appear on the receiver input:<br>• 10 consecutive logic 1s if C1[M] = 0<br>• 11 consecutive logic 1s if C1[M] = 1 and C4[M10] = 0<br>• 12 consecutive logic 1s if C1[M] = 1, C4[M10] = 1, and C1[PE] = 1<br><br>Idle detection is not supported when7816Eis set/enabled and hence this flag is ignored.<br><br>**NOTE:** When RWU is set and WAKE is cleared, an idle line condition sets the IDLE flag if RWUID is set, else the IDLE flag does not become set. |

*Table continues on the next page...*

## UARTx_S1 field descriptions (continued)

| Field | Description |
|---|---|
| | 0 Receiver input is either active now or has never become active since the IDLE flag was last cleared.<br>1 Receiver input has become idle or the flag has not been cleared since it last asserted. |
| 3<br>OR | Receiver Overrun Flag<br><br>OR is set when software fails to prevent the receive data register from overflowing with data. The OR bit is set immediately after the stop bit has been completely received for the dataword that overflows the buffer and all the other error flags (FE, NF, and PF) are prevented from setting. The data in the shift register is lost, but the data already in the UART data registers is not affected. If the OR flag is set, no data is stored in the data buffer even if sufficient room exists. Additionally, while the OR flag is set, the RDRF and IDLE flags are blocked from asserting, that is, transition from an inactive to an active state. To clear OR, read S1 when OR is set and then read D. See functional description for more details regarding the operation of the OR bit.If LBKDE is enabled and a LIN Break is detected, the OR field asserts if S2[LBKDIF] is not cleared before the next data character is received. In 7816 mode, it is possible to configure a NACK to be returned by programing C7816[ONACK].<br><br>0 No overrun has occurred since the last time the flag was cleared.<br>1 Overrun has occurred or the overrun flag has not been cleared since the last overrun occured. |
| 2<br>NF | Noise Flag<br><br>NF is set when the UART detects noise on the receiver input. NF does not become set in the case of an overrun or while the LIN break detect feature is enabled (S2[LBKDE] = 1). When NF is set, it indicates only that a dataword has been received with noise since the last time it was cleared. There is no guarantee that the first dataword read from the receive buffer has noise or that there is only one dataword in the buffer that was received with noise unless the receive buffer has a depth of one. To clear NF, read S1 and then read D.<br><br>0 No noise detected since the last time this flag was cleared. If the receive buffer has a depth greater than 1 then there may be data in the receiver buffer that was received with noise.<br>1 At least one dataword was received with noise detected since the last time the flag was cleared. |
| 1<br>FE | Framing Error Flag<br><br>FE is set when a logic 0 is accepted as the stop bit. FE does not set in the case of an overrun or while the LIN break detect feature is enabled (S2[LBKDE] = 1). FE inhibits further data reception until it is cleared. To clear FE, read S1 with FE set and then read D. The last data in the receive buffer represents the data that was received with the frame error enabled. Framing errors are not supported when 7816E is set/ enabled. However, if this flag is set, data is still not received in 7816 mode.<br><br>0 No framing error detected.<br>1 Framing error. |
| 0<br>PF | Parity Error Flag<br><br>PF is set when PE is set and the parity of the received data does not match its parity bit. The PF is not set in the case of an overrun condition. When PF is set, it indicates only that a dataword was received with parity error since the last time it was cleared. There is no guarantee that the first dataword read from the receive buffer has a parity error or that there is only one dataword in the buffer that was received with a parity error, unless the receive buffer has a depth of one. To clear PF, read S1 and then read D., S2[LBKDE] is disabled,Within the receive buffer structure the received dataword is tagged if it is received with a parity error. This information is available by reading the ED register prior to reading the D register.<br><br>0 No parity error detected since the last time this flag was cleared. If the receive buffer has a depth greater than 1, then there may be data in the receive buffer what was received with a parity error.<br>1 At least one dataword was received with a parity error since the last time this flag was cleared. |

## 51.3.6 UART Status Register 2 (UARTx_S2)

The S2 register provides inputs to the MCU for generation of UART interrupts or DMA requests. Also, this register can be polled by the MCU to check the status of these bits. This register can be read or written at any time, with the exception of the MSBF and RXINV bits, which should be changed by the user only between transmit and receive packets.

Address: Base address + 5h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | LBKDIF | RXEDGIF | MSBF | RXINV | RWUID | BRK13 | LBKDE | RAF |
| Write | w1c | w1c | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_S2 field descriptions**

| Field | Description |
|---|---|
| 7<br>LBKDIF | LIN Break Detect Interrupt Flag<br><br>LBKDIF is set when LBKDE is set and a LIN break character is detected on the receiver input. The LIN break characters are 11 consecutive logic 0s if C1[M] = 0 or 12 consecutive logic 0s if C1[M] = 1. LBKDIF is set after receiving the last LIN break character. LBKDIF is cleared by writing a 1 to it.<br><br>0 No LIN break character detected.<br>1 LIN break character detected. |
| 6<br>RXEDGIF | RxD Pin Active Edge Interrupt Flag<br><br>RXEDGIF is set when an active edge occurs on the RxD pin. The active edge is falling if RXINV = 0, and rising if RXINV=1. RXEDGIF is cleared by writing a 1 to it. See for additional details. RXEDGIF description<br><br>**NOTE:** The active edge is detected only in two wire mode and on receiving data coming from the RxD pin.<br><br>0 No active edge on the receive pin has occurred.<br>1 An active edge on the receive pin has occurred. |
| 5<br>MSBF | Most Significant Bit First<br><br>Setting this field reverses the order of the bits that are transmitted and received on the wire. This field does not affect the polarity of the bits, the location of the parity bit, or the location of the start or stop bits. This field is automatically set when C7816[INIT] and C7816[ISO7816E] are enabled and an initial character is detected in T = 0 protocol mode.<br><br>0 LSB (bit0) is the first bit that is transmitted following the start bit. Further, the first bit received after the start bit is identified as bit0.<br>1 MSB (bit8, bit7 or bit6) is the first bit that is transmitted following the start bit, depending on the setting of C1[M] and C1[PE]. Further, the first bit received after the start bit is identified as bit8, bit7, or bit6, depending on the setting of C1[M] and C1[PE]. |
| 4<br>RXINV | Receive Data Inversion |

*Table continues on the next page...*

## UARTx_S2 field descriptions (continued)

| Field | Description |
|---|---|
| | Setting this field reverses the polarity of the received data input. In NRZ format, a one is represented by a mark and a zero is represented by a space for normal polarity, and the opposite for inverted polarity. In IrDA format, a zero is represented by short high pulse in the middle of a bit time remaining idle low for a one for normal polarity. A zero is represented by a short low pulse in the middle of a bit time remaining idle high for a one for inverted polarity.This field is automatically set when C7816[INIT] and C7816[ISO7816E] are enabled and an initial character is detected in T = 0 protocol mode.<br><br>**NOTE:** Setting RXINV inverts the RxD input for data bits, start and stop bits, break, and idle. When C7816[ISO7816E] is set/enabled, only the data bits and the parity bit are inverted.<br><br>0 Receive data is not inverted.<br>1 Receive data is inverted. |
| 3<br>RWUID | Receive Wakeup Idle Detect<br><br>When RWU is set and WAKE is cleared, this field controls whether the idle character that wakes the receiver sets S1[IDLE]. This field must be cleared when C7816[ISO7816E] is set/enabled.<br><br>0 S1[IDLE] is not set upon detection of an idle character.<br>1 S1[IDLE] is set upon detection of an idle character. |
| 2<br>BRK13 | Break Transmit Character Length<br><br>Determines whether the transmit break character is 10, 11, or 12 bits long, or 13 or 14 bits long. See for the length of the break character for the different configurations. The detection of a framing error is not affected by this field. Transmitting break characters<br><br>0 Break character is 10, 11, or 12 bits long.<br>1 Break character is 13 or 14 bits long. |
| 1<br>LBKDE | LIN Break Detection Enable<br><br>Enables the LIN Break detection feature. While LBKDE is set, S1[RDRF], S1[NF], S1[FE], and S1[PF] are prevented from setting. When LBKDE is set, see . Overrun operationLBKDE must be cleared when C7816[ISO7816E] is set.<br><br>0 Break character detection is disabled.<br>1 Break character is detected at length of 11 bit times if C1[M] = 0 or 12 bits time if C1[M] = 1. |
| 0<br>RAF | Receiver Active Flag<br><br>RAF is set when the UART receiver detects a logic 0 during the RT1 time period of the start bit search. RAF is cleared when the receiver detects an idle character when C7816[ISO7816E] is cleared/disabled. When C7816[ISO7816E] is enabled, the RAF is cleared if the C7816[TTYPE] = 0 expires or the C7816[TTYPE] = 1 expires.<br><br>**NOTE:** In case C7816[ISO7816E] is set and C7816[TTYPE] = 0, it is possible to configure the guard time to 12. However, if a NACK is required to be transmitted, the data transfer actually takes 13 ETU with the 13th ETU slot being a inactive buffer. Therefore, in this situation, the RAF may deassert one ETU prior to actually being inactive.<br><br>0 UART receiver idle/inactive waiting for a start bit.<br>1 UART receiver active, RxD input not idle. |

## 51.3.7 UART Control Register 3 (UARTx_C3)

Writing R8 does not have any effect. TXDIR and TXINV can be changed only between transmit and receive packets.

Address: Base address + 6h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | R8 | T8 | TXDIR | TXINV | ORIE | NEIE | FEIE | PEIE |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_C3 field descriptions**

| Field | Description |
|---|---|
| 7<br>R8 | Received Bit 8<br><br>R8 is the ninth data bit received when the UART is configured for 9-bit data format, that is, if C1[M] = 1 or C4[M10] = 1. The R8 value corresponds to the current data value in the UARTx_D register. To read the 9th bit, read the value of UARTx_C3[R8], then read the UARTx_D register. |
| 6<br>T8 | Transmit Bit 8<br><br>T8 is the ninth data bit transmitted when the UART is configured for 9-bit data format, that is, if C1[M] = 1 or C4[M10] = 1.<br><br>**NOTE:** If the value of T8 is the same as in the previous transmission, T8 does not have to be rewritten. The same value is transmitted until T8 is rewritten.<br><br>To correctly transmit the 9th bit, write UARTx_C3[T8] to the desired value, then write the UARTx_D register with the remaining data. |
| 5<br>TXDIR | Transmitter Pin Data Direction in Single-Wire mode<br><br>Determines whether the TXD pin is used as an input or output in the single-wire mode of operation. This field is relevant only to the single wire mode. When C7816[ISO7816E] is set/enabled and C7816[TTYPE] = 1, this field is automatically cleared after the requested block is transmitted. This condition is detected when TL7816[TLEN] = 0 and 4 additional characters are transmitted. Additionally, if C7816[ISO7816E] is set/enabled and C7816[TTYPE] = 0 and a NACK is being transmitted, the hardware automatically overrides this field as needed. In this situation, TXDIR does not reflect the temporary state associated with the NACK.<br><br>0 TXD pin is an input in single wire mode.<br>1 TXD pin is an output in single wire mode. |
| 4<br>TXINV | Transmit Data Inversion.<br><br>Setting this field reverses the polarity of the transmitted data output. In NRZ format, a one is represented by a mark and a zero is represented by a space for normal polarity, and the opposite for inverted polarity. In IrDA format, a zero is represented by short high pulse in the middle of a bit time remaining idle low for a one for normal polarity, and a zero is represented by short low pulse in the middle of a bit time remaining idle high for a one for inverted polarity.This field is automatically set when C7816[INIT] and C7816[ISO7816E] are enabled and an initial character is detected in T = 0 protocol mode.<br><br>**NOTE:** Setting TXINV inverts all transmitted values, including idle, break, start, and stop bits. In loop mode, if TXINV is set, the receiver gets the transmit inversion bit when RXINV is disabled. When C7816[ISO7816E] is set/enabled then only the transmitted data bits and parity bit are inverted. |

*Table continues on the next page...*

**UARTx_C3 field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Transmit data is not inverted.<br>1 Transmit data is inverted. |
| 3<br>ORIE | Overrun Error Interrupt Enable<br><br>Enables the overrun error flag, S1[OR], to generate interrupt requests.<br><br>0 OR interrupts are disabled.<br>1 OR interrupt requests are enabled. |
| 2<br>NEIE | Noise Error Interrupt Enable<br><br>Enables the noise flag, S1[NF], to generate interrupt requests.<br><br>0 NF interrupt requests are disabled.<br>1 NF interrupt requests are enabled. |
| 1<br>FEIE | Framing Error Interrupt Enable<br><br>Enables the framing error flag, S1[FE], to generate interrupt requests.<br><br>0 FE interrupt requests are disabled.<br>1 FE interrupt requests are enabled. |
| 0<br>PEIE | Parity Error Interrupt Enable<br><br>Enables the parity error flag, S1[PF], to generate interrupt requests.<br><br>0 PF interrupt requests are disabled.<br>1 PF interrupt requests are enabled. |

## 51.3.8 UART Data Register (UARTx_D)

This register is actually two separate registers. Reads return the contents of the read-only receive data register and writes go to the write-only transmit data register.

**NOTE**

- In 8-bit or 9-bit data format, only UART data register (D) needs to be accessed to clear the S1[RDRF] bit (assuming receiver buffer level is less than RWFIFO[RXWATER]). The C3 register needs to be read, prior to the D register, only if the ninth bit of data needs to be captured. Similarly, the ED register needs to be read, prior to the D register, only if the additional flag data for the dataword needs to be captured.
- In the normal 8-bit mode (M bit cleared) if the parity is enabled, you get seven data bits and one parity bit. That one parity bit is loaded into the D register. So, for the data

bits, mask off the parity bit from the value you read out of this register.

- When transmitting in 9-bit data format and using 8-bit write instructions, write first to transmit bit 8 in UART control register 3 (C3[T8]), then D. A write to C3[T8] stores the data in a temporary register. If D register is written first, and then the new data on data bus is stored in D, the temporary value written by the last write to C3[T8] gets stored in the C3[T8] register.

Address: Base address + 7h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | RT | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_D field descriptions**

| Field | Description |
|---|---|
| 7–0<br>RT | Reads return the contents of the read-only receive data register and writes go to the write-only transmit data register. |

## 51.3.9 UART Match Address Registers 1 (UARTx_MA1)

The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated C4[MAEN] field is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded. These registers can be read and written at anytime.

Address: Base address + 8h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | MA | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_MA1 field descriptions**

| Field | Description |
|---|---|
| 7–0<br>MA | Match Address |

## 51.3.10 UART Match Address Registers 2 (UARTx_MA2)

These registers can be read and written at anytime. The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated C4[MAEN] field is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded.

Address: Base address + 9h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | MA | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_MA2 field descriptions**

| Field | Description |
|---|---|
| 7–0<br>MA | Match Address |

## 51.3.11 UART Control Register 4 (UARTx_C4)

Address: Base address + Ah offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | MAEN1 | MAEN2 | M10 | BRFA | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_C4 field descriptions**

| Field | Description |
|---|---|
| 7<br>MAEN1 | Match Address Mode Enable 1<br><br>See Match address operation for more information.<br><br>0 All data received is transferred to the data buffer if MAEN2 is cleared.<br>1 All data received with the most significant bit cleared, is discarded. All data received with the most significant bit set, is compared with contents of MA1 register. If no match occurs, the data is discarded. If match occurs, data is transferred to the data buffer. This field must be cleared when C7816[ISO7816E] is set/enabled. |
| 6<br>MAEN2 | Match Address Mode Enable 2<br><br>See Match address operation for more information.<br><br>0 All data received is transferred to the data buffer if MAEN1 is cleared.<br>1 All data received with the most significant bit cleared, is discarded. All data received with the most significant bit set, is compared with contents of MA2 register. If no match occurs, the data is discarded. If a match occurs, data is transferred to the data buffer. This field must be cleared when C7816[ISO7816E] is set/enabled. |

*Table continues on the next page...*

**UARTx_C4 field descriptions (continued)**

| Field | Description |
|---|---|
| 5<br>M10 | 10-bit Mode select<br><br>Causes a tenth, non-memory mapped bit to be part of the serial transmission. This tenth bit is generated and interpreted as a parity bit. The M10 field does not affect the LIN send or detect break behavior. If M10 is set, then both C1[M] and C1[PE] must also be set. This field must be cleared when C7816[ISO7816E] is set/enabled.<br><br>See Data format (non ISO-7816) for more information.<br><br>0 The parity bit is the ninth bit in the serial transmission.<br>1 The parity bit is the tenth bit in the serial transmission. |
| 4–0<br>BRFA | Baud Rate Fine Adjust<br><br>This bit field is used to add more timing resolution to the average baud frequency, in increments of 1/32. See Baud rate generation for more information. |

## 51.3.12 UART Control Register 5 (UARTx_C5)

Address: Base address + Bh offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | TDMAS | 0 | RDMAS | 0 | | | | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_C5 field descriptions**

| Field | Description |
|---|---|
| 7<br>TDMAS | Transmitter DMA Select<br><br>Configures the transmit data register empty flag, S1[TDRE], to generate interrupt or DMA requests if C2[TIE] is set.<br><br>**NOTE:** • If C2[TIE] is cleared, TDRE DMA and TDRE interrupt request signals are not asserted when the TDRE flag is set, regardless of the state of TDMAS.<br>• If C2[TIE] and TDMAS are both set, then C2[TCIE] must be cleared, and D must not be written unless a DMA request is being serviced.<br><br>0 If C2[TIE] is set and the S1[TDRE] flag is set, the TDRE interrupt request signal is asserted to request interrupt service.<br>1 If C2[TIE] is set and the S1[TDRE] flag is set, the TDRE DMA request signal is asserted to request a DMA transfer. |
| 6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5<br>RDMAS | Receiver Full DMA Select<br><br>Configures the receiver data register full flag, S1[RDRF], to generate interrupt or DMA requests if C2[RIE] is set.<br><br>**NOTE:** If C2[RIE] is cleared, and S1[RDRF] is set, the RDRF DMA and RDFR interrupt request signals are not asserted, regardless of the state of RDMAS. |

*Table continues on the next page...*

**UARTx_C5 field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 If C2[RIE] and S1[RDRF] are set, the RDFR interrupt request signal is asserted to request an interrupt service.<br>1 If C2[RIE] and S1[RDRF] are set, the RDRF DMA request signal is asserted to request a DMA transfer. |
| 4–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 51.3.13 UART Extended Data Register (UARTx_ED)

This register contains additional information flags that are stored with a received dataword. This register may be read at any time but contains valid data only if there is a dataword in the receive FIFO.

**NOTE**

- The data contained in this register represents additional information regarding the conditions on which a dataword was received. The importance of this data varies with the application, and in some cases maybe completely optional. These fields automatically update to reflect the conditions of the next dataword whenever D is read.
- If S1[NF] and S1[PF] have not been set since the last time the receive buffer was empty, the NOISY and PARITYE fields will be zero.

Address: Base address + Ch offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | NOISY | PARITYE | 0 | | | | | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_ED field descriptions**

| Field | Description |
|---|---|
| 7<br>NOISY | The current received dataword contained in D and C3[R8] was received with noise.<br>0 The dataword was received without noise.<br>1 The data was received with noise. |
| 6<br>PARITYE | The current received dataword contained in D and C3[R8] was received with a parity error.<br>0 The dataword was received without a parity error.<br>1 The dataword was received with a parity error. |
| 5–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 51.3.14 UART Modem Register (UARTx_MODEM)

The MODEM register controls options for setting the modem configuration.

**NOTE**

RXRTSE, TXRTSPOL, TXRTSE, and TXCTSE must all be cleared when C7816[ISO7816EN] is enabled. This will cause the RTS to deassert during ISO-7816 wait times. The ISO-7816 protocol does not use the RTS and CTS signals.

Address: Base address + Dh offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | | | | RXRTSE | TXRTSPOL | TXRTSE | TXCTSE |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_MODEM field descriptions**

| Field | Description |
|---|---|
| 7–4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3<br>RXRTSE | Receiver request-to-send enable<br><br>Allows the RTS output to control the CTS input of the transmitting device to prevent receiver overrun.<br><br>**NOTE:** Do not set both RXRTSE and TXRTSE.<br><br>0 The receiver has no effect on RTS.<br>1 RTS is deasserted if the number of characters in the receiver data register (FIFO) is equal to or greater than RWFIFO[RXWATER]. RTS is asserted when the number of characters in the receiver data register (FIFO) is less than RWFIFO[RXWATER]. |
| 2<br>TXRTSPOL | Transmitter request-to-send polarity<br><br>Controls the polarity of the transmitter RTS. TXRTSPOL does not affect the polarity of the receiver RTS. RTS will remain negated in the active low state unless TXRTSE is set.<br><br>0 Transmitter RTS is active low.<br>1 Transmitter RTS is active high. |
| 1<br>TXRTSE | Transmitter request-to-send enable<br><br>Controls RTS before and after a transmission.<br><br>0 The transmitter has no effect on RTS.<br>1 When a character is placed into an empty transmitter data buffer , RTS asserts one bit time before the start bit is transmitted. RTS deasserts one bit time after all characters in the transmitter data buffer and shift register are completely sent, including the last stop bit. (FIFO)(FIFO) |
| 0<br>TXCTSE | Transmitter clear-to-send enable<br><br>TXCTSE controls the operation of the transmitter. TXCTSE can be set independently from the state of TXRTSE and RXRTSE. |

*Table continues on the next page...*

**UARTx_MODEM field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 CTS has no effect on the transmitter.<br>1 Enables clear-to-send operation. The transmitter checks the state of CTS each time it is ready to send a character. If CTS is asserted, the character is sent. If CTS is deasserted, the signal TXD remains in the mark state and transmission is delayed until CTS is asserted. Changes in CTS as a character is being sent do not affect its transmission. |

## 51.3.15 UART Infrared Register (UARTx_IR)

The IR register controls options for setting the infrared configuration.

Address: Base address + Eh offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | | | | | IREN | TNP | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_IR field descriptions**

| Field | Description |
|---|---|
| 7–3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>IREN | Infrared enable<br><br>Enables/disables the infrared modulation/demodulation.<br><br>0 IR disabled.<br>1 IR enabled. |
| 1–0<br>TNP | Transmitter narrow pulse<br><br>Enables whether the UART transmits a 1/16, 3/16, 1/32, or 1/4 narrow pulse.<br><br>00 3/16.<br>01 1/16.<br>10 1/32.<br>11 1/4. |

## 51.3.16 UART FIFO Parameters (UARTx_PFIFO)

This register provides the ability for the programmer to turn on and off FIFO functionality. It also provides the size of the FIFO that has been implemented. This register may be read at any time. This register must be written only when C2[RE] and C2[TE] are cleared/not set and when the data buffer/FIFO is empty.

Address: Base address + 10h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | TXFE | TXFIFOSIZE | | | RXFE | RXFIFOSIZE | | |
| Write | | | | | | | | |
| Reset | 0 | * | * | * | 0 | * | * | * |

* Notes:
- TXFIFOSIZE field: The reset value depends on whether the specific UART instance supports the FIFO and on the size of that FIFO. See the Chip Configuration details for more information on the FIFO size supported for each UART instance.
- RXFIFOSIZE field: The reset value depends on whether the specific UART instance supports the FIFO and on the size of that FIFO. See the Chip Configuration details for more information on the FIFO size supported for each UART instance.

**UARTx_PFIFO field descriptions**

| Field | Description |
|---|---|
| 7<br>TXFE | Transmit FIFO Enable<br><br>When this field is set, the built in FIFO structure for the transmit buffer is enabled. The size of the FIFO structure is indicated by TXFIFOSIZE. If this field is not set, the transmit buffer operates as a FIFO of depth one dataword regardless of the value in TXFIFOSIZE. Both C2[TE] and C2[RE] must be cleared prior to changing this field. Additionally, TXFLUSH and RXFLUSH commands must be issued immediately after changing this field.<br><br>0 Transmit FIFO is not enabled. Buffer is depth 1. (Legacy support).<br>1 Transmit FIFO is enabled. Buffer is depth indicated by TXFIFOSIZE. |
| 6–4<br>TXFIFOSIZE | Transmit FIFO. Buffer Depth<br><br>The maximum number of transmit datawords that can be stored in the transmit buffer. This field is read only.<br><br>000 Transmit FIFO/Buffer depth = 1 dataword.<br>001 Transmit FIFO/Buffer depth = 4 datawords.<br>010 Transmit FIFO/Buffer depth = 8 datawords.<br>011 Transmit FIFO/Buffer depth = 16 datawords.<br>100 Transmit FIFO/Buffer depth = 32 datawords.<br>101 Transmit FIFO/Buffer depth = 64 datawords.<br>110 Transmit FIFO/Buffer depth = 128 datawords.<br>111 Reserved. |
| 3<br>RXFE | Receive FIFO Enable |

*Table continues on the next page...*

**UARTx_PFIFO field descriptions (continued)**

| Field | Description |
|---|---|
| | When this field is set, the built in FIFO structure for the receive buffer is enabled. The size of the FIFO structure is indicated by the RXFIFOSIZE field. If this field is not set, the receive buffer operates as a FIFO of depth one dataword regardless of the value in RXFIFOSIZE. Both C2[TE] and C2[RE] must be cleared prior to changing this field. Additionally, TXFLUSH and RXFLUSH commands must be issued immediately after changing this field.<br><br>0 Receive FIFO is not enabled. Buffer is depth 1. (Legacy support)<br>1 Receive FIFO is enabled. Buffer is depth indicted by RXFIFOSIZE. |
| 2–0<br>RXFIFOSIZE | Receive FIFO. Buffer Depth<br><br>The maximum number of receive datawords that can be stored in the receive buffer before an overrun occurs. This field is read only.<br><br>000 Receive FIFO/Buffer depth = 1 dataword.<br>001 Receive FIFO/Buffer depth = 4 datawords.<br>010 Receive FIFO/Buffer depth = 8 datawords.<br>011 Receive FIFO/Buffer depth = 16 datawords.<br>100 Receive FIFO/Buffer depth = 32 datawords.<br>101 Receive FIFO/Buffer depth = 64 datawords.<br>110 Receive FIFO/Buffer depth = 128 datawords.<br>111 Reserved. |

## 51.3.17 UART FIFO Control Register (UARTx_CFIFO)

This register provides the ability to program various control fields for FIFO operation. This register may be read or written at any time. Note that writing to TXFLUSH and RXFLUSH may result in data loss and requires careful action to prevent unintended/ unpredictable behavior. Therefore, it is recommended that TE and RE be cleared prior to flushing the corresponding FIFO.

Address: Base address + 11h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | 0 | 0 | | | RXOFE | TXOFE | RXUFE |
| Write | TXFLUSH | RXFLUSH | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_CFIFO field descriptions**

| Field | Description |
|---|---|
| 7<br>TXFLUSH | Transmit FIFO/Buffer Flush<br><br>Writing to this field causes all data that is stored in the transmit FIFO/buffer to be flushed. This does not affect data that is in the transmit shift register.<br><br>0 No flush operation occurs.<br>1 All data in the transmit FIFO/Buffer is cleared out. |

*Table continues on the next page...*

**UARTx_CFIFO field descriptions (continued)**

| Field | Description |
|---|---|
| 6<br>RXFLUSH | Receive FIFO/Buffer Flush<br><br>Writing to this field causes all data that is stored in the receive FIFO/buffer to be flushed. This does not affect data that is in the receive shift register.<br><br>0 No flush operation occurs.<br>1 All data in the receive FIFO/buffer is cleared out. |
| 5–3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>RXOFE | Receive FIFO Overflow Interrupt Enable<br><br>When this field is set, the RXOF flag generates an interrupt to the host.<br><br>0 RXOF flag does not generate an interrupt to the host.<br>1 RXOF flag generates an interrupt to the host. |
| 1<br>TXOFE | Transmit FIFO Overflow Interrupt Enable<br><br>When this field is set, the TXOF flag generates an interrupt to the host.<br><br>0 TXOF flag does not generate an interrupt to the host.<br>1 TXOF flag generates an interrupt to the host. |
| 0<br>RXUFE | Receive FIFO Underflow Interrupt Enable<br><br>When this field is set, the RXUF flag generates an interrupt to the host.<br><br>0 RXUF flag does not generate an interrupt to the host.<br>1 RXUF flag generates an interrupt to the host. |

## 51.3.18 UART FIFO Status Register (UARTx_SFIFO)

This register provides status information regarding the transmit and receiver buffers/ FIFOs, including interrupt information. This register may be written to or read at any time.

Address: Base address + 12h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | TXEMPT | RXEMPT | 0 | | | RXOF | TXOF | RXUF |
| Write | | | | | | w1c | w1c | w1c |
| Reset | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_SFIFO field descriptions**

| Field | Description |
|---|---|
| 7<br>TXEMPT | Transmit Buffer/FIFO Empty<br><br>Asserts when there is no data in the Transmit FIFO/buffer. This field does not take into account data that is in the transmit shift register. |

*Table continues on the next page...*

**UARTx_SFIFO field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Transmit buffer is not empty.<br>1 Transmit buffer is empty. |
| 6<br>RXEMPT | Receive Buffer/FIFO Empty<br><br>Asserts when there is no data in the receive FIFO/Buffer. This field does not take into account data that is in the receive shift register.<br><br>0 Receive buffer is not empty.<br>1 Receive buffer is empty. |
| 5–3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>RXOF | Receiver Buffer Overflow Flag<br><br>Indicates that more data has been written to the receive buffer than it can hold. This field will assert regardless of the value of CFIFO[RXOFE]. However, an interrupt will be issued to the host only if CFIFO[RXOFE] is set. This flag is cleared by writing a 1.<br><br>0 No receive buffer overflow has occurred since the last time the flag was cleared.<br>1 At least one receive buffer overflow has occurred since the last time the flag was cleared. |
| 1<br>TXOF | Transmitter Buffer Overflow Flag<br><br>Indicates that more data has been written to the transmit buffer than it can hold. This field will assert regardless of the value of CFIFO[TXOFE]. However, an interrupt will be issued to the host only if CFIFO[TXOFE] is set. This flag is cleared by writing a 1.<br><br>0 No transmit buffer overflow has occurred since the last time the flag was cleared.<br>1 At least one transmit buffer overflow has occurred since the last time the flag was cleared. |
| 0<br>RXUF | Receiver Buffer Underflow Flag<br><br>Indicates that more data has been read from the receive buffer than was present. This field will assert regardless of the value of CFIFO[RXUFE]. However, an interrupt will be issued to the host only if CFIFO[RXUFE] is set. This flag is cleared by writing a 1.<br><br>0 No receive buffer underflow has occurred since the last time the flag was cleared.<br>1 At least one receive buffer underflow has occurred since the last time the flag was cleared. |

## 51.3.19 UART FIFO Transmit Watermark (UARTx_TWFIFO)

This register provides the ability to set a programmable threshold for notification of needing additional transmit data. This register may be read at any time but must be written only when C2[TE] is not set. Changing the value of the watermark will not clear the S1[TDRE] flag.

Address: Base address + 13h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | TXWATER | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_TWFIFO field descriptions**

| Field | Description |
|---|---|
| 7–0<br>TXWATER | Transmit Watermark<br><br>When the number of datawords in the transmit FIFO/buffer is equal to or less than the value in this register field, an interrupt via S1[TDRE] or a DMA request via C5[TDMAS] is generated as determined by C5[TDMAS] and C2[TIE]. For proper operation, the value in TXWATER must be set to be less than the size of the transmit buffer/FIFO size as indicated by PFIFO[TXFIFOSIZE] and PFIFO[TXFE]. |

## 51.3.20 UART FIFO Transmit Count (UARTx_TCFIFO)

This is a read only register that indicates how many datawords are currently in the transmit buffer/FIFO. It may be read at any time.

Address: Base address + 14h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | TXCOUNT | | | | | | | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_TCFIFO field descriptions**

| Field | Description |
|---|---|
| 7–0<br>TXCOUNT | Transmit Counter<br><br>The value in this register indicates the number of datawords that are in the transmit FIFO/buffer. If a dataword is being transmitted, that is, in the transmit shift register, it is not included in the count. This value may be used in conjunction with PFIFO[TXFIFOSIZE] to calculate how much room is left in the transmit FIFO/buffer. |

## 51.3.21 UART FIFO Receive Watermark (UARTx_RWFIFO)

This register provides the ability to set a programmable threshold for notification of the need to remove data from the receiver FIFO/buffer. This register may be read at any time but must be written only when C2[RE] is not asserted. Changing the value in this register will not clear S1[RDRF].

Address: Base address + 15h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | RXWATER | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**UARTx_RWFIFO field descriptions**

| Field | Description |
|---|---|
| 7–0<br>RXWATER | Receive Watermark<br><br>When the number of datawords in the receive FIFO/buffer is equal to or greater than the value in this register field, an interrupt via S1[RDRF] or a DMA request via C5[RDMAS] is generated as determined by C5[RDMAS] and C2[RIE]. For proper operation, the value in RXWATER must be set to be less than the receive FIFO/buffer size as indicated by PFIFO[RXFIFOSIZE] and PFIFO[RXFE] and must be greater than 0. |

## 51.3.22 UART FIFO Receive Count (UARTx_RCFIFO)

This is a read only register that indicates how many datawords are currently in the receive FIFO/buffer. It may be read at any time.

Address: Base address + 16h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | RXCOUNT | | | | | | | |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_RCFIFO field descriptions**

| Field | Description |
|---|---|
| 7–0<br>RXCOUNT | Receive Counter<br><br>The value in this register indicates the number of datawords that are in the receive FIFO/buffer. If a dataword is being received, that is, in the receive shift register, it is not included in the count. This value may be used in conjunction with PFIFO[RXFIFOSIZE] to calculate how much room is left in the receive FIFO/buffer. |

## 51.3.23 UART 7816 Control Register (UARTx_C7816)

The C7816 register is the primary control register for ISO-7816 specific functionality. This register is specific to 7816 functionality and the values in this register have no effect on UART operation and should be ignored if ISO_7816E is not set/enabled. This register may be read at any time but values must be changed only when ISO_7816E is not set.

Address: Base address + 18h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | 0 | | | ONACK | ANACK | INIT | TTYPE | ISO_7816E |
| Write | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## UARTx_C7816 field descriptions

| Field | Description |
|---|---|
| 7–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>ONACK | Generate NACK on Overflow<br><br>When this field is set, the receiver automatically generates a NACK response if a receive buffer overrun occurs, as indicated by S1[OR]. In many systems, this results in the transmitter resending the packet that overflowed until the retransmit threshold for that transmitter is reached. A NACK is generated only if TTYPE=0. This field operates independently of ANACK. See . Overrun NACK considerations<br><br>0 The received data does not generate a NACK when the receipt of the data results in an overflow event.<br>1 If the receiver buffer overflows, a NACK is automatically sent on a received character. |
| 3<br>ANACK | Generate NACK on Error<br><br>When this field is set, the receiver automatically generates a NACK response if a parity error occurs or if INIT is set and an invalid initial character is detected. A NACK is generated only if TTYPE = 0. If ANACK is set, the UART attempts to retransmit the data indefinitely. To stop retransmission attempts, clear C2[TE] or ISO_7816E and do not set until S1[TC] sets C2[TE] again.<br><br>0 No NACK is automatically generated.<br>1 A NACK is automatically generated if a parity error is detected or if an invalid initial character is detected. |
| 2<br>INIT | Detect Initial Character<br><br>When this field is set, all received characters are searched for a valid initial character. If an invalid initial character is identified, and ANACK is set, a NACK is sent. All received data is discarded and error flags blocked (S1[NF], S1[OR], S1[FE], S1[PF], IS7816[WT], IS7816[CWT], IS7816[BWT], IS7816[GTV]) until a valid initial character is detected. Upon detecting a valid initial character, the configuration values S2[MSBF], C3[TXINV], and S2[RXINV] are automatically updated to reflect the initial character that was received. The actual INIT data value is not stored in the receive buffer. Additionally, upon detection of a valid initial character, IS7816[INITD] is set and an interrupt issued as programmed by IE7816[INITDE]. When a valid initial character is detected, INIT is automatically cleared. This Initial Character Detect feature is supported only in T = 0 protocol mode.<br><br>0 Normal operating mode. Receiver does not seek to identify initial character.<br>1 Receiver searches for initial character. |
| 1<br>TTYPE | Transfer Type<br><br>Indicates the transfer protocol being used.<br><br>See ISO-7816 / smartcard support for more details.<br><br>0 T = 0 per the ISO-7816 specification.<br>1 T = 1 per the ISO-7816 specification. |
| 0<br>ISO_7816E | ISO-7816 Functionality Enabled<br><br>Indicates that the UART is operating according to the ISO-7816 protocol.<br><br>**NOTE:** This field must be modified only when no transmit or receive is occurring. If this field is changed during a data transfer, the data being transmitted or received may be transferred incorrectly.<br><br>0 ISO-7816 functionality is turned off/not enabled.<br>1 ISO-7816 functionality is turned on/enabled. |

## 51.3.24 UART 7816 Interrupt Enable Register (UARTx_IE7816)

The IE7816 register controls which flags result in an interrupt being issued. This register is specific to 7816 functionality, the corresponding flags that drive the interrupts are not asserted when 7816E is not set/enabled. However, these flags may remain set if they are asserted while 7816E was set and not subsequently cleared. This register may be read or written to at any time.

Address: Base address + 19h offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | WTE | CWTE | BWTE | INITDE | 0 | GTVE | TXTE | RXTE |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_IE7816 field descriptions**

| Field | Description |
|---|---|
| 7<br>WTE | Wait Timer Interrupt Enable<br><br>0 The assertion of IS7816[WT] does not result in the generation of an interrupt.<br>1 The assertion of IS7816[WT] results in the generation of an interrupt. |
| 6<br>CWTE | Character Wait Timer Interrupt Enable<br><br>0 The assertion of IS7816[CWT] does not result in the generation of an interrupt.<br>1 The assertion of IS7816[CWT] results in the generation of an interrupt. |
| 5<br>BWTE | Block Wait Timer Interrupt Enable<br><br>0 The assertion of IS7816[BWT] does not result in the generation of an interrupt.<br>1 The assertion of IS7816[BWT] results in the generation of an interrupt. |
| 4<br>INITDE | Initial Character Detected Interrupt Enable<br><br>0 The assertion of IS7816[INITD] does not result in the generation of an interrupt.<br>1 The assertion of IS7816[INITD] results in the generation of an interrupt. |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>GTVE | Guard Timer Violated Interrupt Enable<br><br>0 The assertion of IS7816[GTV] does not result in the generation of an interrupt.<br>1 The assertion of IS7816[GTV] results in the generation of an interrupt. |
| 1<br>TXTE | Transmit Threshold Exceeded Interrupt Enable<br><br>0 The assertion of IS7816[TXT] does not result in the generation of an interrupt.<br>1 The assertion of IS7816[TXT] results in the generation of an interrupt. |
| 0<br>RXTE | Receive Threshold Exceeded Interrupt Enable<br><br>0 The assertion of IS7816[RXT] does not result in the generation of an interrupt.<br>1 The assertion of IS7816[RXT] results in the generation of an interrupt. |

## 51.3.25 UART 7816 Interrupt Status Register (UARTx_IS7816)

The IS7816 register provides a mechanism to read and clear the interrupt flags. All flags/interrupts are cleared by writing a 1 to the field location. Writing a 0 has no effect. All bits are "sticky", meaning they indicate that only the flag condition that occurred since the last time the bit was cleared, not that the condition currently exists. The status flags are set regardless of whether the corresponding field in the IE7816 is set or cleared. The IE7816 controls only if an interrupt is issued to the host processor. This register is specific to 7816 functionality and the values in this register have no affect on UART operation and should be ignored if 7816E is not set/enabled. This register may be read or written at anytime.

Address: Base address + 1Ah offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read | WT | CWT | BWT | INITD | 0 | GTV | TXT | RXT |
| Write | w1c | w1c | w1c | w1c | | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_IS7816 field descriptions**

| Field | Description |
|---|---|
| 7<br>WT | Wait Timer Interrupt<br><br>Indicates that the wait time, the time between the leading edge of a character being transmitted and the leading edge of the next response character, has exceeded the programmed value. This flag asserts only when C7816[TTYPE] = 0. This interrupt is cleared by writing 1.<br><br>0 Wait time (WT) has not been violated.<br>1 Wait time (WT) has been violated. |
| 6<br>CWT | Character Wait Timer Interrupt<br><br>Indicates that the character wait time, the time between the leading edges of two consecutive characters in a block, has exceeded the programmed value. This flag asserts only when C7816[TTYPE] = 1. This interrupt is cleared by writing 1.<br><br>0 Character wait time (CWT) has not been violated.<br>1 Character wait time (CWT) has been violated. |
| 5<br>BWT | Block Wait Timer Interrupt<br><br>Indicates that the block wait time, the time between the leading edge of first received character of a block and the leading edge of the last character the previously transmitted block, has exceeded the programmed value. This flag asserts only when C7816[TTYPE] = 1.This interrupt is cleared by writing 1.<br><br>0 Block wait time (BWT) has not been violated.<br>1 Block wait time (BWT) has been violated. |
| 4<br>INITD | Initial Character Detected Interrupt<br><br>Indicates that a valid initial character is received. This interrupt is cleared by writing 1. |

*Table continues on the next page...*

**UARTx_IS7816 field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 A valid initial character has not been received.<br>1 A valid initial character has been received. |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>GTV | Guard Timer Violated Interrupt<br><br>Indicates that one or more of the character guard time, block guard time, or guard time are violated. This interrupt is cleared by writing 1.<br><br>0 A guard time (GT, CGT, or BGT) has not been violated.<br>1 A guard time (GT, CGT, or BGT) has been violated. |
| 1<br>TXT | Transmit Threshold Exceeded Interrupt<br><br>Indicates that the transmit NACK threshold has been exceeded as indicated by ET7816[TXTHRESHOLD]. Regardless of whether this flag is set, the UART continues to retransmit indefinitely. This flag asserts only when C7816[TTYPE] = 0. If 7816E is cleared/disabled, ANACK is cleared/disabled, C2[TE] is cleared/disabled, C7816[TTYPE] = 1, or packet is transferred without receiving a NACK, the internal NACK detection counter is cleared and the count restarts from zero on the next received NACK. This interrupt is cleared by writing 1.<br><br>0 The number of retries and corresponding NACKS does not exceed the value in ET7816[TXTHRESHOLD].<br>1 The number of retries and corresponding NACKS exceeds the value in ET7816[TXTHRESHOLD]. |
| 0<br>RXT | Receive Threshold Exceeded Interrupt<br><br>Indicates that there are more than ET7816[RXTHRESHOLD] consecutive NACKS generated in response to parity errors on received data. This flag requires ANACK to be set. Additionally, this flag asserts only when C7816[TTYPE] = 0. Clearing this field also resets the counter keeping track of consecutive NACKS. The UART will continue to attempt to receive data regardless of whether this flag is set. If 7816E is cleared/disabled, RE is cleared/disabled, C7816[TTYPE] = 1, or packet is received without needing to issue a NACK, the internal NACK detection counter is cleared and the count restarts from zero on the next transmitted NACK. This interrupt is cleared by writing 1.<br><br>0 The number of consecutive NACKS generated as a result of parity errors and buffer overruns is less than or equal to the value in ET7816[RXTHRESHOLD].<br>1 The number of consecutive NACKS generated as a result of parity errors and buffer overruns is greater than the value in ET7816[RXTHRESHOLD]. |

## 51.3.26 UART 7816 Wait Parameter Register (UARTx_WP7816T0)

The WP7816 register contains constants used in the generation of various wait timer counters. To save register space, this register is used differently when C7816[TTYPE] = 0 and C7816[TTYPE] = 1. This register may be read at any time. This register must be written to only when C7816[ISO_7816E] is not set.

Address: Base address + 1Bh offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | WI | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

**UARTx_WP7816T0 field descriptions**

| Field | Description |
|---|---|
| 7–0<br>WI | Wait Time Integer (C7816[TTYPE] = 0)<br><br>Used to calculate the value used for the WT counter. It represents a value between 1 and 255. The value of zero is not valid. This value is used only when C7816[TTYPE] = 0. See Wait time and guard time parameters. |

## 51.3.27 UART 7816 Wait Parameter Register (UARTx_WP7816T1)

The WP7816 register contains constants used in the generation of various wait timer counters. To save register space, this register is used differently when C7816[TTYPE] = 0 and C7816[TTYPE] = 1. This register may be read at any time. This register must be written to only when C7816[ISO_7816E] is not set.

Address: Base address + 1Bh offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | CWI | | | | BWI | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

**UARTx_WP7816T1 field descriptions**

| Field | Description |
|---|---|
| 7–4<br>CWI | Character Wait Time Integer (C7816[TTYPE] = 1)<br><br>Used to calculate the value used for the CWT counter. It represents a value between 0 and 15. This value is used only when C7816[TTYPE] = 1. See Wait time and guard time parameters . |
| 3–0<br>BWI | Block Wait Time Integer(C7816[TTYPE] = 1)<br><br>Used to calculate the value used for the BWT counter. It represent a value between 0 and 15. This value is used only when C7816[TTYPE] = 1. See Wait time and guard time parameters . |

## 51.3.28 UART 7816 Wait N Register (UARTx_WN7816)

The WN7816 register contains a parameter that is used in the calculation of the guard time counter. This register may be read at any time. This register must be written to only when C7816[ISO_7816E] is not set.

Address: Base address + 1Ch offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | GTN | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_WN7816 field descriptions**

| Field | Description |
|---|---|
| 7–0<br>GTN | Guard Band N<br><br>Defines a parameter used in the calculation of GT, CGT, and BGT counters. The value represents an integer number between 0 and 255. See Wait time and guard time parameters . |

## 51.3.29 UART 7816 Wait FD Register (UARTx_WF7816)

The WF7816 contains parameters that are used in the generation of various counters including GT, CGT, BGT, WT, and BWT. This register may be read at any time. This register must be written to only when C7816[ISO_7816E] is not set.

Address: Base address + 1Dh offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | GTFD | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**UARTx_WF7816 field descriptions**

| Field | Description |
|---|---|
| 7–0<br>GTFD | FD Multiplier<br><br>Used as another multiplier in the calculation of WT and BWT. This value represents a number between 1 and 255. The value of 0 is invalid. This value is not used in baud rate generation. See Wait time and guard time parameters and Baud rate generation . |

## 51.3.30 UART 7816 Error Threshold Register (UARTx_ET7816)

The ET7816 register contains fields that determine the number of NACKs that must be received or transmitted before the host processor is notified. This register may be read at anytime. This register must be written to only when C7816[ISO_7816E] is not set.

Address: Base address + 1Eh offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | TXTHRESHOLD | | | | RXTHRESHOLD | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_ET7816 field descriptions**

| Field | Description |
|---|---|
| 7–4<br>TXTHRESHOLD | Transmit NACK Threshold<br><br>The value written to this field indicates the maximum number of failed attempts (NACKs) a transmitted character can have before the host processor is notified. This field is meaningful only when |

*Table continues on the next page...*

**UARTx_ET7816 field descriptions (continued)**

| Field | Description |
|---|---|
| | C7816[TTYPE] = 0 and C7816[ANACK] = 1. The value read from this field represents the number of consecutive NACKs that have been received since the last successful transmission. This counter saturates at 4'hF and does not wrap around. Regardless of how many NACKs that are received, the UART continues to retransmit indefinitely. This flag only asserts when C7816[TTYPE] = 0. For additional information see the IS7816[TXT] field description.<br><br>0 TXT asserts on the first NACK that is received.<br>1 TXT asserts on the second NACK that is received. |
| 3–0<br>RXTHRESHOLD | Receive NACK Threshold<br><br>The value written to this field indicates the maximum number of consecutive NACKs generated as a result of a parity error or receiver buffer overruns before the host processor is notified. After the counter exceeds that value in the field, the IS7816[RXT] is asserted. This field is meaningful only when C7816[TTYPE] = 0. The value read from this field represents the number of consecutive NACKs that have been transmitted since the last successful reception. This counter saturates at 4'hF and does not wrap around. Regardless of the number of NACKs sent, the UART continues to receive valid packets indefinitely. For additional information, see IS7816[RXT] field description. |

## 51.3.31 UART 7816 Transmit Length Register (UARTx_TL7816)

The TL7816 register is used to indicate the number of characters contained in the block being transmitted. This register is used only when C7816[TTYPE] = 1. This register may be read at anytime. This register must be written only when C2[TE] is not enabled.

Address: Base address + 1Fh offset

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Read<br>Write | TLEN | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**UARTx_TL7816 field descriptions**

| Field | Description |
|---|---|
| 7–0<br>TLEN | Transmit Length<br><br>This value plus four indicates the number of characters contained in the block being transmitted. This register is automatically decremented by 1 for each character in the information field portion of the block. Additionally, this register is automatically decremented by 1 for the first character of a CRC in the epilogue field. Therefore, this register must be programmed with the number of bytes in the data packet if an LRC is being transmitted, and the number of bytes + 1 if a CRC is being transmitted. This register is not decremented for characters that are assumed to be part of the Prologue field, that is, the first three characters transmitted in a block, or the LRC or last CRC character in the Epilogue field, that is, the last character transmitted. This field must be programed or adjusted only when C2[TE] is cleared. |

## 51.4 Functional description

This section provides a complete functional description of the UART block.

The UART allows full duplex, asynchronous, NRZ serial communication between the CPU and remote devices, including other CPUs. The UART transmitter and receiver operate independently, although they use the same baud rate generator. The CPU monitors the status of the UART, writes the data to be transmitted, and processes received data.

### 51.4.1 Transmitter

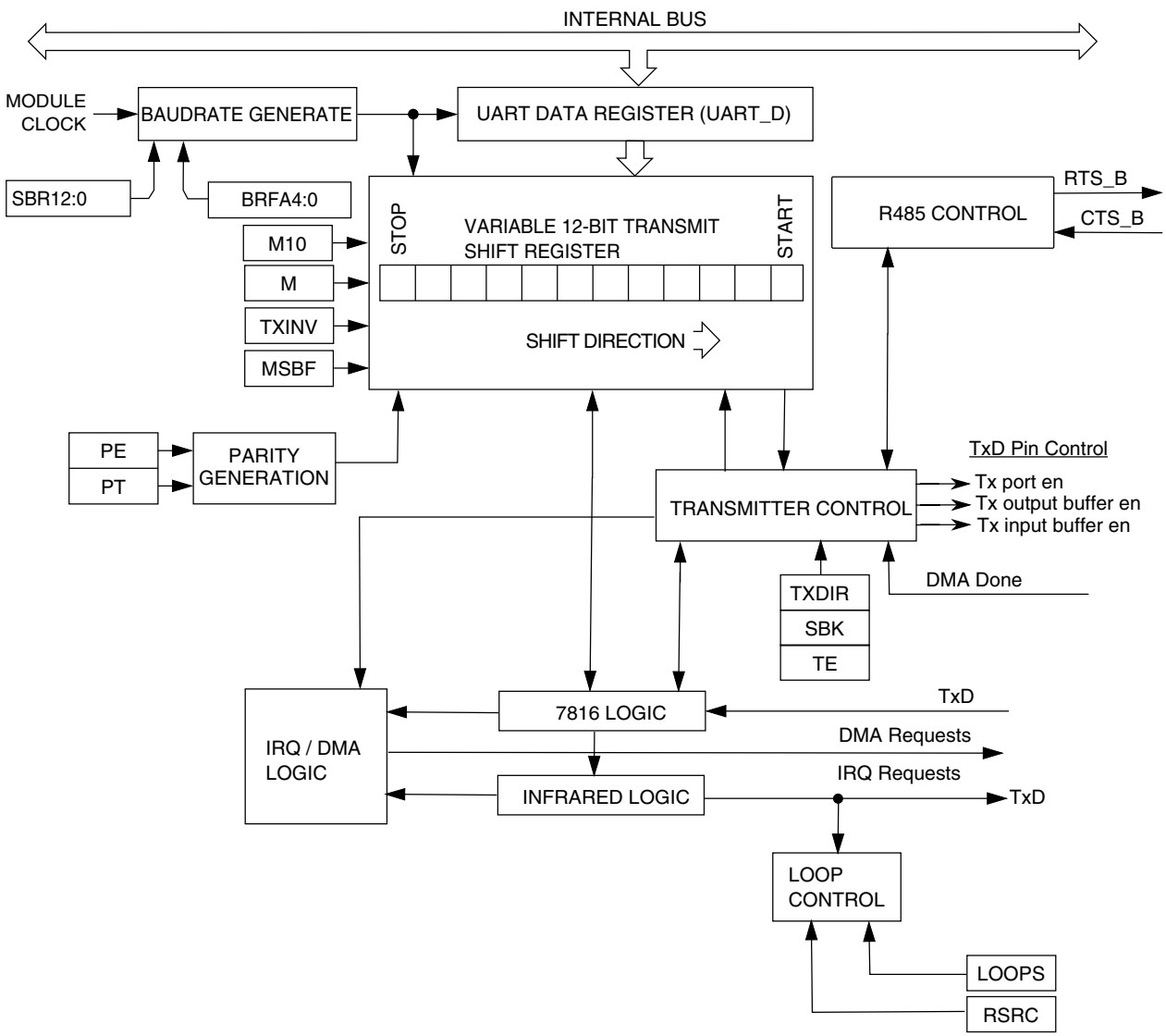



**Figure 51-218. Transmitter Block Diagram**

#### 51.4.1.1 Transmitter character length

The UART transmitter can accommodate either 8, 9, or 10-bit data characters. The state of the C1[M] and C1[PE] bits and the C4[M10] bit determine the length of data characters. When transmitting 9-bit data, bit C3[T8] is the ninth bit (bit 8).

#### 51.4.1.2 Transmission bit order

When S2[MSBF] is set, the UART automatically transmits the MSB of the data word as the first bit after the start bit. Similarly, the LSB of the data word is transmitted immediately preceding the parity bit, or the stop bit if parity is not enabled. All necessary bit ordering is handled automatically by the module. Therefore, the format of the data written to D for transmission is completely independent of the S2[MSBF] setting.

#### 51.4.1.3 Character transmission

To transmit data, the MCU writes the data bits to the UART transmit buffer using UART data registers C3[T8] and D. Data in the transmit buffer is then transferred to the transmitter shift register as needed. The transmit shift register then shifts a frame out through the transmit data output signal after it has prefaced it with any required start and stop bits. The UART data registers, C3[T8] and D, provide access to the transmit buffer structure.

The UART also sets a flag, the transmit data register empty flag S1[TDRE], and generates an interrupt or DMA request (C5[TDMAS]) whenever the number of datawords in the transmit buffer is equal to or less than the value indicated by TWFIFO[TXWATER]. The transmit driver routine may respond to this flag by writing additional datawords to the transmit buffer using C3[T8]/D as space permits.

See Application information for specific programing sequences.

Setting C2[TE] automatically loads the transmit shift register with the following preamble:

- 10 logic 1s if C1[M] = 0
- 11 logic 1s if C1[M] = 1 and C4[M10] = 0
- 12 logic 1s if C1[M] = 1, C4[M10] = 1, C1[PE] = 1

After the preamble shifts out, control logic transfers the data from the D register into the transmit shift register. The transmitter automatically transmits the correct start bit and stop bit before and after the dataword.

When C7816[ISO_7816E] = 1, setting C2[TE] does not result in a preamble being generated. The transmitter starts transmitting as soon as the corresponding guard time expires. When C7816[TTYPE] = 0, the value in GT is used. When C7816[TTYPE] = 1, the value in BGT is used, because C2[TE] will remain asserted until the end of the block transfer. C2[TE] is automatically cleared when C7816[TTYPE] = 1 and the block being transmitted has completed. When C7816[TTYPE] = 0, the transmitter listens for a NACK indication. If no NACK is received, it is assumed that the character was correctly received. If a NACK is received, the transmitter resends the data, assuming that the number of retries for that character, that is, the number of NACKs received, is less than or equal to the value in ET7816[TXTHRESHOLD].

Hardware supports odd or even parity. When parity is enabled, the bit immediately preceding the stop bit is the parity bit.

When the transmit shift register is not transmitting a frame, the transmit data output signal goes to the idle condition, logic 1. If at any time software clears C2[TE], the transmitter enable signal goes low and the transmit signal goes idle.

If the software clears C2[TE] while a transmission is in progress, the character in the transmit shift register continues to shift out, provided S1[TC] was cleared during the data write sequence. To clear S1[TC], the S1 register must be read followed by a write to D register.

If S1[TC] is cleared during character transmission and C2[TE] is cleared, the transmission enable signal is deasserted at the completion of the current frame. Following this, the transmit data out signal enters the idle state even if there is data pending in the UART transmit data buffer. To ensure that all the data written in the FIFO is transmitted on the link before clearing C2[TE], wait for S1[TC] to set. Alternatively, the same can be achieved by setting TWFIFO[TXWATER] to 0x0 and waiting for S1[TDRE] to set.

#### 51.4.1.4 Transmitting break characters

Setting C2[SBK] loads the transmit shift register with a break character. A break character contains all logic 0s and has no start, stop, or parity bit. Break character length depends on C1[M], C1[PE], S2[BRK13] and C4[M10]. See the following table.

**Table 51-227. Transmit break character length**

| S2[BRK13] | C1[M] | C4[M10] | C1[PE] | Bits transmitted |
|---|---|---|---|---|
| 0 | 0 | — | — | 10 |
| 0 | 1 | 1 | 0 | 11 |
| 0 | 1 | 1 | 1 | 12 |
| 1 | 0 | — | — | 13 |
| 1 | 1 | — | — | 14 |

As long as C2[SBK] is set, the transmitter logic continuously loads break characters into the transmit shift register. After the software clears C2[SBK], the shift register finishes transmitting the last break character and then transmits at least one logic 1. The automatic logic 1 at the end of a break character guarantees the recognition of the start bit of the next character. Break bits are not supported when C7816[ISO_7816E] is set/enabled.

**NOTE**

When queuing a break character, it will be transmitted following the completion of the data value currently being shifted out from the shift register. This means that, if data is queued in the data buffer to be transmitted, the break character preempts that queued data. The queued data is then transmitted after the break character is complete.

#### 51.4.1.5 Idle characters

An idle character contains all logic 1s and has no start, stop, or parity bit. Idle character length depends on C1[M], C1[PE] and C4[M10]. The preamble is a synchronizing idle character that begins the first transmission initiated after setting C2[TE]. When C7816[ISO_7816E] is set/enabled, idle characters are not sent or detected. When data is not being transmitted, the data I/O line is in an inactive state.

If C2[TE] is cleared during a transmission, the transmit data output signal becomes idle after completion of the transmission in progress. Clearing and then setting C2[TE] during a transmission queues an idle character to be sent after the dataword currently being transmitted.

**Note**

When queuing an idle character, the idle character will be transmitted following the completion of the data value currently being shifted out from the shift register. This means that if data is queued in the data buffer to be transmitted, the idle character preempts that queued data. The queued data is then transmitted after the idle character is complete.

If C2[TE] is cleared and the transmission is completed, the UART is not the master of the TXD pin.

### 51.4.1.6 Hardware flow control

The transmitter supports hardware flow control by gating the transmission with the value of CTS. If the clear-to-send operation is enabled, the character is transmitted when CTS is asserted. If CTS is deasserted in the middle of a transmission with characters remaining in the receiver data buffer, the character in the shift register is sent and TXD remains in the mark state until CTS is reasserted.

If the clear-to-send operation is disabled, the transmitter ignores the state of CTS. Also, if the transmitter is forced to send a continuous low condition because it is sending a break character, the transmitter ignores the state of CTS regardless of whether the clear-to-send operation is enabled.

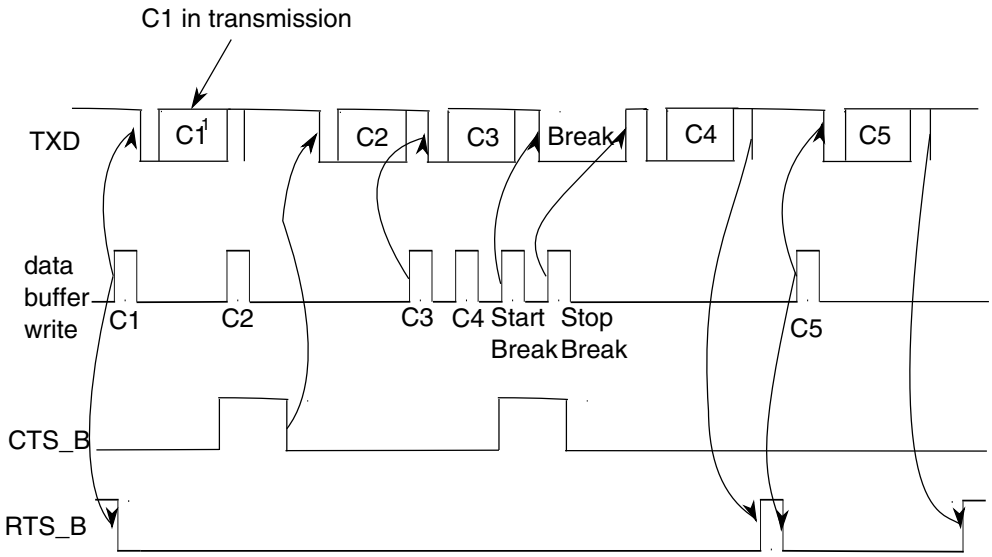
The transmitter's CTS signal can also be enabled even if the same UART receiver's RTS signal is disabled.

### 51.4.1.7 Transceiver driver enable

The transmitter can use RTS as an enable signal for the driver of an external transceiver. See Transceiver driver enable using RTS for details. If the request-to-send operation is enabled, when a character is placed into an empty transmitter data buffer, RTS asserts one bit time before the start bit is transmitted. RTS remains asserted for the whole time that the transmitter data buffer has any characters. RTS deasserts one bit time after all characters in the transmitter data buffer and shift register are completely sent, including the last stop bit. Transmitting a break character also asserts RTS, with the same assertion and deassertion timing as having a character in the transmitter data buffer.

The transmitter's RTS signal asserts only when the transmitter is enabled. However, the transmitter's RTS signal is unaffected by its CTS signal. RTS will remain asserted until the transfer is completed, even if the transmitter is disabled mid-way through a data transfer.

The following figure shows the functional timing information for the transmitter. Along with the actual character itself, TXD shows the start bit. The stop bit is also indicated, with a dashed line if necessary.




1. Cn = transmit characters

**Figure 51-219. Transmitter RTS and CTS timing diagram**

## 51.4.2 Receiver

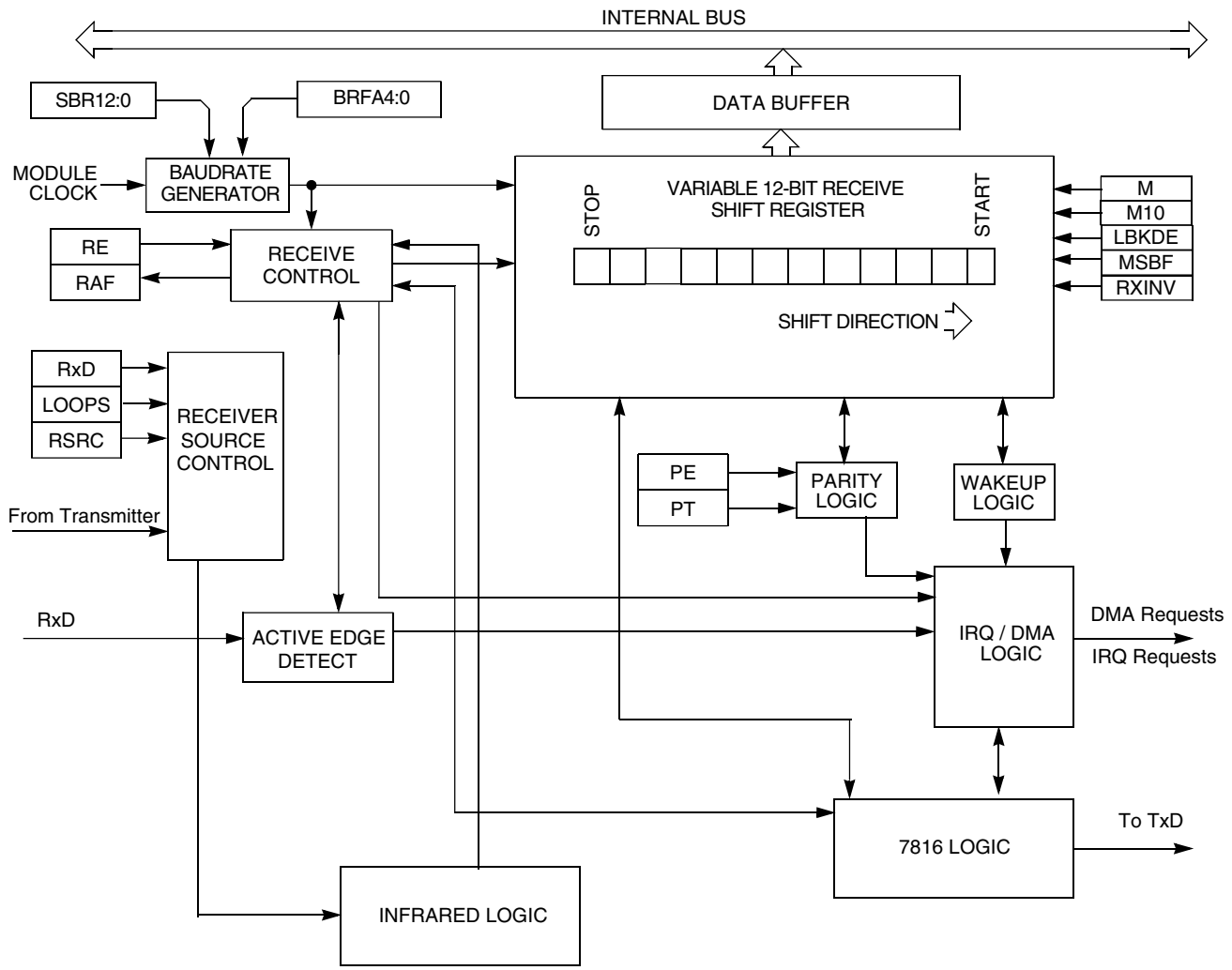



**Figure 51-220. UART receiver block diagram**

### 51.4.2.1 Receiver character length

The UART receiver can accommodate 8-, 9-, or 10-bit data characters. The states of C1[M], C1[PE] and C4[M10] determine the length of data characters. When receiving 9 or 10-bit data, C3[R8] is the ninth bit (bit 8).

### 51.4.2.2 Receiver bit ordering

When S2[MSBF] is set, the receiver operates such that the first bit received after the start bit is the MSB of the dataword. Similarly, the bit received immediately preceding the parity bit, or the stop bit if parity is not enabled, is treated as the LSB for the dataword. All necessary bit ordering is handled automatically by the module. Therefore, the format of the data read from receive data buffer is completely independent of S2[MSBF].

### 51.4.2.3 Character reception

During UART reception, the receive shift register shifts a frame in from the unsynchronized receiver input signal. After a complete frame shifts into the receive shift register, the data portion of the frame transfers to the UART receive buffer. Additionally, the noise and parity error flags that are calculated during the receive process are also captured in the UART receive buffer. The receive data buffer is accessible via the D and C3[T8] registers. Additional received information flags regarding the receive dataword can be read in ED register. S1[RDRF] is set if the number of resulting datawords in the receive buffer is equal to or greater than the number indicated by RWFIFO[RXWATER]. If the C2[RIE] is also set, RDRF generates an RDRF interrupt request. Alternatively, by programming C5[RDMAS], a DMA request can be generated.

When C7816[ISO_7816E] is set/enabled and C7816[TTYPE] = 0, character reception operates slightly differently. Upon receipt of the parity bit, the validity of the parity bit is checked. If C7816[ANACK] is set and the parity check fails, or if INIT and the received character is not a valid initial character, then a NACK is sent by the receiver. If the number of consecutive receive errors exceeds the threshold set by ET7816[RXTHRESHOLD], then IS7816[RXT] is set and an interrupt generated if IE7816[RXTE] is set. If an error is detected due to parity or an invalid initial character, the data is not transferred from the receive shift register to the receive buffer. Instead, the data is overwritten by the next incoming data.

When the C7816[ISO_7816E] is set/enabled, C7816[ONACK] is set/enabled, and the received character results in the receive buffer overflowing, a NACK is issued by the receiver. Additionally, S1[OR] is set and an interrupt is issued if required, and the data in the shift register is discarded.

### 51.4.2.4 Data sampling

The receiver samples the unsynchronized receiver input signal at the RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch, the RT clock (see the following figure) is re-synchronized:

- After every start bit.
- After the receiver detects a data bit change from logic 1 to logic 0 (after the majority of data bit samples at RT8, RT9, and RT10 returns a valid logic 1 and the majority of the next RT8, RT9, and RT10 samples returns a valid logic 0).

To locate the start bit, data recovery logic does an asynchronous search for a logic 0 preceded by three logic 1s. When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.

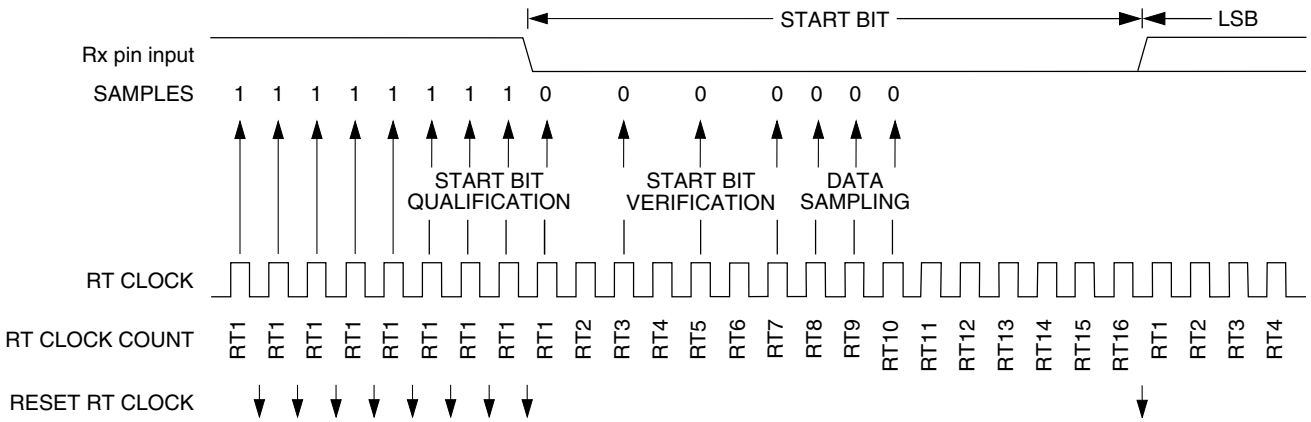



**Figure 51-221. Receiver data sampling**

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7 when C7816[ISO_7816E] is cleared/disabled and RT8, RT9 and RT10 when C7816[ISO_7816E] is set/enabled. The following table summarizes the results of the start bit verification samples.

**Table 51-228. Start bit verification**

| RT3, RT5, and RT7 samples<br>RT8, RT9, RT10 samples when 7816E | Start bit verification | Noise flag |
|---|---|---|
| 000 | Yes | 0 |
| 001 | Yes | 1 |
| 010 | Yes | 1 |
| 011 | No | 0 |
| 100 | Yes | 1 |
| 101 | No | 0 |
| 110 | No | 0 |
| 111 | No | 0 |

If start bit verification is not successful, the RT clock is reset and a new search for a start bit begins.

To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. The following table summarizes the results of the data bit samples.

**Table 51-229. Data bit recovery**

| RT8, RT9, and RT10 samples | Data bit determination | Noise flag |
|---|---|---|
| 000 | 0 | 0 |
| 001 | 0 | 1 |
| 010 | 0 | 1 |
| 011 | 1 | 1 |
| 100 | 0 | 1 |
| 101 | 1 | 1 |
| 110 | 1 | 1 |
| 111 | 1 | 0 |

## Note

The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are logic 1s following a successful start bit verification, the noise flag (S1[NF]) is set and the receiver assumes that the bit is a start bit (logic 0). With the exception of when C7816[ISO_7816E] is set/enabled, where the values of RT8, RT9 and RT10 exclusively determine if a start bit exists.

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. The following table summarizes the results of the stop bit samples. In the event that C7816[ISO_7816E] is set/enabled and C7816[TTYPE] = 0, verification of a stop bit does not take place. Rather, starting with RT8 the receiver transmits a NACK as programmed until time RT9 of the following time period. Framing Error detection is not supported when C7816[ISO_7816E] is set/enabled.

**Table 51-230. Stop bit recovery**

| RT8, RT9, and RT10 samples | Framing error flag | Noise flag |
|---|---|---|
| 000 | 1 | 0 |
| 001 | 1 | 1 |
| 010 | 1 | 1 |
| 011 | 0 | 1 |
| 100 | 1 | 1 |
| 101 | 0 | 1 |
| 110 | 0 | 1 |
| 111 | 0 | 0 |

In the following figure, the verification samples RT3 and RT5 determine that the first low detected was noise and not the beginning of a start bit. In this example C7816[ISO_7816E] = 0. The RT clock is reset and the start bit search begins again. The noise flag is not set because the noise occurred before the start bit was found.

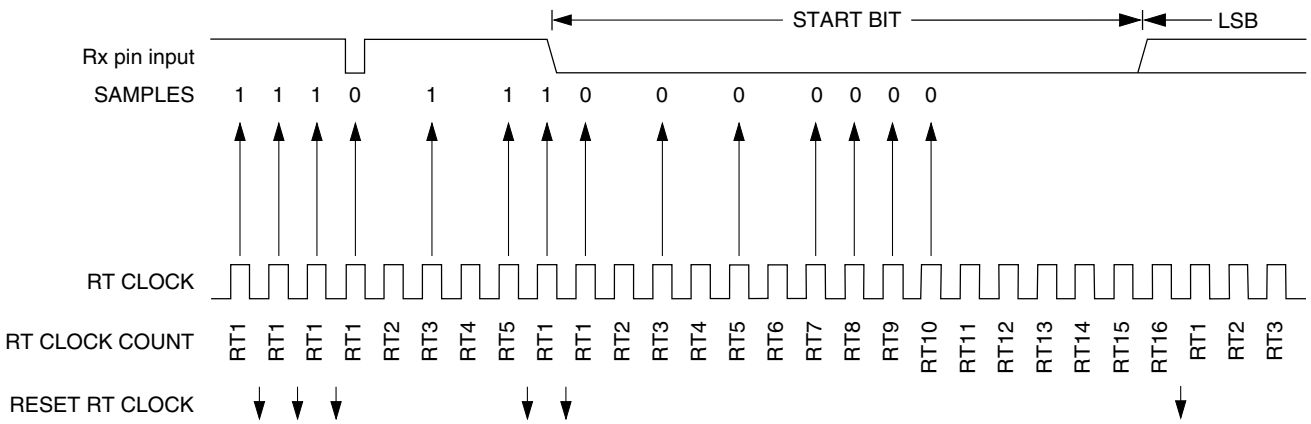



**Figure 51-222. Start bit search example 1 (C7816[ISO_7816E] = 0)**

In the following figure, verification sample at RT3 is high. In this example C7816[ISO_7816E] = 0. The RT3 sample sets the noise flag. Although the perceived bit time is misaligned, the data samples RT8, RT9, and RT10 are within the bit time and data recovery is successful.

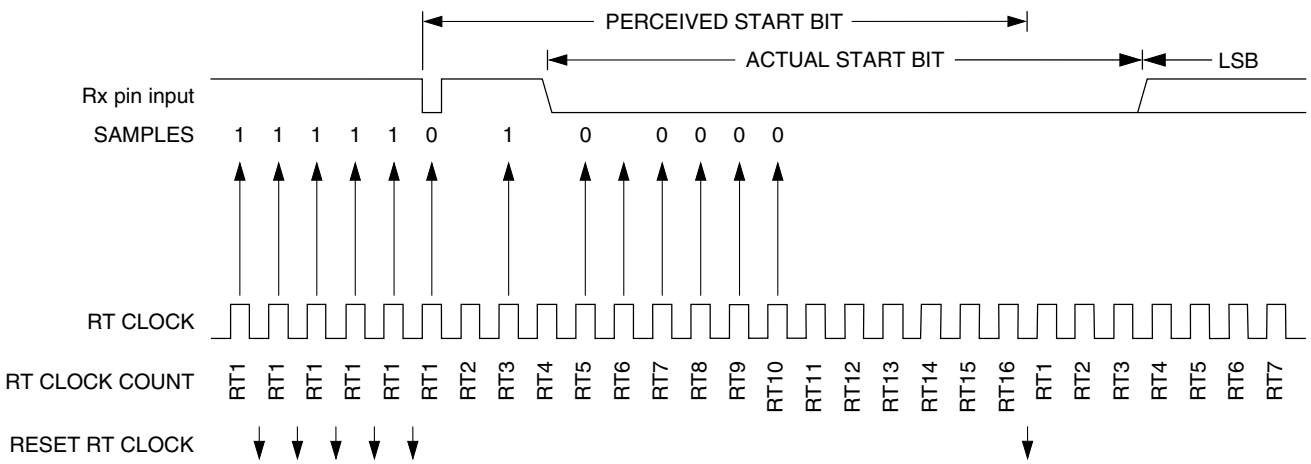



**Figure 51-223. Start bit search example 2 (C7816[ISO_7816E] = 0)**

In the following figure, a large burst of noise is perceived as the beginning of a start bit, although the test sample at RT5 is high. In this example C7816[ISO_7816E] = 0. The RT5 sample sets the noise flag. Although this is a worst-case misalignment of perceived bit time, the data samples RT8, RT9, and RT10 are within the bit time and data recovery is successful.

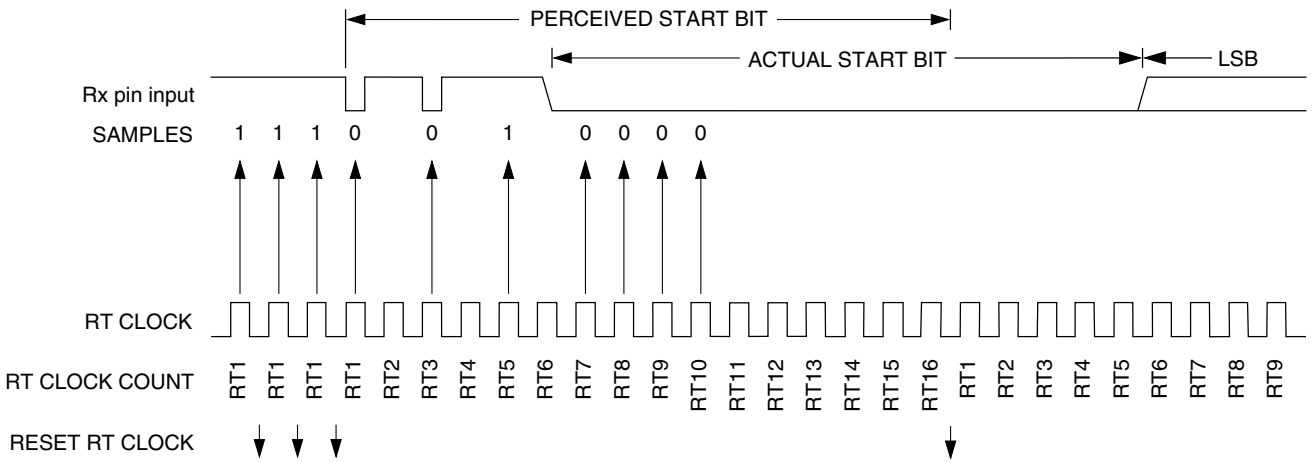



**Figure 51-224. Start bit search example 3 (C7816[ISO_7816E] = 0)**

The following figure shows the effect of noise early in the start bit time. In this example C7816[ISO_7816E] = 0. Although this noise does not affect proper synchronization with the start bit time, it does set the noise flag.

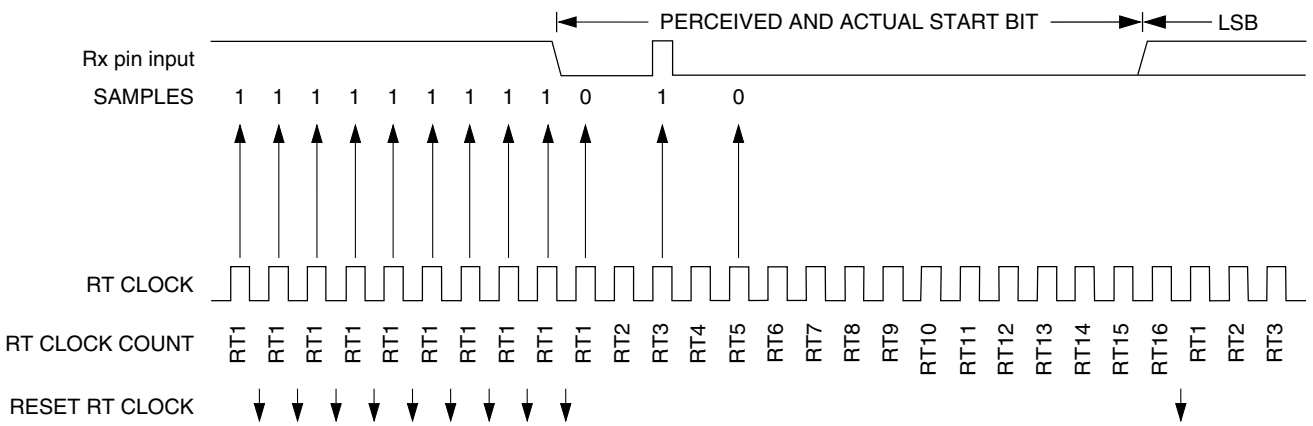



**Figure 51-225. Start bit search example 4 (C7816[ISO_7816E] = 0)**

The following figure shows a burst of noise near the beginning of the start bit that resets the RT clock. In this example C7816[ISO_7816E] = 0. The sample after the reset is low but is not preceded by three high samples that would qualify as a falling edge. Depending on the timing of the start bit search and on the data, the frame may be missed entirely or it may set the framing error flag.

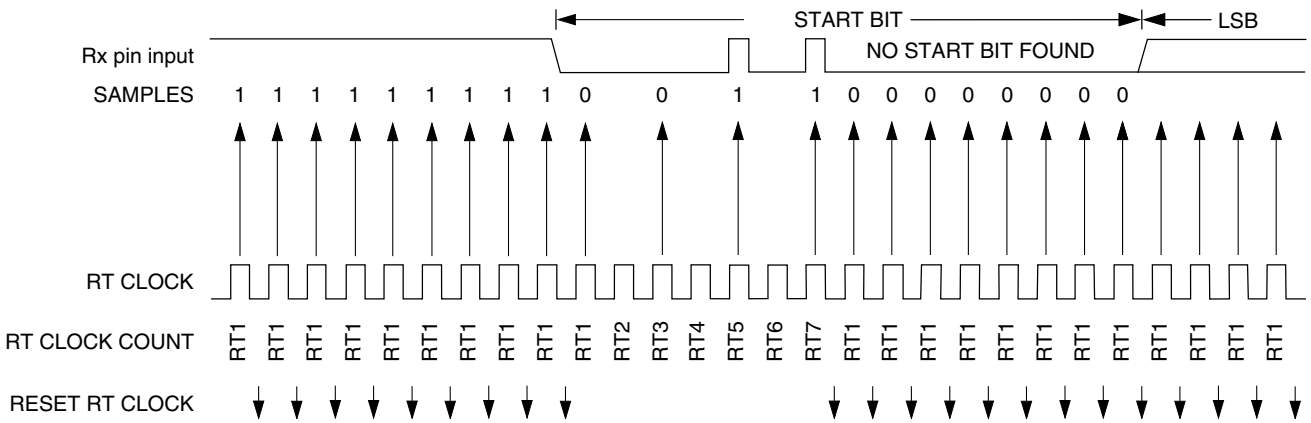



**Figure 51-226. Start bit search example 5 (C7816[ISO_7816E] = 0)**

In the following figure, a noise burst makes the majority of data samples RT8, RT9, and RT10 high. In this example C7816[ISO_7816E] = 0. This sets the noise flag but does not reset the RT clock. In start bits only, the RT8, RT9, and RT10 data samples are ignored. In this example, if C7816[ISO_7816E] = 1 then a start bit would not have been detected at all since at least two of the three samples (RT8, RT9, RT10) were high.

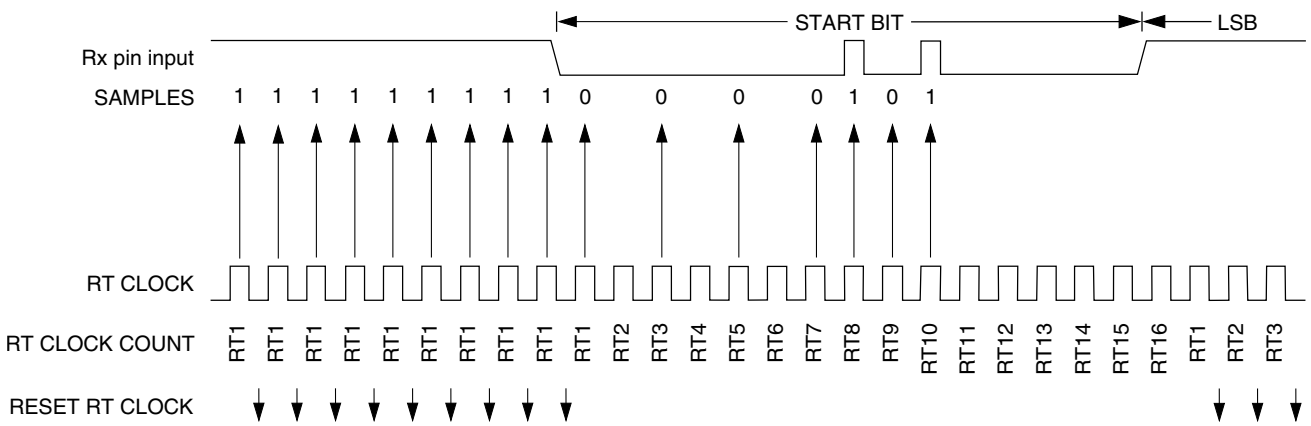



**Figure 51-227. Start bit search example 6**

## 51.4.2.5 Framing errors

If the data recovery logic does not detect a logic 1 where the stop bit should be in an incoming frame, it sets the framing error flag, S1[FE], if S2[LBKDE] is disabled. When S2[LBKDE] is disabled, a break character also sets the S1[FE] because a break character has no stop bit. S1[FE] is set at the same time that received data is placed in the receive data buffer. Framing errors are not supported when C7816[ISO7816E] is set/enabled. However, if S1[FE] is set, data will not be received when C7816[ISO7816E] is set.

### 51.4.2.6 Receiving break characters

The UART recognizes a break character when a start bit is followed by eight, nine, or ten logic 0 data bits and a logic 0 where the stop bit should be. Receiving a break character has these effects on UART registers:

- Sets the framing error flag, S1[FE].
- Writes an all 0 dataword to the data buffer, which may cause S1[RDRF] to set, depending on the watermark and number of values in the data buffer.
- May set the overrun flag, S1[OR], noise flag, S1[NF], parity error flag, S1[PE], or the receiver active flag, S2[RAF].

The detection threshold for a break character can be adjusted when using an internal oscillator in a LIN system by setting S2[LBKDE]. The UART break character detection threshold depends on C1[M], C1[PE], S2[LBKDE] and C4[M10]. See the following table.

**Table 51-231. Receive break character detection threshold**

| LBKDE | M | M10 | PE | Threshold (bits) |
|---|---|---|---|---|
| 0 | 0 | — | — | 10 |
| 0 | 1 | 0 | — | 11 |
| 0 | 1 | 1 | 1 | 12 |
| 1 | 0 | — | — | 11 |
| 1 | 1 | — | — | 12 |

While S2[LBKDE] is set, it will have these effects on the UART registers:

- Prevents S1[RDRF], S1[FE], S1[NF], and S1[PF] from being set. However, if they are already set, they will remain set.
- Sets the LIN break detect interrupt flag, S2[LBKDIF], if a LIN break character is received.

Break characters are not detected or supported when C7816[ISO_7816E] is set/enabled.

### 51.4.2.7 Hardware flow control

To support hardware flow control, the receiver can be programmed to automatically deassert and assert RTS.

- RTS remains asserted until the transfer is complete, even if the transmitter is disabled midway through a data transfer. See Transceiver driver enable using RTS for more details.
- If the receiver request-to-send functionality is enabled, the receiver automatically deasserts RTS if the number of characters in the receiver data register is equal to or greater than receiver data buffer's watermark, RWFIFO[RXWATER].
- The receiver asserts RTS when the number of characters in the receiver data register is less than the watermark. It is not affected if RDRF is asserted.
- Even if RTS is deasserted, the receiver continues to receive characters until the receiver data buffer is full or is overrun.
- If the receiver request-to-send functionality is disabled, the receiver RTS remains deasserted.

The following figure shows receiver hardware flow control functional timing. Along with the actual character itself, RXD shows the start bit. The stop bit can also indicated, with a dashed line, if necessary. The watermark is set to 2.

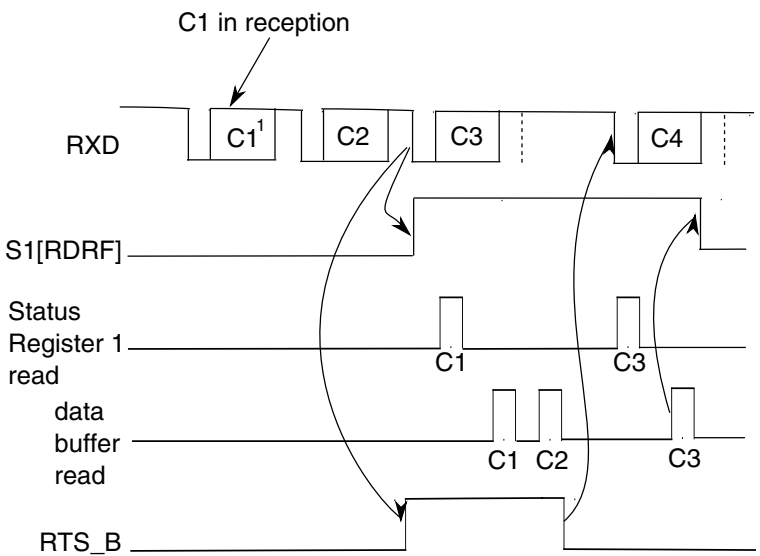



**Figure 51-228. Receiver hardware flow control timing diagram**

## 51.4.2.8 Infrared decoder

The infrared decoder converts the received character from the IrDA format to the NRZ format used by the receiver. It also has a 16-RT clock counter that filters noise and indicates when a 1 is received.

#### 51.4.2.8.1 Start bit detection

When S2[RXINV] is cleared, the first rising edge of the received character corresponds to the start bit. The infrared decoder resets its counter. At this time, the receiver also begins its start bit detection process. After the start bit is detected, the receiver synchronizes its bit times to this start bit time. For the rest of the character reception, the infrared decoder's counter and the receiver's bit time counter count independently from each other.

#### 51.4.2.8.2 Noise filtering

Any further rising edges detected during the first half of the infrared decoder counter are ignored by the decoder. Any pulses less than one RT clocks can be undetected by it regardless of whether it is seen in the first or second half of the count.

#### 51.4.2.8.3 Low-bit detection

During the second half of the decoder count, a rising edge is decoded as a 0, which is sent to the receiver. The decoder counter is also reset.

#### 51.4.2.8.4 High-bit detection

At 16-RT clocks after the previous rising edge, if a rising edge is not seen, then the decoder sends a 1 to the receiver.

If the next bit is a 0, which arrives late, then a low-bit is detected according to Low-bit detection. The value sent to the receiver is changed from 1 to a 0. Then, if a noise pulse occurs outside the receiver's bit time sampling period, then the delay of a 0 is not recorded as noise.

### 51.4.2.9 Baud rate tolerance

A transmitting device may be operating at a baud rate below or above the receiver baud rate. Accumulated bit time misalignment can cause one of the three stop bit data samples (RT8, RT9, and RT10) to fall outside the actual stop bit. A noise error will occur if the RT8, RT9, and RT10 samples are not all the same logical values. A framing error will occur if the receiver clock is misaligned in such a way that the majority of the RT8, RT9, and RT10 stop bit samples are a logic 0.

As the receiver samples an incoming frame, it resynchronizes the RT clock on any valid falling edge within the frame. Resynchronization within frames corrects a misalignment between transmitter bit times and receiver bit times.

#### 51.4.2.9.1 Slow data tolerance

The following figure shows how much a slow received frame can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

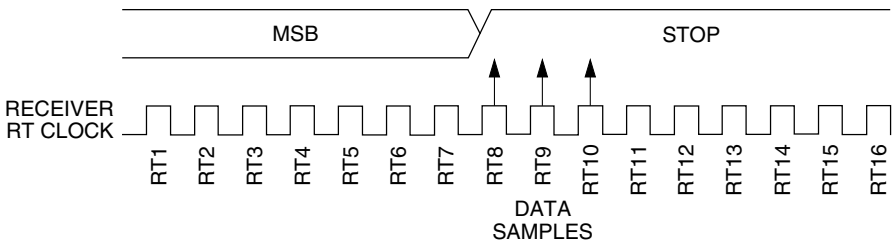



**Figure 51-229. Slow data**

For an 8-bit data character, data sampling of the stop bit takes the receiver 154 RT cycles (9 bit times × 16 RT cycles + 10 RT cycles).

With the misaligned character shown in the Figure 51-229, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 147 RT cycles (9 bit times × 16 RT cycles + 3 RT cycles).

The maximum percent difference between the receiver count and the transmitter count of a slow 8-bit data character with no errors is:

$$((154 - 147) \div 154) \times 100 = 4.54\%$$

For a 9-bit data character, data sampling of the stop bit takes the receiver 170 RT cycles (10 bit times × 16 RT cycles + 10 RT cycles).

With the misaligned character shown in the Figure 51-229, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 163 RT cycles (10 bit times × 16 RT cycles + 3 RT cycles).

The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is:

$$((170 - 163) \div 170) \times 100 = 4.12\%$$

#### 51.4.2.9.2 Fast data tolerance

The following figure shows how much a fast received frame can be misaligned. The fast stop bit ends at RT10 instead of RT16 but is still sampled at RT8, RT9, and RT10.

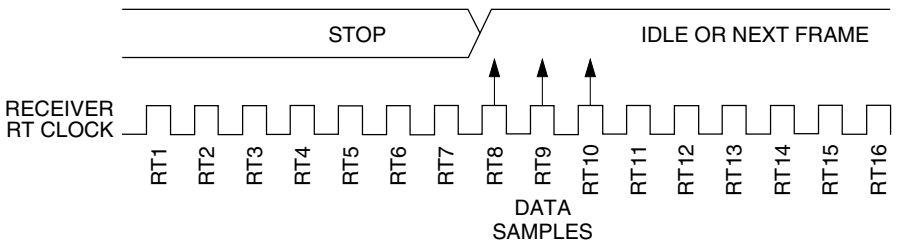



**Figure 51-230. Fast data**

For an 8-bit data character, data sampling of the stop bit takes the receiver 154 RT cycles (9 bit times × 16 RT cycles + 10 RT cycles).

With the misaligned character shown in the Figure 51-230, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 160 RT cycles (10 bit times × 16 RT cycles).

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is:

$$((154 - 160) \div 154) \times 100 = 3.90\%$$

For a 9-bit data character, data sampling of the stop bit takes the receiver 170 RT cycles (10 bit times × 16 RT cycles + 10 RT cycles).

With the misaligned character shown in the Figure 51-230, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 176 RT cycles (11 bit times × 16 RT cycles).

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is:

$$((170 - 176) \div 170) \times 100 = 3.53\%$$

### 51.4.2.10 Receiver wakeup

C1[WAKE] determines how the UART is brought out of the standby state to process an incoming message. C1[WAKE] enables either idle line wakeup or address mark wakeup.

Receiver wakeup is not supported when C7816[ISO_7816E] is set/enabled because multi-receiver systems are not allowed.

#### 51.4.2.10.1 Idle input line wakeup (C1[WAKE] = 0)

In this wakeup method, an idle condition on the unsynchronized receiver input signal clears C2[RWU] and wakes the UART. The initial frame or frames of every message contain addressing information. All receivers evaluate the addressing information, and

receivers for which the message is addressed process the frames that follow. Any receiver for which a message is not addressed can set its C2[RWU] and return to the standby state. C2[RWU] remains set and the receiver remains in standby until another idle character appears on the unsynchronized receiver input signal.

Idle line wakeup requires that messages be separated by at least one idle character and that no message contains idle characters.

When C2[RWU] is 1 and S2[RWUID] is 0, the idle character that wakes the receiver does not set S1[IDLE] or the receive data register full flag, S1[RDRF]. The receiver wakes and waits for the first data character of the next message which is stored in the receive data buffer. When S2[RWUID] and C2[RWU] are set and C1[WAKE] is cleared, any idle condition sets S1[IDLE] and generates an interrupt if enabled.

Idle input line wakeup is not supported when C7816[ISO_7816E] is set/enabled.

#### 51.4.2.10.2 Address mark wakeup (C1[WAKE] = 1)

In this wakeup method, a logic 1 in the bit position immediately preceding the stop bit of a frame clears C2[RWU] and wakes the UART. A logic 1 in the bit position immediately preceeding the stop bit marks a frame as an address frame that contains addressing information. All receivers evaluate the addressing information, and the receivers for which the message is addressed process the frames that follow. Any receiver for which a message is not addressed can set its C2[RWU] and return to the standby state. C2[RWU] remains set and the receiver remains in standby until another address frame appears on the unsynchronized receiver input signal.

A logic 1 in the bit position immediately preceding the stop bit clears the receiver's C2[RWU] after the stop bit is received and places the received data into the receiver data buffer. Note that if Match Address operation is enabled i.e. C4[MAEN1] or C4[MAEN2] is set, then received frame is transferred to receive buffer only if the comparison matches.

Address mark wakeup allows messages to contain idle characters but requires that the bit position immediately preceding the stop bit be reserved for use in address frames.

If module is in standby mode and nothing triggers to wake the UART, no error flag is set even if an invalid error condition is detected on the receiving data line.

Address mark wakeup is not supported when C7816[ISO_7816E] is set/enabled.

#### 51.4.2.10.3 Match address operation

Match address operation is enabled when C4[MAEN1] or C4[MAEN2] is set. In this function, a frame received by the RX pin with a logic 1 in the bit position of the address mark is considered an address and is compared with the associated MA1 or MA2 register.

The frame is transferred to the receive buffer, and S1[RDRF] is set, only if the comparison matches. All subsequent frames received with a logic 0 in the bit position of the address mark are considered to be data associated with the address and are transferred to the receive data buffer. If no marked address match occurs, then no transfer is made to the receive data buffer, and all following frames with logic 0 in the bit position of the address mark are also discarded. If both C4[MAEN1] and C4[MAEN2] are negated, the receiver operates normally and all data received is transferred to the receive data buffer.

Match address operation functions in the same way for both MA1 and MA2 registers. Note that the position of the address mark is the same as the Parity Bit when parity is enabled for 8 bit and 9 bit data formats.

- If only one of C4[MAEN1] and C4[MAEN2] is asserted, a marked address is compared only with the associated match register and data is transferred to the receive data buffer only on a match.
- If C4[MAEN1] and C4[MAEN2] are asserted, a marked address is compared with both match registers and data is transferred only on a match with either register.

Address match operation is not supported when C7816[ISO_7816E] is set/enabled.

### 51.4.3 Baud rate generation

A 13-bit modulus counter and a 5-bit fractional fine-adjust counter in the baud rate generator derive the baud rate for both the receiver and the transmitter. The value from 1 to 8191 written to SBR[12:0] determines the module clock divisor. The SBR bits are in the UART baud rate registers, BDH and BDL. The baud rate clock is synchronized with the module clock and drives the receiver. The fractional fine-adjust counter adds fractional delays to the baud rate clock to allow fine trimming of the baud rate to match the system baud rate. The transmitter is driven by the baud rate clock divided by 16. The receiver has an acquisition rate of 16 samples per bit time.

Baud rate generation is subject to two sources of error:

- Integer division of the module clock may not give the exact target frequency. This error can be reduced with the fine-adjust counter.
- Synchronization with the module clock can cause phase shift.

The Table 51-232 lists the available baud divisor fine adjust values.

UART baud rate = UART module clock / (16 × (SBR[12:0] + BRFD))

The following table lists some examples of achieving target baud rates with a module clock frequency of 10.2 MHz, with and without fractional fine adjustment.

**Table 51-232. Baud rates (example: module clock = 10.2 MHz)**

| Bits SBR (decimal) | Bits BRFA | BRFD value | Receiver clock (Hz) | Transmitter clock (Hz) | Target Baud rate | Error (%) |
|---|---|---|---|---|---|---|
| 17 | 00000 | 0 | 600,000.0 | 37,500.0 | 38,400 | 2.3 |
| 16 | 10011 | 19/32=0.59375 | 614,689.3 | 38,418.08 | 38,400 | 0.047 |
| 33 | 00000 | 0 | 309,090.9 | 19,318.2 | 19,200 | 0.62 |
| 33 | 00110 | 6/32=0.1875 | 307,344.6 | 19,209.04 | 19,200 | 0.047 |
| 66 | 00000 | 0 | 154,545.5 | 9659.1 | 9600 | 0.62 |
| 133 | 00000 | 0 | 76,691.7 | 4793.2 | 4800 | 0.14 |
| 266 | 00000 | 0 | 38,345.9 | 2396.6 | 2400 | 0.14 |
| 531 | 00000 | 0 | 19,209.0 | 1200.6 | 1200 | 0.11 |
| 1062 | 00000 | 0 | 9604.5 | 600.3 | 600 | 0.05 |
| 2125 | 00000 | 0 | 4800.0 | 300.0 | 300 | 0.00 |
| 4250 | 00000 | 0 | 2400.0 | 150.0 | 150 | 0.00 |
| 5795 | 00000 | 0 | 1760.1 | 110.0 | 110 | 0.00 |

**Table 51-233. Baud rate fine adjust**

| BRFA | Baud Rate Fractional Divisor (BRFD) |
|---|---|
| 0 0 0 0 0 | 0/32 = 0 |
| 0 0 0 0 1 | 1/32 = 0.03125 |
| 0 0 0 1 0 | 2/32 = 0.0625 |
| 0 0 0 1 1 | 3/32 = 0.09375 |
| 0 0 1 0 0 | 4/32 = 0.125 |
| 0 0 1 0 1 | 5/32 = 0.15625 |
| 0 0 1 1 0 | 6/32 = 0.1875 |
| 0 0 1 1 1 | 7/32 = 0.21875 |
| 0 1 0 0 0 | 8/32 = 0.25 |
| 0 1 0 0 1 | 9/32 = 0.28125 |
| 0 1 0 1 0 | 10/32 = 0.3125 |
| 0 1 0 1 1 | 11/32 = 0.34375 |
| 0 1 1 0 0 | 12/32 = 0.375 |
| 0 1 1 0 1 | 13/32 = 0.40625 |
| 0 1 1 1 0 | 14/32 = 0.4375 |
| 0 1 1 1 1 | 15/32 = 0.46875 |
| 1 0 0 0 0 | 16/32 = 0.5 |
| 1 0 0 0 1 | 17/32 = 0.53125 |
| 1 0 0 1 0 | 18/32 = 0.5625 |

*Table continues on the next page...*

**Table 51-233. Baud rate fine adjust (continued)**

| BRFA | Baud Rate Fractional Divisor (BRFD) |
|---|---|
| 1 0 0 1 1 | 19/32 = 0.59375 |
| 1 0 1 0 0 | 20/32 = 0.625 |
| 1 0 1 0 1 | 21/32 = 0.65625 |
| 1 0 1 1 0 | 22/32 = 0.6875 |
| 1 0 1 1 1 | 23/32 = 0.71875 |
| 1 1 0 0 0 | 24/32 = 0.75 |
| 1 1 0 0 1 | 25/32 = 0.78125 |
| 1 1 0 1 0 | 26/32 = 0.8125 |
| 1 1 0 1 1 | 27/32 = 0.84375 |
| 1 1 1 0 0 | 28/32 = 0.875 |
| 1 1 1 0 1 | 29/32 = 0.90625 |
| 1 1 1 1 0 | 30/32 = 0.9375 |
| 1 1 1 1 1 | 31/32 = 0.96875 |

## 51.4.4 Data format (non ISO-7816)

Each data character is contained in a frame that includes a start bit and a stop bit. The rest of the data format depends upon C1[M], C1[PE], S2[MSBF] and C4[M10].

### 51.4.4.1 Eight-bit configuration

Clearing C1[M] configures the UART for 8-bit data characters, that is, eight bits are memory mapped in D. A frame with eight data bits has a total of 10 bits. The most significant bit of the eight data bits can be used as an address mark to wake the receiver. If the most significant bit is used in this way, then it serves as an address or data indication, leaving the remaining seven bits as actual data. When C1[PE] is set, the eighth data bit is automatically calculated as the parity bit. See the following table.

**Table 51-234. Configuration of 8-bit data format**

| UART_C1[PE] | Start bit | Data bits | Address bits | Parity bits | Stop bit |
|---|---|---|---|---|---|
| 0 | 1 | 8 | 0 | 0 | 1 |
| 0 | 1 | 7 | 1[1] | 0 | 1 |
| 1 | 1 | 7 | 0 | 1 | 1 |

1. The address bit identifies the frame as an address character. See Receiver wakeup.

### 51.4.4.2 Nine-bit configuration

When C1[M] is set and C4[M10] is cleared, the UART is configured for 9-bit data characters. If C1[PE] is enabled, the ninth bit is either C3[T8/R8] or the internally generated parity bit. This results in a frame consisting of a total of 11 bits. In the event that the ninth data bit is selected to be C3[T8], it will remain unchanged after transmission and can be used repeatedly without rewriting it, unless the value needs to be changed. This feature may be useful when the ninth data bit is being used as an address mark.

When C1[M] and C4[M10] are set, the UART is configured for 9-bit data characters, but the frame consists of a total of 12 bits. The 12 bits include the start and stop bits, the 9 data character bits, and a tenth internal data bit. Note that if C4[M10] is set, C1[PE] must also be set. In this case, the tenth bit is the internally generated parity bit. The ninth bit can either be used as an address mark or a ninth data bit.

See the following table.

**Table 51-235. Configuration of 9-bit data formats**

| C1[PE] | UC1[M] | C1[M10] | Start bit | Data bits | Address bits | Parity bits | Stop bit |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | See Eight-bit configuration | | | | |
| 0 | 0 | 1 | Invalid configuration | | | | |
| 0 | 1 | 0 | 1 | 9 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 8 | 1[1] | 0 | 1 |
| 0 | 1 | 1 | Invalid Configuration | | | | |
| 1 | 0 | 0 | See Eight-bit configuration | | | | |
| 1 | 0 | 1 | Invalid Configuration | | | | |
| 1 | 1 | 0 | 1 | 8 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 9 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 8 | 1[2] | 1 | 1 |

1. The address bit identifies the frame as an address character.
2. The address bit identifies the frame as an address character.

**Note**

Unless in 9-bit mode with M10 set, do not use address mark wakeup with parity enabled.

#### 51.4.4.3 Timing examples

Timing examples of these configurations in the NRZ mark/space data format are illustrated in the following figures. The timing examples show all of the configurations in the following sub-sections along with the LSB and MSB first variations.

##### 51.4.4.3.1 Eight-bit format with parity disabled

The most significant bit can be used for address mark wakeup.

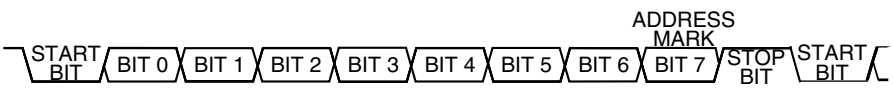



**Figure 51-231. Eight bits of data with LSB first**

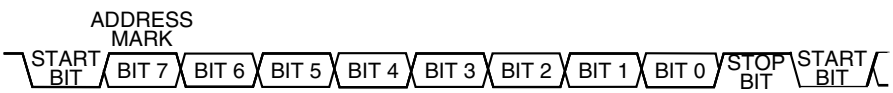



**Figure 51-232. Eight bits of data with MSB first**

##### 51.4.4.3.2 Eight-bit format with parity enabled

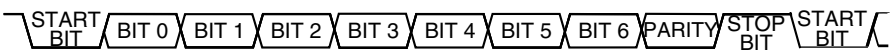



**Figure 51-233. Seven bits of data with LSB first and parity**

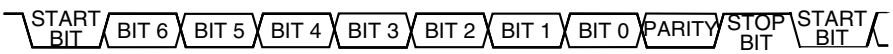



**Figure 51-234. Seven bits of data with MSB first and parity**

##### 51.4.4.3.3 Nine-bit format with parity disabled

The most significant bit can be used for address mark wakeup.


**Figure 51-235. Nine bits of data with LSB first**


**Figure 51-236. Nine bits of data with MSB first**

#### 51.4.4.3.4 Nine-bit format with parity enabled

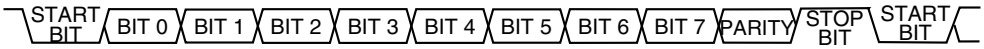



**Figure 51-237. Eight bits of data with LSB first and parity**

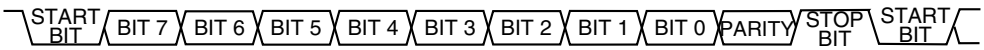



**Figure 51-238. Eight bits of data with MSB first and parity**

#### 51.4.4.3.5 Non-memory mapped tenth bit for parity

The most significant memory-mapped bit can be used for address mark wakeup.


**Figure 51-239. Nine bits of data with LSB first and parity**

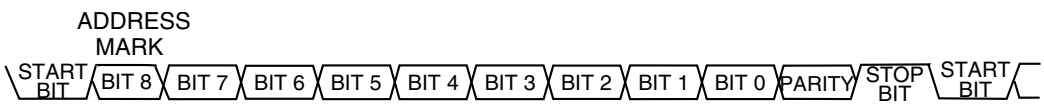



**Figure 51-240. Nine bits of data with MSB first and parity**

### 51.4.5 Single-wire operation

Normally, the UART uses two pins for transmitting and receiving. In single wire operation, the RXD pin is disconnected from the UART and the UART implements a half-duplex serial connection. The UART uses the TXD pin for both receiving and transmitting.

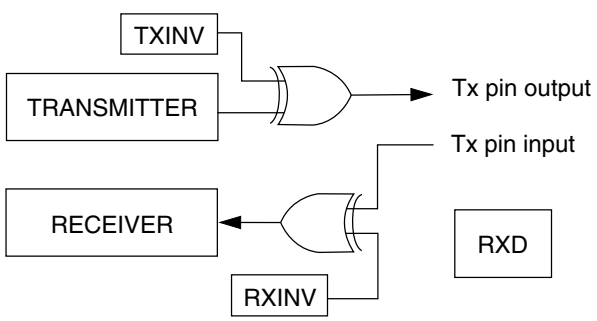



**Figure 51-241. Single-wire operation (C1[LOOPS] = 1, C1[RSRC] = 1)**

Enable single wire operation by setting C1[LOOPS] and the receiver source field, C1[RSRC]. Setting C1[LOOPS] disables the path from the unsynchronized receiver input signal to the receiver. Setting C1[RSRC] connects the receiver input to the output of the

TXD pin driver. Both the transmitter and receiver must be enabled (C2[TE] = 1 and C2[RE] = 1). When C7816[ISO_7816EN] is set, it is not required that both C2[TE] and C2[RE] are set.

## 51.4.6 Loop operation

In loop operation, the transmitter output goes to the receiver input. The unsynchronized receiver input signal is disconnected from the UART.

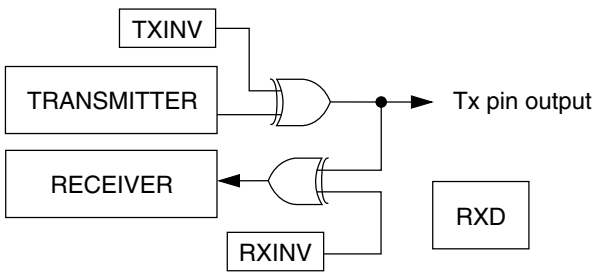



**Figure 51-242. Loop operation (C1[LOOPS] = 1, C1[RSRC] = 0)**

Enable loop operation by setting C1[LOOPS] and clearing C1[RSRC]. Setting C1[LOOPS] disables the path from the unsynchronized receiver input signal to the receiver. Clearing C1[RSRC] connects the transmitter output to the receiver input. Both the transmitter and receiver must be enabled (C2[TE] = 1 and C2[RE] = 1). When C7816[ISO_7816EN] is set, it is not required that both C2[TE] and C2[RE] are set.

## 51.4.7 ISO-7816/smartcard support

The UART provides mechanisms to support the ISO-7816 protocol that is commonly used to interface with smartcards. The ISO-7816 protocol is an NRZ, single wire, half-duplex interface. The TxD pin is used in open-drain mode because the data signal is used for both transmitting and receiving. There are multiple subprotocols within the ISO-7816 standard. The UART supports both T = 0 and T = 1 protocols. The module also provides for automated initial character detection and configuration, which allows for support of both direct convention and inverse convention data formats. A variety of interrupts specific to 7816 are provided in addition to the general interrupts to assist software. Additionally, the module is able to provide automated NACK responses and has programmed automated retransmission of failed packets. An assortment of programmable timeouts and guard band times are also supported.

The term elemental time unit (ETU) is frequently used in the context of ISO-7816. This concept is used to relate the frequency that the system (UART) is running at and the frequency that data is being transmitted and received. One ETU represents the time it

takes to transmit or receive a single bit. For example, a standard 7816 packet, excluding any guard time or NACK elements is 10 ETUs (start bit, 8 data bits, and a parity bit). Guard times and wait times are also measured in ETUs.,

**NOTE**

The ISO-7816 specification may have certain configuration options that are reserved. To maintain maximum flexibility to support future 7816 enhancements or devices that may not strictly conform to the specification, the UART does not prevent those options being used. Further, the UART may provide configuration options that exceed the flexibility of options explicitly allowed by the 7816 specification. Failure to correctly configure the UART may result in unexpected behavior or incompatibility with the ISO-7816 specification.

### 51.4.7.1 Initial characters

In ISO-7816 with T = 0 mode, the UART can be configured to use C7816[INIT] to detect the next valid initial character, referred to by the ISO-7816 specifically as a TS character. When the initial character is detected, the UART provides the host processor with an interrupt if IE7816[INITDE] is set. Additionally, the UART will alter S2[MSBF], C3[TXINV], and S2[RXINV] automatically, based on the initial character. The corresponding initial character and resulting register settings are listed in the following table.

**Table 51-236. Initial character automated settings**

| Initial character (bit 1-10) | Initial character (hex) | MSBF | TXINV | RXINV |
|---|---|---|---|---|
| LHHL LLL LLH<br>inverse convention | 3F | 1 | 1 | 1 |
| LHHL HHH LLH<br>direct convention | 3B | 0 | 0 | 0 |

S2[MSBF], C3[TXINV], and S2[RXINV] must be reset to their default values before C7816[INIT] is set. Once C7816[INIT] is set, the receiver searches all received data for the first valid initial character. Detecting a Direct Convention Initial Character will cause no change to S2[MSBF], C3[TXINV], and S2[RXINV], while detecting an Inverse Convention Initial Character will cause these fields to set automatically. All data received, which is not a valid initial character, is ignored and all flags resulting from the invalid data are blocked from asserting. If C7816[ANACK] is set, a NACK is returned for invalid received initial characters and an RXT interrupt is generated as programmed.

### 51.4.7.2 Protocol T = 0

When T = 0 protocol is selected, a relatively complex error detection scheme is used. Data characters are formatted as illustrated in the following figure. This scheme is also used for answer to reset and Peripheral Pin Select (PPS) formats.

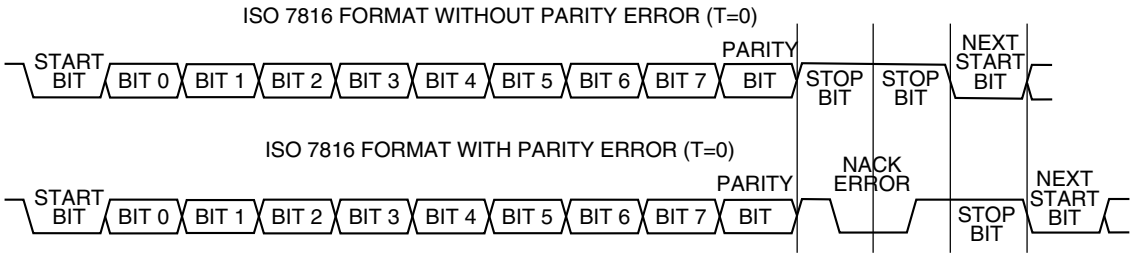



**Figure 51-243. ISO-7816 T = 0 data format**

As with other protocols supported by the UART, the data character includes a start bit. However, in this case, there are two stop bits rather than the typical single stop bit. In addition to a standard even parity check, the receiver has the ability to generate and return a NACK during the second half of the first stop bit period. The NACK must be at least one time period (ETU) in length and no more than two time periods (ETU) in length. The transmitter must wait for at least two time units (ETU) after detection of the error signal before attempting to retransmit the character.

It is assumed that the UART and the device (smartcard) know in advance which device is receiving and which is transmitting. No special mechanism is supplied by the UART to control receive and transmit in the mode other than C2[TE] and C2[RE]. Initial Character Detect feature is also supported in this mode.

### 51.4.7.3 Protocol T = 1

When T = 1 protocol is selected, the NACK error detection scheme is not used. Rather, the parity bit is used on a character basis and a CRC or LRC is used on the block basis, that is, for each group of characters. In this mode, the data format allows for a single stop bit although additional inactive bit periods may be present between the stop bit and the next start bit. Data characters are formatted as illustrated in the following figure.

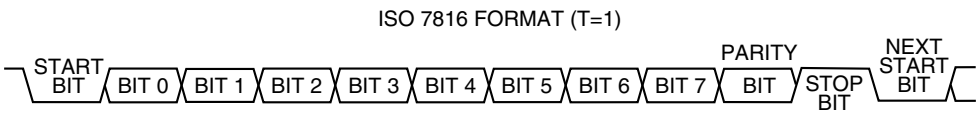



**Figure 51-244. ISO 7816 T=1 data format**

The smallest data unit that is transferred is a block. A block is made up of several data characters and may vary in size depending on the block type. The UART does not provide a mechanism to decode the block type. As part of the block, an LRC or CRC is included. The UART does not calculate the CRC or LRC for transmitted blocks, nor does it verify the validity of the CRC or LRC for received blocks. The 7816 protocol requires that the initiator and the smartcard (device) takes alternate turns in transmitting and receiving blocks. When the UART detects that the last character in a block has been transmitted it will automatically clear C2[TE], C3[TXDIR] and enter receive mode. Therefore, the software must program the transmit buffer with the next data to be transmitted and then enable C2[TE] and set C3[TXDIR], once the software has determined that the last character of the received block has been received. The UART detects that the last character of the transmit block has been sent when TL7816[TLEN] = 0 and four additional characters have been sent. The four additional characters are made up of three prior to TL7816[TLEN] decrementing (prologue) and one after TL7816[TLEN] = 0, the final character of the epilogue.

#### 51.4.7.4 Wait time and guard time parameters

The ISO-7816 specification defines several wait time and guard time parameters. The UART allows for flexible configuration and violation detection of these settings. On reset, the wait time (IS7816[WT]) defaults to 9600 ETUs and guard time (GT) to 12 ETUs. These values are controlled by parameters in the WP7816, WN7816, and WF7816 registers. Additionally, the value of C7816[TTYPE] also factors into the calculation. The formulae used to calculate the number ETUs for each wait time and guard time value are shown in Table 51-237.

Wait time (WT) is defined as the maximum allowable time between the leading edge of a character transmitted by the smartcard device and the leading edge of the previous character that was transmitted by the UART or the device. Similarly, character wait time (CWT) is defined as the maximum allowable time between the leading edge of two characters within the same block. Block wait time (BWT) is defined as the maximum time between the leading edge character of the last block received by the smartcard device and the leading edge of the first character transmitted by the smartcard device.

Guard time (GT) is defined as the minimum allowable time between the leading edge of two consecutive characters. Character guard time (CGT) is the minimum allowable time between the leading edges of two consecutive characters in the same direction, that is, transmission or reception. Block guard time (BGT) is the minimum allowable time between the leading edges of two consecutive characters in opposite directions, that is, transmission then reception or reception then transmission.

The GT and WT counters reset whenever C7816[TTYPE] = 1 or C7816[ISO_7816E] = 0 or a new dataword start bit has been received or transmitted as specified by the counter descriptions. The CWT, CGT, BWT, BGT counters reset whenever C7816[TTYPE] = 0 or C7816[ISO_7816E] = 0 or a new dataword start bit is received or transmitted as specified by the counter descriptions. When C7816[TTYPE] = 1, some of the counter values require an assumption regarding the first data transferred when the UART first starts. This assumption is required when the 7816E is disabled, when transition from C7816[TTYPE] = 0 to C7816[TTYPE] = 1 or when coming out of reset. In this case, it is assumed that the previous non-existent transfer was a received transfer.

The UART will automatically handle GT, CGT, and BGT such that the UART will not send a packet before the corresponding guard time expiring.

**Table 51-237. Wait and guard time calculations**

| Parameter | Reset value [ETU] | C7816[TTYPE] = 0 [ETU] | C7816[TTYPE] = 1 [ETU] |
|---|---|---|---|
| Wait time (WT) | 9600 | ((WI + 1) × 960 × (GTFD + 1)) - 1 | Not used |
| Character wait time (CWT) | Not used | Not used | $11 + 2^{(CWI - 1)}$ |
| Block wait time (BWT) | Not used | Not used | $10 + 2^{BWI} \times 960 \times (GTFD + 1)$ |
| Guard time (GT) | 12 | **GTN not equal to 255**<br>12 + GTN<br>**GTN equal to 255**<br>12 | Not used |
| Character guard time (CGT) | Not used | Not used | **GTN not equal to 255**<br>12 + GTN<br>**GTN equal to 255**<br>11 |
| Block guard time (BGT) | Not used | Not used | 22 |

### 51.4.7.5 Baud rate generation

The value in WF7816[GTFD] does not impact the clock frequency. SBR and BRFD are used to generate the clock frequency. This clock frequency is used by the UART only and is not seen by the smartcard device. The transmitter clocks operates at 1/16 the frequency of the receive clock so that the receiver is able to sample the received value 16 times during the ETU.

#### 51.4.7.6 UART restrictions in ISO-7816 operation

Due to the flexibility of the UART module, there are several features and interrupts that are not supported while running in ISO-7816 mode. These restrictions are documented within the register field definitions.

### 51.4.8 Infrared interface

The UART provides the capability of transmitting narrow pulses to an IR LED and receiving narrow pulses and transforming them to serial bits, which are sent to the UART. The IrDA physical layer specification defines a half-duplex infrared communication link for exchanging data. The full standard includes data rates up to 16 Mbits/s. This design covers data rates only between 2.4 kbits/s and 115.2 kbits/s.

The UART has an infrared transmit encoder and receive decoder. The UART transmits serial bits of data that are encoded by the infrared submodule to transmit a narrow pulse for every zero bit. No pulse is transmitted for every one bit. When receiving data, the IR pulses are detected using an IR photo diode and transformed to CMOS levels by the IR receive decoder, external from the MCU. The narrow pulses are then stretched by the infrared receive decoder to get back to a serial bit stream to be received by the UART. The polarity of transmitted pulses and expected receive pulses can be inverted so that a direct connection can be made to external IrDA transceiver modules that use active low pulses.

The infrared submodule receives its clock sources from the UART. One of these two clocks are selected in the infrared submodule to generate either 3/16, 1/16, 1/32, or 1/4 narrow pulses during transmission.

#### 51.4.8.1 Infrared transmit encoder

The infrared transmit encoder converts serial bits of data from transmit shift register to the TXD signal. A narrow pulse is transmitted for a zero bit and no pulse for a one bit. The narrow pulse is sent in the middle of the bit with a duration of 1/32, 1/16, 3/16, or 1/4 of a bit time. A narrow high pulse is transmitted for a zero bit when C3[TXINV] is cleared, while a narrow low pulse is transmitted for a zero bit when C3[TXINV] is set.

### 51.4.8.2 Infrared receive decoder

The infrared receive block converts data from the RXD signal to the receive shift register. A narrow pulse is expected for each zero received and no pulse is expected for each one received. A narrow high pulse is expected for a zero bit when S2[RXINV] is cleared, while a narrow low pulse is expected for a zero bit when S2[RXINV] is set. This receive decoder meets the edge jitter requirement as defined by the IrDA serial infrared physical layer specification.

## 51.5 Reset

All registers reset to a particular value are indicated in Memory map and registers.

## 51.6 System level interrupt sources

There are several interrupt signals that are sent from the UART. The following table lists the interrupt sources generated by the UART. The local enables for the UART interrupt sources are described in this table. Details regarding the individual operation of each interrupt are contained under various sub-sections of Memory map and registers. However, RXEDGIF description also outlines additional details regarding the RXEDGIF interrupt because of its complexity of operation. Any of the UART interrupt requests listed in the table can be used to bring the CPU out of Wait mode.

**Table 51-238. UART interrupt sources**

| Interrupt Source | Flag | Local enable | DMA select |
|---|---|---|---|
| Transmitter | TDRE | TIE | TDMAS = 0 |
| Transmitter | TC | TCIE | - |
| Receiver | IDLE | ILIE | - |
| Receiver | RDRF | RIE | RDMAS = 0 |
| Receiver | LBKDIF | LBKDIE | - |
| Receiver | RXEDGIF | RXEDGIE | - |
| Receiver | OR | ORIE | - |
| Receiver | NF | NEIE | - |
| Receiver | FE | FEIE | - |
| Receiver | PF | PEIE | - |
| Receiver | RXUF | RXUFE | - |
| Transmitter | TXOF | TXOFE | - |
| Receiver | WT | WTWE | - |
| Receiver | CWT | CWTE | - |

*Table continues on the next page...*

Table 51-238. UART interrupt sources (continued)

| Interrupt Source | Flag | Local enable | DMA select |
|---|---|---|---|
| Receiver | BWT | BWTE | - |
| Receiver | INITD | INITDE | - |
| Receiver | TXT | TXTE | - |
| Receiver | RXT | RXTE | - |
| Receiver | GTV | GTVE | - |

## 51.6.1 RXEDGIF description

S2[RXEDGIF] is set when an active edge is detected on the RxD pin. Therefore, the active edge can be detected only when in two wire mode. A RXEDGIF interrupt is generated only when S2[RXEDGIF] is set. If RXEDGIE is not enabled before S2[RXEDGIF] is set, an interrupt is not generated until S2[RXEDGIF] is set.

### 51.6.1.1 RxD edge detect sensitivity

Edge sensitivity can be software programmed to be either falling or rising. The polarity of the edge sensitivity is selected using S2[RXINV]. To detect the falling edge, S2[RXINV] is programmed to 0. To detect the rising edge, S2[RXINV] is programmed to 1.

Synchronizing logic is used prior to detect edges. Prior to detecting an edge, the receive data on RxD input must be at the deasserted logic level. A falling edge is detected when the RxD input signal is seen as a logic 1 (the deasserted level) during one module clock cycle, and then a logic 0 (the asserted level) during the next cycle. A rising edge is detected when the input is seen as a logic 0 during one module clock cycle and then a logic 1 during the next cycle.

### 51.6.1.2 Clearing RXEDGIF interrupt request

Writing a logic 1 to S2[RXEDGIF] immediately clears the RXEDGIF interrupt request even if the RxD input remains asserted. S2[RXEDGIF] remains set if another active edge is detected on RxD while attempting to clear S2[RXEDGIF] by writing a 1 to it.

#### 51.6.1.3 Exit from low-power modes

The receive input active edge detect circuit is still active on low power modes (Wait and Stop). An active edge on the receive input brings the CPU out of low power mode if the interrupt is not masked (S2[RXEDGIF] = 1).

## 51.7 DMA operation

In the transmitter, S1[TDRE] can be configured to assert a DMA transfer request. In the receiver, S1[RDRF], can be configured to assert a DMA transfer request. The following table shows the configuration field settings required to configure each flag for DMA operation.

**Table 51-239. DMA configuration**

| Flag | Request enable bit | DMA select bit |
|---|---|---|
| TDRE | TIE = 1 | TDMAS = 1 |
| RDRF | RIE = 1 | RDMAS = 1 |

When a flag is configured for a DMA request, its associated DMA request is asserted when the flag is set. When S1[RDRF] is configured as a DMA request, the clearing mechanism of reading S1, followed by reading D, does not clear the associated flag. The DMA request remains asserted until an indication is received that the DMA transactions are done. When this indication is received, the flag bit and the associated DMA request is cleared. If the DMA operation failed to remove the situation that caused the DMA request, another request is issued.

## 51.8 Application information

This section describes the UART application information.

### 51.8.1 Transmit/receive data buffer operation

The UART has independent receive and transmit buffers. The size of these buffers may vary depending on the implementation of the module. The implemented size of the buffers is a fixed constant via PFIFO[TXFIFOSIZE] and PFIFO[RXFIFOSIZE]. Additionally, legacy support is provided that allows for the FIFO structure to operate as a

depth of one. This is the default/reset behavior of the module and can be adjusted using the PFIFO[RXFE] and PFIFO[TXFE] bits. Individual watermark levels are also provided for transmit and receive.

There are multiple ways to ensure that a data block, which is a set of characters, has completed transmission. These methods include:

1. Set TXFIFO[TXWATER] to 0. TDRE asserts when there is no further data in the transmit buffer. Alternatively the S1[TC] flag can be used to indicate when the transmit shift register is also empty.
2. Poll TCFIFO[TXCOUNT]. Assuming that only data for a data block has been put into the data buffer, when TCFIFO[TXCOUNT] = 0, all data has been transmitted or is in the process of transmission.
3. S1[TC] can be monitored. When S1[TC] asserts, it indicates that all data has been transmitted and there is no data currently being transmitted in the shift register.

### 51.8.2 ISO-7816 initialization sequence

This section outlines how to program the UART for ISO-7816 operation. Elements such as procedures to power up or power down the smartcard, and when to take those actions, are beyond the scope of this description. To set up the UART for ISO-7816 operation:

1. Select a baud rate. Write this value to the UART baud registers (BDH/L) to begin the baud rate generator. Remember that the baud rate generator is disabled when the baud rate is zero. Writing to the BDH has no effect without also writing to BDL. According to the 7816 specification the initial (default) baud rating setting should be Fi = 372 and Di = 1 and a maximum frequency of 5 MHz. In other words, the BDH, BDL, and C4 registers should be programmed such that the transmission frequency provided to the smartcard device must be 1/372th of the clock and must not exceed 5 MHz.
2. Write to set BDH[LBKDIE] = 0.
3. Write to C1 to configure word length, parity, and other configuration fields (LOOPS, RSRC) and set C1[M] = 1, C1[PE] = 1, and C1[PT] = 0.
4. Write to set S2[RWUID] = 0 and S2[LBKDE] = 0.
5. Write to set MODEM[RXRTSE] = 0, MODEM[TXRTSPOL] = 0, MODEM[TXRTSE] = 0, and MODEM[TXCTSE] = 0.

6. Write to set up interrupt enable fields desired (C3[ORIE], C3[NEIE], C3[PEIE], and C3[FEIE])
7. Write to set C4[MAEN1] = 0 and C4[MAEN2] = 0.
8. Write to C5 register and configure DMA control register fields as desired for application.
9. Write to set C7816[INIT] = 1,C7816[ TTYPE] = 0, and C7816[ISO_7816E] = 1. Program C7816[ONACK] and C7816[ANACK] as desired.
10. Write to IE7816 to set interrupt enable parameters as desired.
11. Write to ET7816 and set as desired.
12. Write to set C2[ILIE] = 0, C2[RE] = 1, C2[TE] = 1, C2[RWU] = 0, and C2[SBK] = 0. Set up interrupt enables C2[TIE], C2[TCIE], and C2[RIE] as desired.

At this time, the UART will start listening for an initial character. After being identified, it will automatically adjust S2[MSBF], C3[TXINV], and S2[RXINV]. The software must then receive and process an answer to reset. Upon processing the answer to reset, the software must write to set C2[RE] = 0 and C2[TE] = 0. The software should then adjust 7816 specific and UART generic parameters to match and configure data that was received during the answer on reset period. After the new settings have been programmed, including the new baud rate and C7816[TTYPE], C2[RE] and C2[TE] can be reenabled as required.

#### 51.8.2.1 Transmission procedure for (C7816[TTYPE] = 0)

When the protocol selected is C7816[TTYPE] = 0, it is assumed that the software has a prior knowledge of who should be transmitting and receiving. Therefore, no mechanism is provided for automated transmission/receipt control. The software must monitor S1[TDRE], or configure for an interrupt, and provide additional data for transmission, as appropriate. Additionally, software should set C2[TE] = 1 and control TXDIR whenever it is the UART's turn to transmit information. For ease of monitoring, it is suggested that only data be transmitted until the next receiver/transmit switchover is loaded into the transmit FIFO/buffer.

### 51.8.2.2 Transmission procedure for (C7816[TTYPE] = 1)

When the protocol selected is C7816[TTYPE] = 1, data is transferred in blocks. Before starting a transmission, the software must write the size, in number of bytes, for the Information Field portion of the block into TLEN. If a CRC is being transmitted for the block, the value in TLEN must be one more than the size of the information field. The software must then set C2[TE] = 1 and C2[RE] = 1. The software must then monitor S1[TDRE]/interrupt and write the prologue, information, and epilogue field to the transmit buffer. TLEN automatically decrements, except for prologue bytes and the final epilogue byte. When the final epilogue byte has been transmitted, the UART automatically clears C2[TE] and C3[TXDIR] to 0, and the UART automatically starts capturing the response to the block that was transmitted. After the software has detected the receipt of the response, the transmission process must be repeated as needed with sufficient urgency to ensure that the block wait time and character wait times are not violated.

## 51.8.3 Initialization sequence (non ISO-7816)

To initiate a UART transmission:

1. Configure the UART.
    a. Select a baud rate. Write this value to the UART baud registers (BDH/L) to begin the baud rate generator. Remember that the baud rate generator is disabled when the baud rate is zero. Writing to the BDH has no effect without also writing to BDL.
    b. Write to C1 to configure word length, parity, and other configuration bits (LOOPS, RSRC, M, WAKE, ILT, PE, and PT). Write to C4, MA1, and MA2 to configure.
    c. Enable the transmitter, interrupts, receiver, and wakeup as required, by writing to C2 (TIE, TCIE, RIE, ILIE, TE, RE, RWU, and SBK), S2 (MSBF and BRK13), and C3 (ORIE, NEIE, PEIE, and FEIE). A preamble or idle character is then shifted out of the transmitter shift register.
2. Transmit procedure for each byte.
    a. Monitor S1[TDRE] by reading S1 or responding to the TDRE interrupt. The amount of free space in the transmit buffer directly using TCFIFO[TXCOUNT] can also be monitored.

b. If the TDRE flag is set, or there is space in the transmit buffer, write the data to be transmitted to (C3[T8]/D). A new transmission will not result until data exists in the transmit buffer.

3. Repeat step 2 for each subsequent transmission.

**Note**

During normal operation, S1[TDRE] is set when the shift register is loaded with the next data to be transmitted from the transmit buffer and the number of datawords contained in the transmit buffer is less than or equal to the value in TWFIFO[TXWATER]. This occurs 9/16ths of a bit time after the start of the stop bit of the previous frame.

To separate messages with preambles with minimum idle line time, use this sequence between messages.

1. Write the last dataword of the first message to C3[T8]/D.
2. Wait for S1[TDRE] to go high with TWFIFO[TXWATER] = 0, indicating the transfer of the last frame to the transmit shift register.
3. Queue a preamble by clearing and then setting C2[TE].
4. Write the first and subsequent datawords of the second message to C3[T8]/D.

## 51.8.4 Overrun (OR) flag implications

To be flexible, the overrun flag (OR) operates slight differently depending on the mode of operation. There may be implications that need to be carefully considered. This section clarifies the behavior and the resulting implications. Regardless of mode, if a dataword is received while S1[OR] is set, S1[RDRF] and S1[IDLE] are blocked from asserting. If S1[RDRF] or S1[IDLE] were previously asserted, they will remain asserted until cleared.

### 51.8.4.1 Overrun operation

The assertion of S1[OR] indicates that a significant event has occurred. The assertion indicates that received data has been lost because there was a lack of room to store it in the data buffer. Therefore, while S1[OR] is set, no further data is stored in the data buffer until S1[OR] is cleared. This ensures that the application will be able to handle the overrun condition.

In most applications, because the total amount of lost data is known, the application will attempt to return the system to a known state. Before S1[OR] is cleared, all received data will be dropped. For this, the software does the following.

1. Remove data from the receive data buffer. This could be done by reading data from the data buffer and processing it if the data in the FIFO was still valuable when the overrun event occurred, or using CFIFO[RXFLUSH] to clear the buffer.
2. Clear S1[OR]. Note that if data was cleared using CFIFO[RXFLUSH], then clearing S1[OR] will result in SFIFO[RXUF] asserting. This is because the only way to clear S1[OR] requires reading additional information from the FIFO. Care should be taken to disable the SFIFO[RXUF] interrupt prior to clearing the OR flag and then clearing SFIFO[RXUF] after the OR flag has been cleared.

Note that, in some applications, if an overrun event is responded to fast enough, the lost data can be recovered. For example, when C7816[ISO_7816E] is asserted, C7816[TTYPE]=1 and C7816[ONACK] = 1, the application may reasonably be able to determine whether the lost data will be resent by the device. In this scenario, flushing the receiver data buffer may not be required. Rather, if S1[OR] is cleared, the lost data may be resent and therefore may be recoverable.

When LIN break detect (LBKDE) is asserted, S1[OR] has significantly different behavior than in other modes. S1[OR] will be set, regardless of how much space is actually available in the data buffer, if a LIN break character has been detected and the corresponding flag, S2[LBKDIF], is not cleared before the first data character is received after S2[LBKDIF] asserted. This behavior is intended to allow the software sufficient time to read the LIN break character from the data buffer to ensure that a break character was actually detected. The checking of the break character was used on some older implementations and is therefore supported for legacy reasons. Applications that do not require this checking can simply clear S2[LBKDIF] without checking the stored value to ensure it is a break character.

### 51.8.5 Overrun NACK considerations

When C7816[ISO_7816E] is enabled and C7816[TTYPE] = 0, the retransmission feature of the 7816 protocol can be used to help avoid lost data when the data buffer overflows. Using C7816[ONACK], the module can be programmed to issue a NACK on an overflow event. Assuming that the smartcard device has implemented retransmission, the lost data will be retransmitted. While useful, there is a programming implication that may require special consideration. The need to transmit a NACK must be determined and committed to prior to the dataword being fully received. While the NACK is being received, it is possible that the application code will read the data buffer such that

sufficient room will be made to store the dataword that is being NACKed. Even if room has been made in the data buffer after the transmission of a NACK is completed, the received data will always be discarded as a result of an overflow and the ET7816[RXTHRESHOLD] value will be incremented by one. However, if sufficient space now exists to write the received data which was NACK'ed, S1[OR] will be blocked and kept from asserting.

## 51.8.6 Match address registers

The two match address registers allow a second match address function for a broadcast or general call address to the serial bus, as an example.

## 51.8.7 Modem feature

This section describes the modem features.

### 51.8.7.1 Ready-to-receive using RTS

To help to stop overrun of the receiver data buffer, the RTS signal can be used by the receiver to indicate to another UART that it is ready to receive data. The other UART can send the data when its CTS signal is asserted. This handshaking conforms to the TIA-232-E standard. A transceiver is necessary if the required voltage levels of the communication link do not match the voltage levels of the UART's RTS and CTS signals.

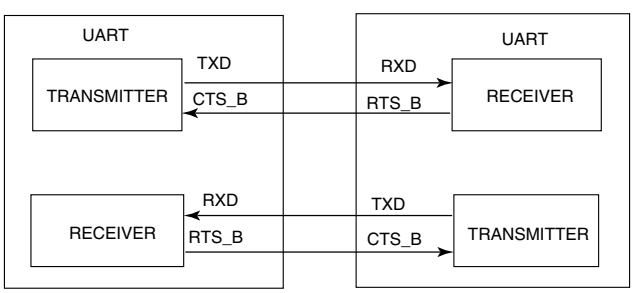



**Figure 51-245. Ready-to-receive**

The transmitter's CTS signal can be used for hardware flow control whether its RTS signal is used for hardware flow control, transceiver driver enable, or not at all.

### 51.8.7.2 Transceiver driver enable using RTS

RS-485 is a multiple drop communication protocol in which the UART transceiver's driver is 3-stated unless the UART is driving. The RTS signal can be used by the transmitter to enable the driver of a transceiver. The polarity of RTS can be matched to the polarity of the transceiver's driver enable signal. See the following figure.

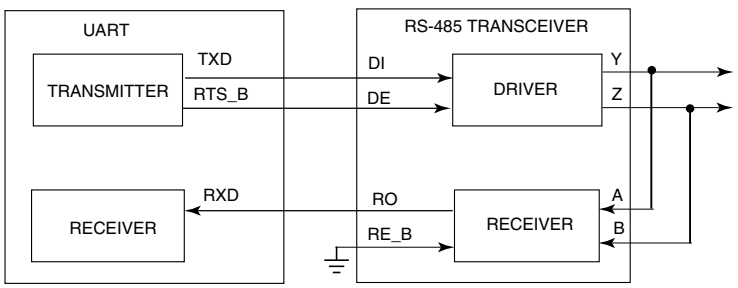



**Figure 51-246. Transceiver driver enable using RTS**

In the figure, the receiver enable signal is asserted. Another option for this connection is to connect RTS_B to both DE and RE_B. The transceiver's receiver is disabled while driving. A pullup can pull RXD to a non-floating value during this time. This option can be refined further by operating the UART in single wire mode, freeing the RXD pin for other uses.

## 51.8.8 IrDA minimum pulse width

The IrDA specifies a minimum pulse width of 1.6 µs. The UART hardware does not include a mechanism to restrict/force the pulse width to be greater than or equal to 1.6 µs. However, configuring the baud rate to 115.2 kbit/s and the narrow pulse width to 3/16 of a bit time results in a pulse width of 1.6 µs.

## 51.8.9 Clearing 7816 wait timer (WT, BWT, CWT) interrupts

The 7816 wait timer interrupts associated with IS7816[WT], IS7816[BWT], and IS7816[CWT] will automatically reassert if they are cleared and the wait time is still violated. This behavior is similar to most of the other interrupts on the UART. In most cases, if the condition that caused the interrupt to trigger still exists when the interrupt is cleared, then the interrupt will reassert. For example, consider the following scenario:

1. IS7816[WT] is programmed to assert after 9600 cycles of unresponsiveness.
2. The 9600 cycles pass without a response resulting in the WT interrupt asserting.
3. The IS7816[WT] is cleared at cycle 9700 by the interrupt service routine.

4. After the WT interrupt has been cleared, the smartcard remains unresponsive. At cycle 9701 the WT interrupt will be reasserted.

If the intent of clearing the interrupt is such that it does not reassert, the interrupt service routine must remove or clear the condition that originally caused the interrupt to assert prior to clearing the interrupt. There are multiple ways that this can be accomplished, including ensuring that an event that results in the wait timer resetting occurs, such as, the transmission of another packet.

### 51.8.10 Legacy and reverse compatibility considerations

Recent versions of the UART have added several new features. Whenever reasonably possible, reverse compatibility was maintained. However, in some cases this was either not feasible or the behavior was deemed as not intended. This section describes several differences to legacy operation that resulted from these recent enhancements. If application code from previous versions is used, it must be reviewed and modified to take the following items into account. Depending on the application code, additional items that are not listed here may also need to be considered.

1. Various reserved registers and register bits are used, such as, MSFB and M10.
2. This module now generates an error when invalid address spaces are used.
3. While documentation indicated otherwise, in some cases it was possible for S1[IDLE] to assert even if S1[OR] was set.
4. S1[OR] will be set only if the data buffer (FIFO) does not have sufficient room. Previously, the data buffer was always a fixed size of one and the S1[OR] flag would set so long as S1[RDRF] was set even if there was room in the data buffer. While the clearing mechanism has remained the same for S1[RDRF], keeping the OR flag assertion tied to the RDRF event rather than the data buffer being full would have greatly reduced the usefulness of the buffer when its size is larger than one.
5. Previously, when C2[RWU] was set (and WAKE = 0), the IDLE flag could reassert up to every bit period causing an interrupt and requiring the host processor to reassert C2[RWU]. This behavior has been modified. Now, when C2[RWU] is set (and WAKE = 0), at least one non-idle bit must be detected before an idle can be detected.

# Chapter 52
# Secured digital host controller (SDHC)

## 52.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The chapter is intended for a module driver software developer. It describes module-level operation and programming.

## 52.2 Overview

### 52.2.1 Supported types of cards

Different types of cards supported by the SDHC are described briefly as follows:

The multimedia card (MMC) is a universal low-cost data storage and communication media that is designed to cover a wide area of applications including mobile video and gaming. Old MMC cards are based on a 7-pin serial bus with a single data pin, while the new high-speed MMC communication is based on an advanced 11-pin serial bus designed to operate in the low-voltage range.

The secure digital card (SD) is an evolution of the old MMC technology. It is specifically designed to meet the security, capacity, performance, and environment requirements inherent in newly emerging audio and video consumer electronic devices. The physical form factor, pin assignment, and data transfer protocol are forward compatible with the old MMC with some additions.

Under the SD protocol, it can be categorized into memory card, I/O card and combo card, which has both memory and I/O functions. The memory card invokes a copyright protection mechanism that complies with the security of the SDMI standard. The I/O card, which is also known as SDIO card, provides high-speed data I/O with low power consumption for mobile electronic devices. For the sake of simplicity, the figure does not show cards with reduced size or mini cards.

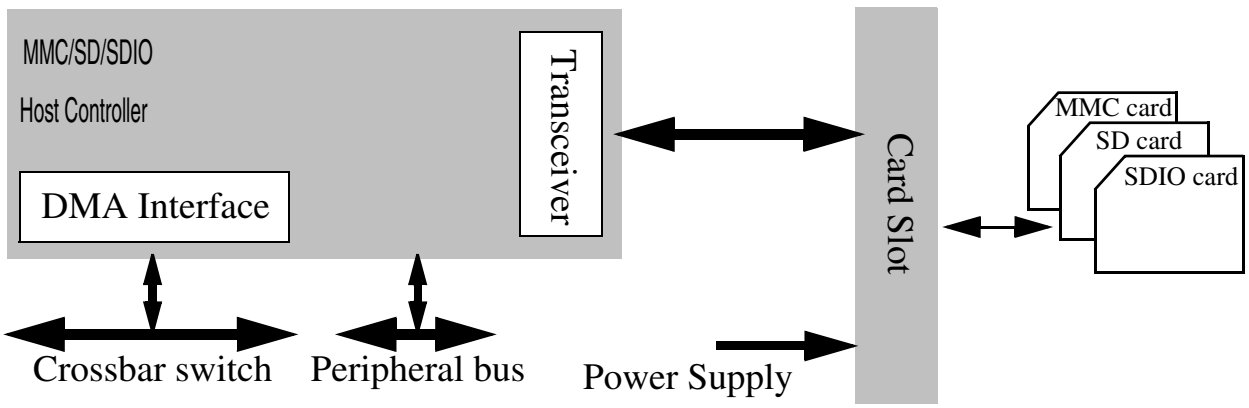



**Figure 52-1. System connection of the SDHC**

CE-ATA is a hard drive interface that is optimized for embedded applications storage. The device is layered on the top of the MMC protocol stack using the same physical interface. The interface electrical and signaling definition is defined like that in the MMC specification. See the CE-ATA specification for more details.

## 52.2.2 SDHC block diagram

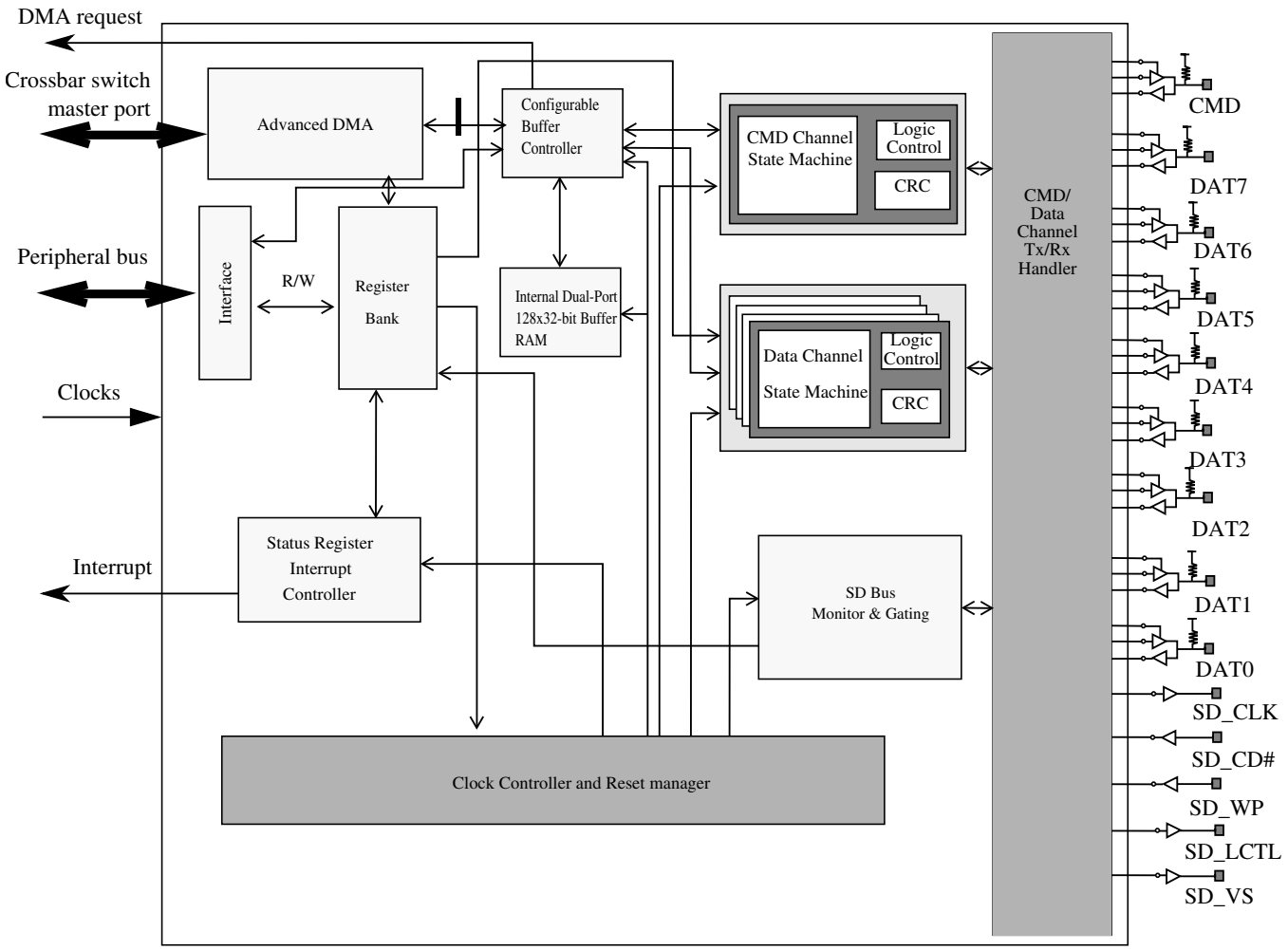



**Figure 52-2. Enhanced secure digital host controller block diagram**

## 52.2.3 Features

The features of the SDHC module include:

- Conforms to the SD Host Controller Standard Specification version 2.0 including test event register support
- Compatible with the MMC System Specification version 4.2/4.3
- Compatible with the SD Memory Card Specification version 2.0 and supports the high capacity SD memory card
- Compatible with the SDIO Card Specification version 2.0
- Compatible with the CE-ATA Card Specification version 1.0

- Designed to work with CE-ATA, SD memory, miniSD memory, SDIO, miniSDIO, SD Combo, MMC, MMC plus, and MMC RS cards
- Card bus clock frequency up to 52 MHz
- Supports 1-bit/4-bit SD and SDIO modes, 1-bit/4-bit / 8-bit MMC modes, 1-bit/4-bit/ 8-bit CE-ATA devices
  - Up to 200 Mbps of data transfer for SD/SDIO cards using 4 parallel data lines
  - Up to 416 Mbps of data transfer for MMC cards using 8 parallel data lines in Single Data Rate (SDR) mode
- Supports single block, multiblock read and write
- Supports block sizes of 1 ~ 4096 bytes
- Supports the write protection switch for write operations
- Supports both synchronous and asynchronous abort (both hardware and software CMD12)
- Supports pause during the data transfer at block gap
- Supports SDIO Read Wait and Suspend Resume operations
- Supports auto CMD12 for multiblock transfer
- Host can initiate non-data transfer command while data transfer is in progress
- Allows cards to interrupt the host in 1-bit and 4-bit SDIO modes, also supports interrupt period
- Embodies a fully configurable 128x32-bit FIFO for read/write data
- Supports internal and external DMA capabilities
- Supports advanced DMA to perform linked memory access

### 52.2.4 Modes and operations

The SDHC can select the following modes for data transfer:

- SD 1-bit
- SD 4-bit
- MMC 1-bit

- MMC 4-bit
- MMC 8-bit
- CE-ATA 1-bit
- CE-ATA 4-bit
- CE-ATA 8-bit
- Identification mode up to 400 kHz
- MMC Full Speed mode up to 20 MHz
- MMC High Speed mode up to 52 MHz
- SD/SDIO Full Speed mode up to 25 MHz
- SD/SDIO High Speed mode up to 50 MHz

## 52.3 SDHC signal descriptions

**Table 52-1. SDHC signal descriptions**

| Signal | Description | I/O |
|---|---|---|
| SDHC_DCLK | Generated clock used to drive the MMC, SD, SDIO or CE-ATA cards. | O |
| SDHC_CMD | Send commands to and receive responses from the card. | I/O |
| SDHC_D0 | DAT0 line or busy-state detect | I/O |
| SDHC_D1 | 8-bit mode: DAT1 line<br>4-bit mode: DAT1 line or interrupt detect<br>1-bit mode: Interrupt detect | I/O |
| SDHC_D2 | 4-/8-bit mode: DAT2 line or read wait<br>1-bit mode: Read wait | I/O |
| SDHC_D3 | 4-/8-bit mode: DAT3 line or configured as card detection pin<br>1-bit mode: May be configured as card detection pin | I/O |
| SDHC_D4 | DAT4 line in 8-bit mode<br>Not used in other modes | I/O |
| SDHC_D5 | DAT5 line in 8-bit mode<br>Not used in other modes | I/O |
| SDHC_D6 | DAT6 line in 8-bit mode<br>Not used in other modes | I/O |

*Table continues on the next page...*

**Table 52-1. SDHC signal descriptions (continued)**

| Signal | Description | I/O |
|---|---|---|
| SDHC_D7 | DAT7 line in 8-bit mode<br>Not used in other modes | I/O |

## 52.4 Memory map and register definition

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

**SDHC memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 400B_1000 | DMA System Address register (SDHC_DSADDR) | 32 | R/W | 0000_0000h | 52.4.1/1455 |
| 400B_1004 | Block Attributes register (SDHC_BLKATTR) | 32 | R/W | 0000_0000h | 52.4.2/1456 |
| 400B_1008 | Command Argument register (SDHC_CMDARG) | 32 | R/W | 0000_0000h | 52.4.3/1457 |
| 400B_100C | Transfer Type register (SDHC_XFERTYP) | 32 | R/W | 0000_0000h | 52.4.4/1457 |
| 400B_1010 | Command Response 0 (SDHC_CMDRSP0) | 32 | R | 0000_0000h | 52.4.5/1461 |
| 400B_1014 | Command Response 1 (SDHC_CMDRSP1) | 32 | R | 0000_0000h | 52.4.6/1462 |
| 400B_1018 | Command Response 2 (SDHC_CMDRSP2) | 32 | R | 0000_0000h | 52.4.7/1462 |
| 400B_101C | Command Response 3 (SDHC_CMDRSP3) | 32 | R | 0000_0000h | 52.4.8/1462 |
| 400B_1020 | Buffer Data Port register (SDHC_DATPORT) | 32 | R/W | 0000_0000h | 52.4.9/1464 |
| 400B_1024 | Present State register (SDHC_PRSSTAT) | 32 | R | 0000_0000h | 52.4.10/ 1464 |
| 400B_1028 | Protocol Control register (SDHC_PROCTL) | 32 | R/W | 0000_0020h | 52.4.11/ 1469 |
| 400B_102C | System Control register (SDHC_SYSCTL) | 32 | R/W | 0000_8008h | 52.4.12/ 1473 |
| 400B_1030 | Interrupt Status register (SDHC_IRQSTAT) | 32 | R/W | 0000_0000h | 52.4.13/ 1476 |
| 400B_1034 | Interrupt Status Enable register (SDHC_IRQSTATEN) | 32 | R/W | 117F_013Fh | 52.4.14/ 1481 |
| 400B_1038 | Interrupt Signal Enable register (SDHC_IRQSIGEN) | 32 | R/W | 0000_0000h | 52.4.15/ 1484 |
| 400B_103C | Auto CMD12 Error Status Register (SDHC_AC12ERR) | 32 | R | 0000_0000h | 52.4.16/ 1486 |
| 400B_1040 | Host Controller Capabilities (SDHC_HTCAPBLT) | 32 | R | 07F3_0000h | 52.4.17/ 1490 |
| 400B_1044 | Watermark Level Register (SDHC_WML) | 32 | R/W | 0010_0010h | 52.4.18/ 1492 |
| 400B_1050 | Force Event register (SDHC_FEVT) | 32 | W (always reads 0) | 0000_0000h | 52.4.19/ 1493 |

*Table continues on the next page...*

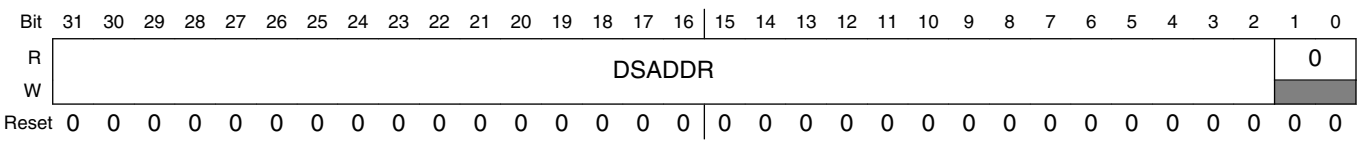
**SDHC memory map (continued)**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 400B_1054 | ADMA Error Status register (SDHC_ADMAES) | 32 | R | 0000_0000h | 52.4.20/ 1495 |
| 400B_1058 | ADMA System Addressregister (SDHC_ADSADDR) | 32 | R/W | 0000_0000h | 52.4.21/ 1498 |
| 400B_10C0 | Vendor Specific register (SDHC_VENDOR) | 32 | R/W | 0000_0001h | 52.4.22/ 1499 |
| 400B_10C4 | MMC Boot register (SDHC_MMCBOOT) | 32 | R/W | 0000_0000h | 52.4.23/ 1500 |
| 400B_10FC | Host Controller Version (SDHC_HOSTVER) | 32 | R | 0000_1201h | 52.4.24/ 1501 |

## 52.4.1 DMA System Address register (SDHC_DSADDR)

This register contains the physical system memory address used for DMA transfers.

Address: 400B_1000h base + 0h offset = 400B_1000h




**SDHC_DSADDR field descriptions**

| Field | Description |
|---|---|
| 31–2<br>DSADDR | DMA System Address<br><br>Contains the 32-bit system memory address for a DMA transfer. Because the address must be word (4 bytes) align, the least 2 bits are reserved, always 0. When the SDHC stops a DMA transfer, this register points to the system address of the next contiguous data position. It can be accessed only when no transaction is executing, that is, after a transaction has stopped. Read operation during transfers may return an invalid value. The host driver shall initialize this register before starting a DMA transaction. After DMA has stopped, the system address of the next contiguous data position can be read from this register. This register is protected during a data transfer. When data lines are active, write to this register is ignored. The host driver shall wait, until PRSSTAT[DLA] is cleared, before writing to this register.<br><br>The SDHC internal DMA does not support a virtual memory system. It supports only continuous physical memory access. And due to AHB burst limitations, if the burst must cross the 1 KB boundary, SDHC will automatically change SEQ burst type to NSEQ.<br><br>Because this register supports dynamic address reflecting, when IRQSTAT[TC] bit is set, it automatically alters the value of internal address counter, so SW cannot change this register when IRQSTAT[TC] is set. |
| 1–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 52.4.2 Block Attributes register (SDHC_BLKATTR)

This register is used to configure the number of data blocks and the number of bytes in each block.

Address: 400B_1000h base + 4h offset = 400B_1004h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BLKCNT | | | | | | | | | | | | | | | | 0 | | | BLKSIZE | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### SDHC_BLKATTR field descriptions

| Field | Description |
|---|---|
| 31–16<br>BLKCNT | Blocks Count For Current Transfer<br><br>This register is enabled when XFERTYP[BCEN] is set to 1 and is valid only for multiple block transfers. For single block transfer, this register will always read as 1. The host driver shall set this register to a value between 1 and the maximum block count. The SDHC decrements the block count after each block transfer and stops when the count reaches zero. Setting the block count to 0 results in no data blocks being transferred.<br><br>This register must be accessed only when no transaction is executing, that is, after transactions are stopped. During data transfer, read operations on this register may return an invalid value and write operations are ignored.<br><br>When saving transfer content as a result of a suspend command, the number of blocks yet to be transferred can be determined by reading this register. The reading of this register must be applied after transfer is paused by stop at block gap operation and before sending the command marked as suspend. This is because when suspend command is sent out, SDHC will regard the current transfer as aborted and change BLKCNT back to its original value instead of keeping the dynamical indicator of remained block count.<br><br>When restoring transfer content prior to issuing a resume command, the host driver shall restore the previously saved block count.<br><br>**NOTE:** Although the BLKCNT field is 0 after reset, the read of reset value is 0x1. This is because when XFERTYP[MSBSEL] is 0, indicating a single block transfer, the read value of BLKCNT is always 1.<br><br>0000h Stop count.<br>0001h 1 block<br>0002h 2 blocks<br>...<br>FFFFh 65535 blocks |
| 15–13<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 12–0<br>BLKSIZE | Transfer Block Size<br><br>Specifies the block size for block data transfers. Values ranging from 1 byte up to the maximum buffer size can be set. It can be accessed only when no transaction is executing, that is, after a transaction has stopped. Read operations during transfers may return an invalid value, and write operations will be ignored. |

*Table continues on the next page...*

**SDHC_BLKATTR field descriptions (continued)**

| Field | Description |
|---|---|
| | 000h No data transfer.<br>001h 1 Byte<br>002h 2 Bytes<br>003h 3 Bytes<br>004h 4 Bytes<br>...<br>1FFh 511 Bytes<br>200h 512 Bytes<br>...<br>800h 2048 Bytes<br>...<br>1000h 4096 Bytes |

## 52.4.3 Command Argument register (SDHC_CMDARG)

This register contains the SD/MMC command argument.

Address: 400B_1000h base + 8h offset = 400B_1008h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CMDARG | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SDHC_CMDARG field descriptions**

| Field | Description |
|---|---|
| 31–0<br>CMDARG | Command Argument<br><br>The SD/MMC command argument is specified as bits 39-8 of the command format in the SD or MMC specification. This register is write protected when PRSSTAT[CDIHB0] is set. |

## 52.4.4 Transfer Type register (SDHC_XFERTYP)

This register is used to control the operation of data transfers. The host driver shall set this register before issuing a command followed by a data transfer, or before issuing a resume command. To prevent data loss, the SDHC prevents writing to the bits that are involved in the data transfer of this register, when data transfer is active. These bits are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN, and DMAEN.

The host driver shall check PRSSTAT[CDIHB] and PRSSTAT[CIHB] before writing to this register. When PRSSTAT[CDIHB] is set, any attempt to send a command with data by writing to this register is ignored; when PRSSTAT[CIHB] bit is set, any write to this register is ignored.

On sending commands with data transfer involved, it is mandatory that the block size is nonzero. Besides, block count must also be nonzero, or indicated as single block transfer (bit 5 of this register is 0 when written), or block count is disabled (bit 1 of this register is 0 when written), otherwise SDHC will ignore the sending of this command and do nothing. For write command, with all above restrictions, it is also mandatory that the write protect switch is not active (WPSPL bit of Present State Register is 1), otherwise SDHC will also ignore the command.

If the commands with data transfer does not receive the response in 64 clock cycles, that is, response time-out, SDHC will regard the external device does not accept the command and abort the data transfer. In this scenario, the driver must issue the command again to retry the transfer. It is also possible that, for some reason, the card responds to the command but SDHC does not receive the response, and if it is internal DMA (either simple DMA or ADMA) read operation, the external system memory is over-written by the internal DMA with data sent back from the card.

The following table shows the summary of how register settings determine the type of data transfer.

**Table 52-7. Transfer Type register setting for various transfer types**

| Multi/Single block select | Block count enable | Block count | Function |
|---|---|---|---|
| 0 | Don't care | Don't care | Single transfer |
| 1 | 0 | Don't care | Infinite transfer |
| 1 | 1 | Positive number | Multiple transfer |
| 1 | 1 | Zero | No data transfer |

The following table shows the relationship between XFERTYP[CICEN] and XFERTYP[CCCEN], in regards to XFERTYP[RSPTYP] as well as the name of the response type.

**Table 52-8. Relationship between parameters and the name of the response type**

| Response type (RSPTYP) | Index check enable (CICEN) | CRC check enable (CCCEN) | Name of response type |
|---|---|---|---|
| 00 | 0 | 0 | No Response |
| 01 | 0 | 1 | IR2 |
| 10 | 0 | 0 | R3,R4 |
| 10 | 1 | 1 | R1,R5,R6 |
| 11 | 1 | 1 | R1b,R5b |

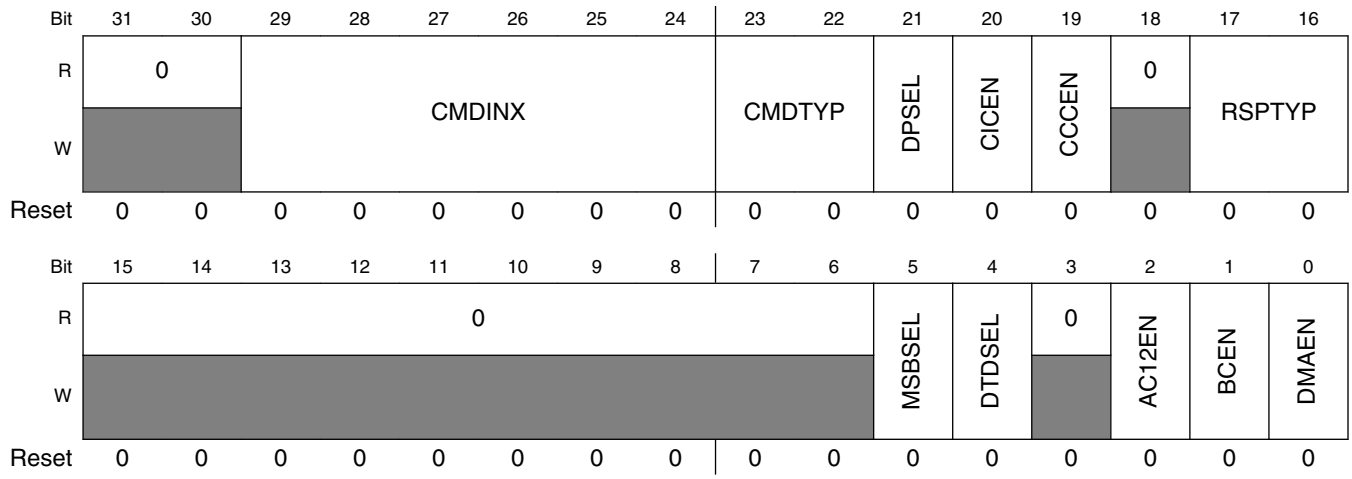
**NOTE**

- In the SDIO specification, response type notation for R5b is not defined. R5 includes R5b in the SDIO specification. But R5b is defined in this specification to specify that the

SDHC will check the busy status after receiving a response. For example, usually CMD52 is used with R5, but the I/O abort command shall be used with R5b.

- The CRC field for R3 and R4 is expected to be all 1 bits. The CRC check shall be disabled for these response types.

Address: 400B_1000h base + Ch offset = 400B_100Ch




## SDHC_XFERTYP field descriptions

| Field | Description |
|---|---|
| 31–30<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 29–24<br>CMDINX | Command Index<br><br>These bits shall be set to the command number that is specified in bits 45-40 of the command-format in the SD Memory Card Physical Layer Specification and SDIO Card Specification. |
| 23–22<br>CMDTYP | Command Type<br><br>There are three types of special commands: suspend, resume, and abort. These bits shall be set to 00b for all other commands.<br>• Suspend command: If the suspend command succeeds, the SDHC shall assume that the card bus has been released and that it is possible to issue the next command which uses the DAT line. Because the SDHC does not monitor the content of command response, it does not know if the suspend command succeeded or not. It is the host driver's responsibility to check the status of the suspend command and send another command marked as suspend to inform the SDHC that a suspend command was successfully issued. After the end bit of command is sent, the SDHC deasserts read wait for read transactions and stops checking busy for write transactions. In 4-bit mode, the interrupt cycle starts. If the suspend command fails, the SDHC will maintain its current state, and the host driver shall restart the transfer by setting PROCTL[CREQ].<br>• Resume command: The host driver restarts the data transfer by restoring the registers saved before sending the suspend command and then sends the resume command. The SDHC will check for a pending busy state before starting write transfers.<br>• Abort command: If this command is set when executing a read transfer, the SDHC will stop reads to the buffer. If this command is set when executing a write transfer, the SDHC will stop driving the DAT line. After issuing the abort command, the host driver must issue a software reset (abort transaction). |

*Table continues on the next page...*

## SDHC_XFERTYP field descriptions (continued)

| Field | Description |
|---|---|
| | 00b Normal other commands.<br>01b Suspend CMD52 for writing bus suspend in CCCR.<br>10b Resume CMD52 for writing function select in CCCR.<br>11b Abort CMD12, CMD52 for writing I/O abort in CCCR. |
| 21<br>DPSEL | Data Present Select<br><br>This bit is set to 1 to indicate that data is present and shall be transferred using the DAT line. It is set to 0 for the following:<br>• Commands using only the CMD line, for example: CMD52.<br>• Commands with no data transfer, but using the busy signal on DAT[0] line, R1b or R5b, for example: CMD38.<br><br>**NOTE:** In resume command, this bit shall be set, and other bits in this register shall be set the same as when the transfer was initially launched. When the Write Protect switch is on, that is, the WPSPL bit is active as 0, any command with a write operation will be ignored. That is to say, when this bit is set, while the DTDSEL bit is 0, writes to the register Transfer Type are ignored.<br><br>0b No data present.<br>1b Data present. |
| 20<br>CICEN | Command Index Check Enable<br><br>If this bit is set to 1, the SDHC will check the index field in the response to see if it has the same value as the command index. If it is not, it is reported as a command index error. If this bit is set to 0, the index field is not checked.<br><br>0b Disable<br>1b Enable |
| 19<br>CCCEN | Command CRC Check Enable<br><br>If this bit is set to 1, the SDHC shall check the CRC field in the response. If an error is detected, it is reported as a Command CRC Error. If this bit is set to 0, the CRC field is not checked. The number of bits checked by the CRC field value changes according to the length of the response.<br><br>0b Disable<br>1b Enable |
| 18<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 17–16<br>RSPTYP | Response Type Select<br><br>00b No response.<br>01b Response length 136.<br>10b Response length 48.<br>11b Response length 48, check busy after response. |
| 15–6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5<br>MSBSEL | Multi/Single Block Select<br><br>Enables multiple block DAT line data transfers. For any other commands, this bit shall be set to 0. If this bit is 0, it is not necessary to set the block count register.<br><br>0b Single block.<br>1b Multiple blocks. |

*Table continues on the next page...*

**SDHC_XFERTYP field descriptions (continued)**

| Field | Description |
|---|---|
| 4<br>DTDSEL | Data Transfer Direction Select<br><br>Defines the direction of DAT line data transfers. The bit is set to 1 by the host driver to transfer data from the SD card to the SDHC and is set to 0 for all other commands.<br><br>0b Write host to card.<br>1b Read card to host. |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2<br>AC12EN | Auto CMD12 Enable<br><br>Multiple block transfers for memory require a CMD12 to stop the transaction. When this bit is set to 1, the SDHC will issue a CMD12 automatically when the last block transfer has completed. The host driver shall not set this bit to issue commands that do not require CMD12 to stop a multiple block data transfer. In particular, secure commands defined in File Security Specification (see reference list) do not require CMD12. In single block transfer, the SDHC will ignore this bit whether it is set or not.<br><br>0b Disable<br>1b Enable |
| 1<br>BCEN | Block Count Enable<br><br>Used to enable the Block Count register, which is only relevant for multiple block transfers. When this bit is 0, the internal counter for block is disabled, which is useful in executing an infinite transfer.<br><br>0b Disable<br>1b Enable |
| 0<br>DMAEN | DMA Enable<br><br>Enables DMA functionality. If this bit is set to 1, a DMA operation shall begin when the host driver sets the DPSEL bit of this register. Whether the simple DMA, or the advanced DMA, is active depends on PROCTL[DMAS].<br><br>0b Disable<br>1b Enable |

## 52.4.5 Command Response 0 (SDHC_CMDRSP0)

This register is used to store part 0 of the response bits from the card.

Address: 400B_1000h base + 10h offset = 400B_1010h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CMDRSP0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SDHC_CMDRSP0 field descriptions**

| Field | Description |
|---|---|
| 31–0<br>CMDRSP0 | Command Response 0 |

## 52.4.6 Command Response 1 (SDHC_CMDRSP1)

This register is used to store part 1 of the response bits from the card.

Address: 400B_1000h base + 14h offset = 400B_1014h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CMDRSP1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SDHC_CMDRSP1 field descriptions**

| Field | Description |
|---|---|
| 31–0<br>CMDRSP1 | Command Response 1 |

## 52.4.7 Command Response 2 (SDHC_CMDRSP2)

This register is used to store part 2 of the response bits from the card.

Address: 400B_1000h base + 18h offset = 400B_1018h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CMDRSP2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SDHC_CMDRSP2 field descriptions**

| Field | Description |
|---|---|
| 31–0<br>CMDRSP2 | Command Response 2 |

## 52.4.8 Command Response 3 (SDHC_CMDRSP3)

This register is used to store part 3 of the response bits from the card.

The following table describes the mapping of command responses from the SD bus to command response registers for each response type. In the table, R[ ] refers to a bit range within the response data as transmitted on the SD bus.

**Table 52-13. Response bit definition for each response type**

| Response type | Meaning of response | Response field | Response register |
|---|---|---|---|
| R1,R1b (normal response) | Card status | R[39:8] | CMDRSP0 |
| R1b (Auto CMD12 response) | Card status for auto CMD12 | R[39:8] | CMDRSP3 |
| R2 (CID, CSD register) | CID/CSD register [127:8] | R[127:8] | {CMDRSP3[23:0], CMDRSP2, CMDRSP1, CMDRSP0} |
| R3 (OCR register) | OCR register for memory | R[39:8] | CMDRSP0 |
| R4 (OCR register) | OCR register for I/O etc. | R[39:8] | CMDRSP0 |
| R5, R5b | SDIO response | R[39:8] | CMDRSP0 |
| R6 (Publish RCA) | New published RCA[31:16] and card status[15:0] | R[39:9] | CMDRSP0 |

This table shows that most responses with a length of 48 (R[47:0]) have 32-bit of the response data (R[39:8]) stored in the CMDRSP0 register. Responses of type R1b (auto CMD12 responses) have response data bits (R[39:8]) stored in the CMDRSP3 register. Responses with length 136 (R[135:0]) have 120-bit of the response data (R[127:8]) stored in the CMDRSP0, 1, 2, and 3 registers.

To be able to read the response status efficiently, the SDHC stores only a part of the response data in the command response registers. This enables the host driver to efficiently read 32-bit of response data in one read cycle on a 32-bit bus system. Parts of the response, the index field and the CRC, are checked by the SDHC, as specified by XFERTYP[CICEN] and XFERTYP[CCCEN], and generate an error interrupt if any error is detected. The bit range for the CRC check depends on the response length. If the response length is 48, the SDHC will check R[47:1], and if the response length is 136 the SDHC will check R[119:1].

Because the SDHC may have a multiple block data transfer executing concurrently with a CMD_wo_DAT command, the SDHC stores the auto CMD12 response in the CMDRSP3 register. The CMD_wo_DAT response is stored in CMDRSP0. This allows the SDHC to avoid overwriting the Auto CMD12 response with the CMD_wo_DAT and vice versa. When the SDHC modifies part of the command response registers, as shown in the table above, it preserves the unmodified bits.

Address: 400B_1000h base + 1Ch offset = 400B_101Ch

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CMDRSP3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SDHC_CMDRSP3 field descriptions**

| Field | Description |
|---|---|
| 31–0<br>CMDRSP3 | Command Response 3 |

## 52.4.9 Buffer Data Port register (SDHC_DATPORT)

This is a 32-bit data port register used to access the internal buffer and it cannot be updated in Idle mode.

Address: 400B_1000h base + 20h offset = 400B_1020h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATCONT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SDHC_DATPORT field descriptions**

| Field | Description |
|---|---|
| 31–0<br>DATCONT | Data Content<br><br>The Buffer Data Port register is for 32-bit data access by the CPU or the external DMA. When the internal DMA is enabled, any write to this register is ignored, and any read from this register will always yield 0s. |

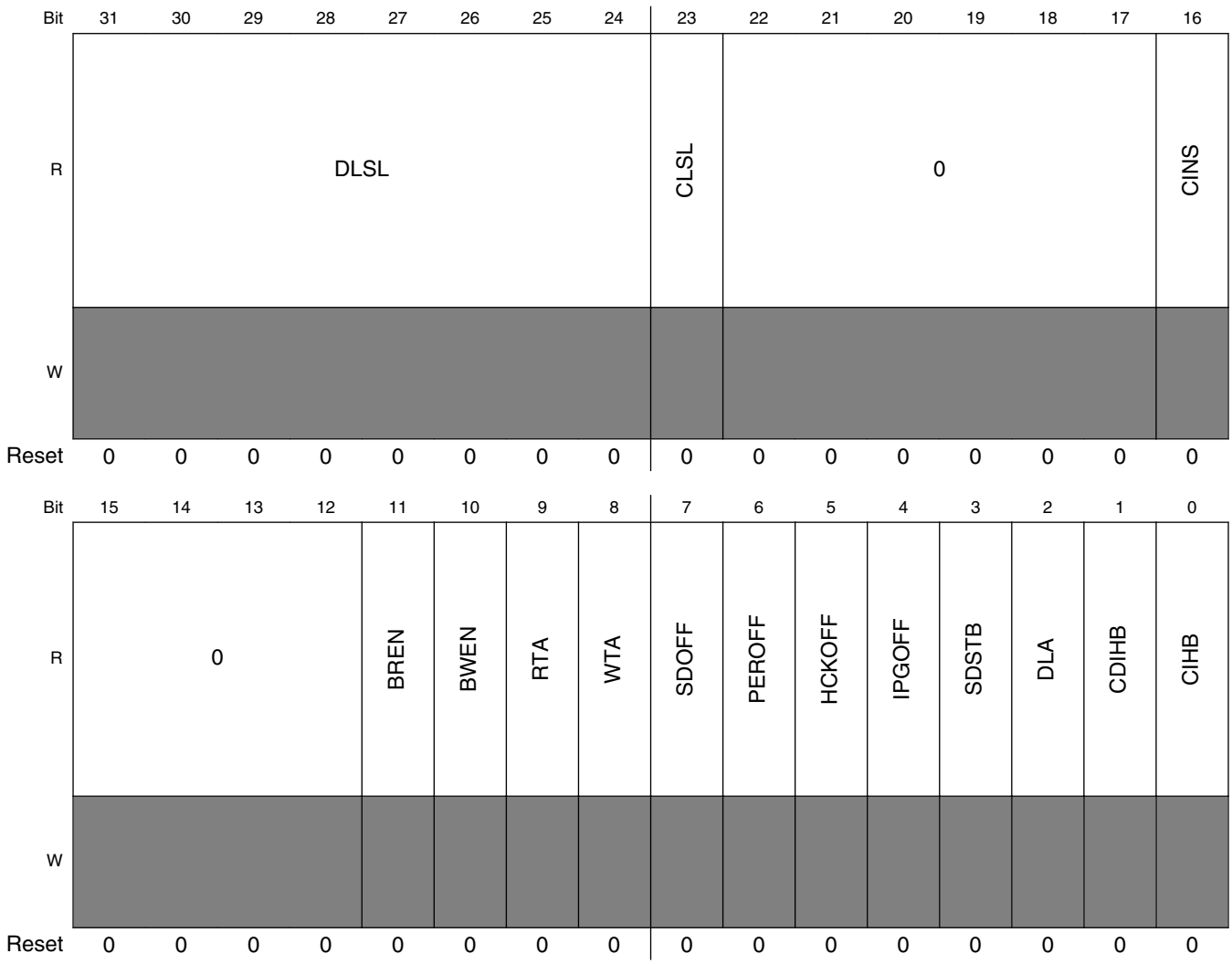
## 52.4.10 Present State register (SDHC_PRSSTAT)

The host driver can get status of the SDHC from this 32-bit read-only register.

**NOTE**

The host driver can issue CMD0, CMD12, CMD13 (for memory) and CMD52 (for SDIO) when the DAT lines are busy during a data transfer. These commands can be issued when Command Inhibit (CIHB) is set to zero. Other commands shall be issued when Command Inhibit (CDIHB) is set to zero. Possible changes to the SD Physical Specification may add other commands to this list in the future.

Address: 400B_1000h base + 24h offset = 400B_1024h




## SDHC_PRSSTAT field descriptions

| Field | Description |
|---|---|
| 31–24<br>DLSL | DAT Line Signal Level<br><br>Used to check the DAT line level to recover from errors, and for debugging. This is especially useful in detecting the busy signal level from DAT[0]. The reset value is effected by the external pullup/pulldown resistors. By default, the read value of this field after reset is 8'b11110111, when DAT[3] is pulled down and the other lines are pulled up.<br><br>DAT[0] Data 0 line signal level.<br>DAT[1] Data 1 line signal level.<br>DAT[2] Data 2 line signal level.<br>DAT[3] Data 3 line signal level.<br>DAT[4] Data 4 line signal level.<br>DAT[5] Data 5 line signal level.<br>DAT[6] Data 6 line signal level.<br>DAT[7] Data 7 line signal level. |

*Table continues on the next page...*

## SDHC_PRSSTAT field descriptions (continued)

| Field | Description |
|---|---|
| 23<br>CLSL | CMD Line Signal Level<br><br>Used to check the CMD line level to recover from errors, and for debugging. The reset value is effected by the external pullup/pulldown resistor, by default, the read value of this bit after reset is 1b, when the command line is pulled up. |
| 22–17<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 16<br>CINS | Card Inserted<br><br>Indicates whether a card has been inserted. The SDHC debounces this signal so that the host driver will not need to wait for it to stabilize. Changing from a 0 to 1 generates a card insertion interrupt in the Interrupt Status register. Changing from a 1 to 0 generates a card removal interrupt in the Interrupt Status register. A write to the force event register does not effect this bit.<br><br>SYSCTL[RSTA] does not effect this bit. A software reset does not effect this bit.<br><br>0b Power on reset or no card.<br>1b Card inserted. |
| 15–12<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 11<br>BREN | Buffer Read Enable<br><br>Used for non-DMA read transfers. The SDHC may implement multiple buffers to transfer data efficiently. This read-only flag indicates that valid data exists in the host side buffer. If this bit is high, valid data greater than the watermark level exist in the buffer. This read-only flag indicates that valid data exists in the host side buffer.<br><br>0b Read disable, valid data less than the watermark level exist in the buffer.<br>1b Read enable, valid data greater than the watermark level exist in the buffer. |
| 10<br>BWEN | Buffer Write Enable<br><br>Used for non-DMA write transfers. The SDHC can implement multiple buffers to transfer data efficiently. This read-only flag indicates whether space is available for write data. If this bit is 1, valid data greater than the watermark level can be written to the buffer. This read-only flag indicates whether space is available for write data.<br><br>0b Write disable, the buffer can hold valid data less than the write watermark level.<br>1b Write enable, the buffer can hold valid data greater than the write watermark level. |
| 9<br>RTA | Read Transfer Active<br>Used for detecting completion of a read transfer.<br>This bit is set for either of the following conditions:<br>• After the end bit of the read command.<br>• When writing a 1 to PROCTL[CREQ] to restart a read transfer.<br><br>A transfer complete interrupt is generated when this bit changes to 0. This bit is cleared for either of the following conditions:<br>• When the last data block as specified by block length is transferred to the system, that is, all data are read away from SDHC internal buffer.<br>• When all valid data blocks have been transferred from SDHC internal buffer to the system and no current block transfers are being sent as a result of the stop at block gap request being set to 1.<br><br>0b No valid data.<br>1b Transferring data. |

*Table continues on the next page...*

## SDHC_PRSSTAT field descriptions (continued)

| Field | Description |
|---|---|
| 8<br>WTA | Write Transfer Active<br>Indicates that a write transfer is active. If this bit is 0, it means no valid write data exists in the SDHC. This bit is set in either of the following cases:<br>• After the end bit of the write command.<br>• When writing 1 to PROCTL[CREQ] to restart a write transfer.<br><br>This bit is cleared in either of the following cases:<br>• After getting the CRC status of the last data block as specified by the transfer count (single and multiple).<br>• After getting the CRC status of any block where data transmission is about to be stopped by a stop at block gap request.<br><br>During a write transaction, a block gap event interrupt is generated when this bit is changed to 0, as result of the stop at block gap request being set. This status is useful for the host driver in determining when to issue commands during write busy state.<br><br>0b No valid data.<br>1b Transferring data. |
| 7<br>SDOFF | SD Clock Gated Off Internally<br><br>Indicates that the SD clock is internally gated off, because of buffer over/under-run or read pause without read wait assertion, or the driver has cleared SYSCTL[SDCLKEN] to stop the SD clock. This bit is for the host driver to debug data transaction on the SD bus.<br><br>0b SD clock is active.<br>1b SD clock is gated off. |
| 6<br>PEROFF | SDHC clock<br>Gated Off Internally<br><br>Indicates that the is internally gated off. This bit is for the host driver to debug transaction on the SD bus. When INITA bit is set, SDHC sending 80 clock cycles to the card, SDCLKEN must be 1 to enable the output card clock, otherwise the will never be gate off, so and will be always active. SDHC clockSDHC clockSDHC clockbus clock<br><br>0b SDHC clock is active.<br>1b SDHC clock is gated off. |
| 5<br>HCKOFF | System Clock<br>Gated Off Internally<br><br>Indicates that the system clock is internally gated off. This bit is for the host driver to debug during a data transfer.<br><br>0b System clock is active.<br>1b System clock is gated off. |
| 4<br>IPGOFF | Bus Clock |

*Table continues on the next page...*

## SDHC_PRSSTAT field descriptions (continued)

| Field | Description |
|---|---|
| | Gated Off Internally<br><br>Indicates that the bus clock is internally gated off. This bit is for the host driver to debug.<br><br>0b Bus clock is active.<br>1b Bus clock is gated off. |
| 3<br>SDSTB | SD Clock Stable<br><br>Indicates that the internal card clock is stable. This bit is for the host driver to poll clock status when changing the clock frequency. It is recommended to clear SYSCTL[SDCLKEN] to remove glitch on the card clock when the frequency is changing.<br><br>0b Clock is changing frequency and not stable.<br>1b Clock is stable. |
| 2<br>DLA | Data Line Active<br>Indicates whether one of the DAT lines on the SD bus is in use.<br><br>**In the case of read transactions:**<br><br>This status indicates whether a read transfer is executing on the SD bus. Changes in this value from 1 to 0, between data blocks, generates a block gap event interrupt in the Interrupt Status register.<br><br>This bit will be set in either of the following cases:<br>• After the end bit of the read command.<br>• When writing a 1 to PROCTL[CREQ] to restart a read transfer.<br><br>This bit will be cleared in either of the following cases:<br>1. When the end bit of the last data block is sent from the SD bus to the SDHC.<br>2. When the read wait state is stopped by a suspend command and the DAT2 line is released.<br><br>The SDHC will wait at the next block gap by driving read wait at the start of the interrupt cycle. If the read wait signal is already driven (data buffer cannot receive data), the SDHC can wait for a current block gap by continuing to drive the read wait signal. It is necessary to support read wait to use the suspend / resume function. This bit will remain 1 during read wait.<br><br>**In the case of write transactions:**<br><br>This status indicates that a write transfer is executing on the SD bus. Changes in this value from 1 to 0 generate a transfer complete interrupt in the interrupt status register.<br><br>This bit will be set in either of the following cases:<br>• After the end bit of the write command.<br>• When writing to 1 to PROCTL[CREQ] to continue a write transfer.<br><br>This bit will be cleared in either of the following cases:<br>• When the SD card releases write busy of the last data block, the SDHC will also detect if the output is not busy. If the SD card does not drive the busy signal after the CRC status is received, the SDHC shall assume the card drive "Not busy".<br>• When the SD card releases write busy, prior to waiting for write transfer, and as a result of a stop at block gap request.<br><br>**In the case of command with busy pending:**<br><br>This status indicates that a busy state follows the command and the data line is in use. This bit will be cleared when the DAT0 line is released.<br><br>0b DAT line inactive.<br>1b DAT line active. |

*Table continues on the next page...*

**SDHC_PRSSTAT field descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>CDIHB | Command Inhibit (DAT)<br><br>This status bit is generated if either the DLA or the RTA is set to 1. If this bit is 0, it indicates that the SDHC can issue the next SD/MMC Command. Commands with a busy signal belong to CDIHB, for example, R1b, R5b type. Except in the case when the command busy is finished, changing from 1 to 0 generates a transfer complete interrupt in the Interrupt Status register.<br><br>**NOTE:** The SD host driver can save registers for a suspend transaction after this bit has changed from 1 to 0.<br><br>0b Can issue command which uses the DAT line.<br>1b Cannot issue command which uses the DAT line. |
| 0<br>CIHB | Command Inhibit (CMD)<br><br>If this status bit is 0, it indicates that the CMD line is not in use and the SDHC can issue a SD/MMC Command using the CMD line.<br><br>This bit is set also immediately after the Transfer Type register is written. This bit is cleared when the command response is received. Even if the CDIHB bit is set to 1, Commands using only the CMD line can be issued if this bit is 0. Changing from 1 to 0 generates a command complete interrupt in the interrupt status register. If the SDHC cannot issue the command because of a command conflict error (see command CRC error) or because of a command not issued by auto CMD12 error, this bit will remain 1 and the command complete is not set. The status of issuing an auto CMD12 does not show on this bit.<br><br>0b Can issue command using only CMD line.<br>1b Cannot issue command. |

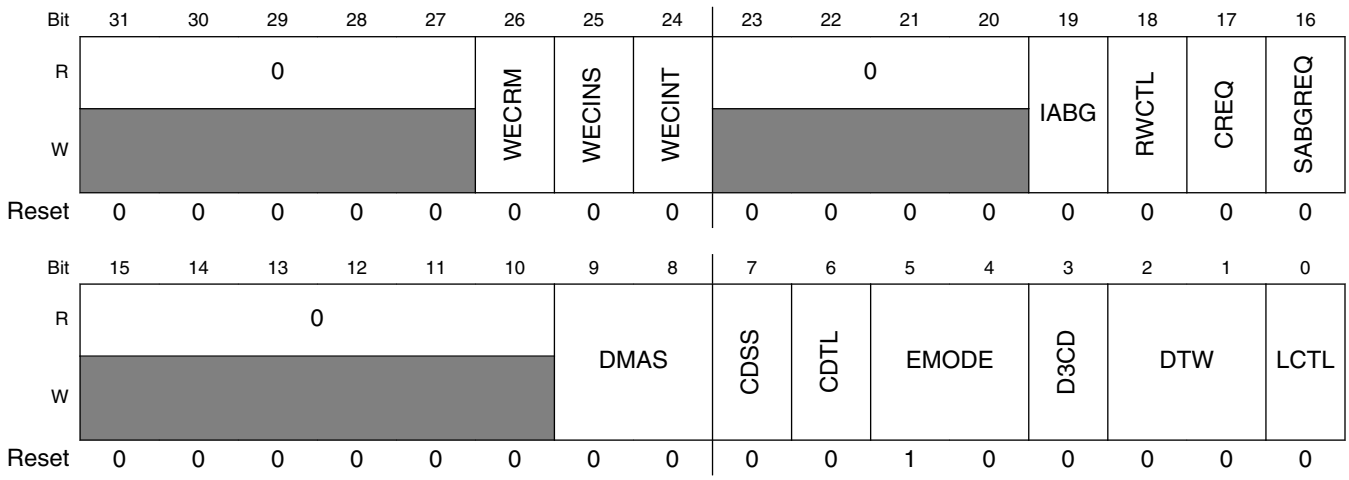
## 52.4.11 Protocol Control register (SDHC_PROCTL)

There are three cases to restart the transfer after stop at the block gap. Which case is appropriate depends on whether the SDHC issues a suspend command or the SD card accepts the suspend command:

- If the host driver does not issue a suspend command, the continue request shall be used to restart the transfer.
- If the host driver issues a suspend command and the SD card accepts it, a resume command shall be used to restart the transfer.
- If the host driver issues a suspend command and the SD card does not accept it, the continue request shall be used to restart the transfer.

Any time stop at block gap request stops the data transfer, the host driver shall wait for a transfer complete (in the interrupt status register), before attempting to restart the transfer. When restarting the data transfer by continue request, the host driver shall clear the stop at block gap request before or simultaneously.

Address: 400B_1000h base + 28h offset = 400B_1028h




## SDHC_PROCTL field descriptions

| Field | Description |
|---|---|
| 31–27<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 26<br>WECRM | Wakeup Event Enable On SD Card Removal<br><br>Enables a wakeup event, via IRQSTAT[CRM]. FN_WUS (Wake Up Support) in CIS does not effect this bit. When this bit is set, IRQSTAT[CRM] and the SDHC interrupt can be asserted without SD_CLK toggling. When the wakeup feature is not enabled, the SD_CLK must be active to assert IRQSTAT[CRM] and the SDHC interrupt.<br><br>0b Disabled<br>1b Enabled |
| 25<br>WECINS | Wakeup Event Enable On SD Card Insertion<br><br>Enables a wakeup event, via IRQSTAT[CINS]. FN_WUS (Wake Up Support) in CIS does not effect this bit. When this bit is set, IRQSTATEN[CINSEN] and the SDHC interrupt can be asserted without SD_CLK toggling. When the wakeup feature is not enabled, the SD_CLK must be active to assert IRQSTATEN[CINSEN] and the SDHC interrupt.<br><br>0b Disabled<br>1b Enabled |
| 24<br>WECINT | Wakeup Event Enable On Card Interrupt<br><br>Enables a wakeup event, via IRQSTAT[CINT]. This bit can be set to 1 if FN_WUS (Wake Up Support) in CIS is set to 1. When this bit is set, the card interrupt status and the SDHC interrupt can be asserted without SD_CLK toggling. When the wakeup feature is not enabled, the SD_CLK must be active to assert the card interrupt status and the SDHC interrupt.<br><br>0b Disabled<br>1b Enabled |
| 23–20<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 19<br>IABG | Interrupt At Block Gap<br><br>Valid only in 4-bit mode, of the SDIO card, and selects a sample point in the interrupt cycle. Setting to 1 enables interrupt detection at the block gap for a multiple block transfer. Setting to 0 disables interrupt |

*Table continues on the next page...*

## SDHC_PROCTL field descriptions (continued)

| Field | Description |
|---|---|
| | detection during a multiple block transfer. If the SDIO card can't signal an interrupt during a multiple block transfer, this bit must be set to 0 to avoid an inadvertent interrupt. When the host driver detects an SDIO card insertion, it shall set this bit according to the CCCR of the card.<br><br>0b Disabled<br>1b Enabled |
| 18<br>RWCTL | Read Wait Control<br><br>The read wait function is optional for SDIO cards. If the card supports read wait, set this bit to enable use of the read wait protocol to stop read data using the DAT[2] line. Otherwise, the SDHC has to stop the SD Clock to hold read data, which restricts commands generation. When the host driver detects an SDIO card insertion, it shall set this bit according to the CCCR of the card. If the card does not support read wait, this bit shall never be set to 1, otherwise DAT line conflicts may occur. If this bit is set to 0, stop at block gap during read operation is also supported, but the SDHC will stop the SD Clock to pause reading operation.<br><br>0b Disable read wait control, and stop SD clock at block gap when SABGREQ is set.<br>1b Enable read wait control, and assert read wait without stopping SD clock at block gap when SABGREQ bit is set. |
| 17<br>CREQ | Continue Request<br><br>Used to restart a transaction which was stopped using the PROCTL[SABGREQ]. When a suspend operation is not accepted by the card, it is also by setting this bit to restart the paused transfer. To cancel stop at the block gap, set PROCTL[SABGREQ] to 0 and set this bit to 1 to restart the transfer.<br><br>The SDHC automatically clears this bit, therefore it is not necessary for the host driver to set this bit to 0. If both PROCTL[SABGREQ] and this bit are 1, the continue request is ignored.<br><br>0b No effect.<br>1b Restart |
| 16<br>SABGREQ | Stop At Block Gap Request<br><br>Used to stop executing a transaction at the next block gap for both DMA and non-DMA transfers. Until the IRQSTATEN[TCSEN] is set to 1, indicating a transfer completion, the host driver shall leave this bit set to 1. Clearing both PROCTL[SABGREQ] and PROCTL[CREQ] does not cause the transaction to restart. Read Wait is used to stop the read transaction at the block gap. The SDHC will honor the PROCTL[SABGREQ] for write transfers, but for read transfers it requires that SDIO card support read wait. Therefore, the host driver shall not set this bit during read transfers unless the SDIO card supports read wait and has set PROCTL[RWCTL] to 1, otherwise the SDHC will stop the SD bus clock to pause the read operation during block gap. In the case of write transfers in which the host driver writes data to the data port register, the host driver shall set this bit after all block data is written. If this bit is set to 1, the host driver shall not write data to the Data Port register after a block is sent. Once this bit is set, the host driver shall not clear this bit before IRQSTATEN[TCSEN] is set, otherwise the SDHC's behavior is undefined.<br><br>This bit effects PRSSTAT[RTA], PRSSTAT[WTA], and PRSSTAT[CDIHB].<br><br>0b Transfer<br>1b Stop |
| 15–10<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 9–8<br>DMAS | DMA Select<br><br>This field is valid while DMA (SDMA or ADMA) is enabled and selects the DMA operation.<br><br>00 No DMA or simple DMA is selected. |

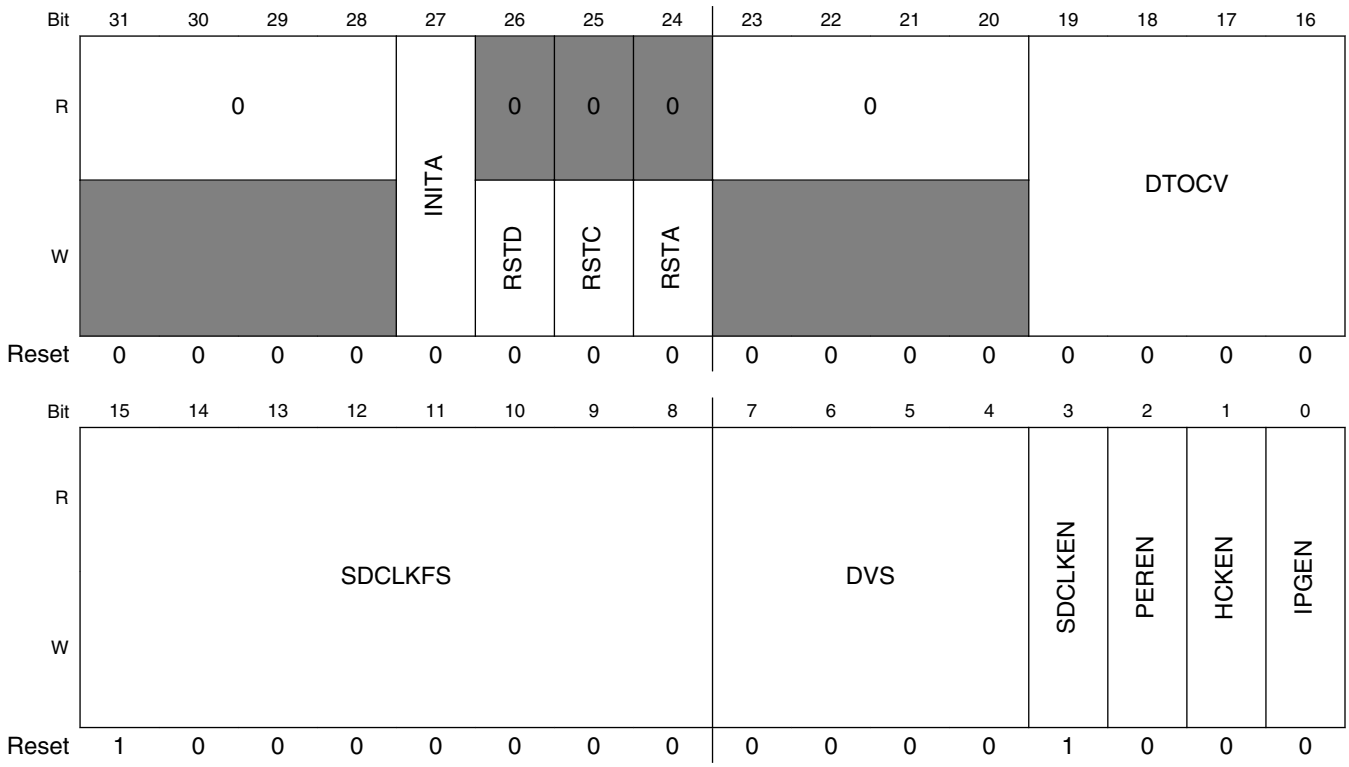
*Table continues on the next page...*

## SDHC_PROCTL field descriptions (continued)

| Field | Description |
|---|---|
| | 01 ADMA1 is selected.<br>10 ADMA2 is selected.<br>11 Reserved |
| 7<br>CDSS | Card Detect Signal Selection<br><br>Selects the source for the card detection.<br><br>0b Card detection level is selected for normal purpose.<br>1b Card detection test level is selected for test purpose. |
| 6<br>CDTL | Card Detect Test Level<br><br>Enabled while the CDSS is set to 1 and it indicates card insertion.<br><br>0b Card detect test level is 0, no card inserted.<br>1b Card detect test level is 1, card inserted. |
| 5–4<br>EMODE | Endian Mode<br><br>The SDHC supports all four endian modes in data transfer.<br><br>00b Big endian mode<br>01b Half word big endian mode<br>10b Little endian mode<br>11b Reserved |
| 3<br>D3CD | DAT3 As Card Detection Pin<br><br>If this bit is set, DAT3 should be pulled down to act as a card detection pin. Be cautious when using this feature, because DAT3 is also a chip-select for the SPI mode. A pulldown on this pin and CMD0 may set the card into the SPI mode, which the SDHC does not support. Note: Keep this bit set if SDIO interrupt is used.<br><br>0b DAT3 does not monitor card Insertion.<br>1b DAT3 as card detection pin. |
| 2–1<br>DTW | Data Transfer Width<br><br>Selects the data width of the SD bus for a data transfer. The host driver shall set it to match the data width of the card. Possible data transfer width is 1-bit, 4-bits or 8-bits.<br><br>00b 1-bit mode<br>01b 4-bit mode<br>10b 8-bit mode<br>11b Reserved |
| 0<br>LCTL | LED Control<br><br>This bit, fully controlled by the host driver, is used to caution the user not to remove the card while the card is being accessed. If the software is going to issue multiple SD commands, this bit can be set during all these transactions. It is not necessary to change for each transaction. When the software issues multiple SD commands, setting the bit once before the first command is sufficient: it is not necessary to reset the bit between commands.<br><br>0b LED off.<br>1b LED on. |

## 52.4.12 System Control register (SDHC_SYSCTL)

Address: 400B_1000h base + 2Ch offset = 400B_102Ch




### SDHC_SYSCTL field descriptions

| Field | Description |
|---|---|
| 31–28<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 27<br>INITA | Initialization Active<br><br>When this bit is set, 80 SD-clocks are sent to the card. After the 80 clocks are sent, this bit is self-cleared. This bit is very useful during the card power-up period when 74 SD-clocks are needed and the clock auto gating feature is enabled. Writing 1 to this bit when this bit is already 1 has no effect. Writing 0 to this bit at any time has no effect. When either of the PRSSTAT[CIHB] and PRSSTAT[CDIHB] bits are set, writing 1 to this bit is ignored, that is, when command line or data lines are active, write to this bit is not allowed. On the otherhand, when this bit is set, that is, during intialization active period, it is allowed to issue command, and the command bit stream will appear on the CMD pad after all 80 clock cycles are done. So when this command ends, the driver can make sure the 80 clock cycles are sent out. This is very useful when the driver needs send 80 cycles to the card and does not want to wait till this bit is self-cleared. |
| 26<br>RSTD | Software Reset For DAT Line<br>Only part of the data circuit is reset. DMA circuit is also reset.<br>The following registers and bits are cleared by this bit:<br>• Data Port register<br>• Buffer Is Cleared And Initialized.Present State register<br>• Buffer Read Enable<br>• Buffer Write Enable<br>• Read Transfer Active |

*Table continues on the next page...*

## SDHC_SYSCTL field descriptions (continued)

| Field | Description |
|---|---|
| | • Write Transfer Active<br>• DAT Line Active<br>• Command Inhibit (DAT) Protocol Control register<br>• Continue Request<br>• Stop At Block Gap Request Interrupt Status register<br>• Buffer Read Ready<br>• Buffer Write Ready<br>• DMA Interrupt<br>• Block Gap Event<br>• Transfer Complete<br><br>0b No reset.<br>1b Reset. |
| 25<br>RSTC | Software Reset For CMD Line<br>Only part of the command circuit is reset.<br>The following registers and bits are cleared by this bit:<br>• PRSSTAT[CIHB]<br>• IRQSTAT[CC]<br><br>0b No reset.<br>1b Reset. |
| 24<br>RSTA | Software Reset For ALL<br><br>Effects the entire host controller except for the card detection circuit. Register bits of type ROC, RW, RW1C, RWAC are cleared. During its initialization, the host driver shall set this bit to 1 to reset the SDHC. The SDHC shall reset this bit to 0 when the capabilities registers are valid and the host driver can read them. Additional use of software reset for all does not affect the value of the capabilities registers. After this bit is set, it is recommended that the host driver reset the external card and reinitialize it.<br><br>0b No reset.<br>1b Reset. |
| 23–20<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 19–16<br>DTOCV | Data Timeout Counter Value<br><br>Determines the interval by which DAT line timeouts are detected. See IRQSTAT[DTOE] for information on factors that dictate time-out generation. Time-out clock frequency will be generated by dividing the base clock SDCLK value by this value.<br><br>The host driver can clear IRQSTATEN[DTOESEN] to prevent inadvertent time-out events.<br><br>0000b SDCLK x $2^{13}$<br>0001b SDCLK x $2^{14}$<br>...<br>1110b SDCLK x $2^{27}$<br>1111b Reserved |
| 15–8<br>SDCLKFS | SDCLK Frequency Select<br><br>Used to select the frequency of the SDCLK pin. The frequency is not programmed directly. Rather this register holds the prescaler (this register) and divisor (next register) of the base clock frequency register.<br><br>Setting 00h bypasses the frequency prescaler of the SD Clock. Multiple bits must not be set, or the behavior of this prescaler is undefined. The two default divider values can be calculated by the frequency of SDHC clock and the following divisor bits. |

*Table continues on the next page...*

## SDHC_SYSCTL field descriptions (continued)

| Field | Description |
|---|---|
| | The frequency of SDCLK is set by the following formula: Clock frequency = (Base clock) / (prescaler x divisor)<br><br>For example, if the base clock frequency is 96 MHz, and the target frequency is 25 MHz, then choosing the prescaler value of 01h and divisor value of 1h will yield 24 MHz, which is the nearest frequency less than or equal to the target. Similarly, to approach a clock value of 400 kHz, the prescaler value of 08h and divisor value of eh yields the exact clock value of 400 kHz. The reset value of this field is 80h, so if the input base clock ( SDHC clock ) is about 96 MHz, the default SD clock after reset is 375 kHz.<br><br>According to the SD Physical Specification Version 1.1 and the SDIO Card Specification Version 1.2, the maximum SD clock frequency is 50 MHz and shall never exceed this limit.<br><br>Only the following settings are allowed:<br><br>01h Base clock divided by 2.<br>02h Base clock divided by 4.<br>04h Base clock divided by 8.<br>08h Base clock divided by 16.<br>10h Base clock divided by 32.<br>20h Base clock divided by 64.<br>40h Base clock divided by 128.<br>80h Base clock divided by 256. |
| 7–4<br>DVS | Divisor<br><br>Used to provide a more exact divisor to generate the desired SD clock frequency. Note the divider can even support odd divisor without deterioration of duty cycle.<br><br>The setting are as following:<br><br>0h Divisor by 1.<br>1h Divisor by 2.<br>...<br>Eh Divisor by 15.<br>Fh Divisor by 16. |
| 3<br>SDCLKEN | SD Clock Enable<br><br>The host controller shall stop SDCLK when writing this bit to 0. SDCLK frequency can be changed when this bit is 0. Then, the host controller shall maintain the same clock frequency until SDCLK is stopped (stop at SDCLK = 0). If the IRQSTAT[CINS] is cleared, this bit must be cleared by the host driver to save power. |
| 2<br>PEREN | Peripheral Clock Enable<br>If this bit is set, SDHC clock will always be active and no automatic gating is applied. Thus the SDCLK is active except for when auto gating-off during buffer danger (buffer about to over-run or under-run). When this bit is cleared, the SDHC clock will be automatically off whenever there is no transaction on the SD bus. Because this bit is only a feature enabling bit, clearing this bit does not stop SDCLK immediately. The SDHC clock will be internally gated off, if none of the following factors are met:<br>• The cmd part is reset, or<br>• Data part is reset, or<br>• A soft reset, or<br>• The cmd is about to send, or<br>• Clock divisor is just updated, or<br>• Continue request is just set, or<br>• This bit is set, or<br>• Card insertion is detected, or<br>• Card removal is detected, or |

*Table continues on the next page...*

SDHC_SYSCTL field descriptions (continued)

| Field | Description |
|---|---|
| | • Card external interrupt is detected, or<br>• 80 clocks for initialization phase is ongoing<br><br>0b SDHC clock will be internally gated off.<br>1b SDHC clock will not be automatically gated off. |
| 1<br>HCKEN | System Clock Enable<br><br>If this bit is set, system clock will always be active and no automatic gating is applied. When this bit is cleared,<br><br>system clock<br><br>will be automatically off when no data transfer is on the SD bus.<br><br>0b System clock will be internally gated off.<br>1b System clock will not be automatically gated off. |
| 0<br>IPGEN | IPG Clock Enable<br>If this bit is set, bus clock will always be active and no automatic gating is applied.<br>The bus clock will be internally gated off, if none of the following factors are met:<br>• The cmd part is reset, or<br>• Data part is reset, or<br>• Soft reset, or<br>• The cmd is about to send, or<br>• Clock divisor is just updated, or<br>• Continue request is just set, or<br>• This bit is set, or<br>• Card insertion is detected, or<br>• Card removal is detected, or<br>• Card external interrupt is detected, or<br>• The SDHC clock is not gated off<br><br>**NOTE:** The bus clock will not be auto gated off if the SDHC clock is not gated off. So clearing only this bit has no effect unless the PEREN bit is also cleared.<br><br>0b Bus clock will be internally gated off.<br>1b Bus clock will not be automatically gated off. |

## 52.4.13 Interrupt Status register (SDHC_IRQSTAT)

An interrupt is generated when the Normal Interrupt Signal Enable is enabled and at least one of the status bits is set to 1. For all bits, writing 1 to a bit clears it; writing to 0 keeps the bit unchanged. More than one status can be cleared with a single register write. For Card Interrupt, before writing 1 to clear, it is required that the card stops asserting the interrupt, meaning that when the Card Driver services the interrupt condition, otherwise the CINT bit will be asserted again.

The table below shows the relationship between the CTOE and the CC bits.

**Table 52-19. SDHC status for CTOE/CC bit combinations**

| Command complete | Command timeout error | Meaning of the status |
|---|---|---|
| 0 | 0 | X |
| X | 1 | Response not received within 64 SDCLK cycles |
| 1 | 0 | Response received |

The table below shows the relationship between the Transfer Complete and the Data Timeout Error.

**Table 52-20. SDHC status for data timeout error/transfer complete bit combinations**

| Transfer complete | Data timeout error | Meaning of the status |
|---|---|---|
| 0 | 0 | X |
| 0 | 1 | Timeout occurred during transfer |
| 1 | X | Data transfer complete |

The table below shows the relationship between the command CRC Error (CCE) and Command Timeout Error (CTOE).

**Table 52-21. SDHC status for CCE/CTOE Bit Combinations**

| Command complete | Command timeout error | Meaning of the status |
|---|---|---|
| 0 | 0 | No error |
| 0 | 1 | Response timeout error |
| 1 | 0 | Response CRC error |
| 1 | 1 | CMD line conflict |

Address: 400B_1000h base + 30h offset = 400B_1030h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | DMAE | 0 | | | AC12E | 0 | DEBE | DCE | DTOE | CIE | CEBE | CCE | CTOE |
| 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | CINT | CRM | CINS | BRR | BWR | DINT | BGE | TC | CC |
| 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 |

## SDHC_IRQSTAT field descriptions

| Field | Description |
|---|---|
| 31–29<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 28<br>DMAE | DMA Error<br><br>Occurs when an Internal DMA transfer has failed. This bit is set to 1, when some error occurs in the data transfer. This error can be caused by either Simple DMA or ADMA, depending on which DMA is in use. The value in DMA System Address register is the next fetch address where the error occurs. Because any error corrupts the whole data block, the host driver shall restart the transfer from the corrupted block boundary. The address of the block boundary can be calculated either from the current DSADDR value or from the remaining number of blocks and the block size.<br><br>0b No error.<br>1b Error. |
| 27–25<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 24<br>AC12E | Auto CMD12 Error<br><br>Occurs when detecting that one of the bits in the Auto CMD12 Error Status register has changed from 0 to 1. This bit is set to 1, not only when the errors in Auto CMD12 occur, but also when the Auto CMD12 is not executed due to the previous command error.<br><br>0b No error.<br>1b Error. |
| 23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22<br>DEBE | Data End Bit Error<br><br>Occurs either when detecting 0 at the end bit position of read data, which uses the DAT line, or at the end bit position of the CRC.<br><br>0b No error.<br>1b Error. |
| 21<br>DCE | Data CRC Error<br><br>Occurs when detecting a CRC error when transferring read data, which uses the DAT line, or when detecting the Write CRC status having a value other than 010.<br><br>0b No error.<br>1b Error. |

*Table continues on the next page...*

## SDHC_IRQSTAT field descriptions (continued)

| Field | Description |
|---|---|
| 20<br>DTOE | Data Timeout Error<br><br>Occurs when detecting one of following time-out conditions.<br>• Busy time-out for R1b,R5b type<br>• Busy time-out after Write CRC status<br>• Read Data time-out<br><br>0b No error.<br>1b Time out. |
| 19<br>CIE | Command Index Error<br><br>Occurs if a Command Index error occurs in the command response.<br><br>0b No error.<br>1b Error. |
| 18<br>CEBE | Command End Bit Error<br><br>Occurs when detecting that the end bit of a command response is 0.<br><br>0b No error.<br>1b End Bit Error generated. |
| 17<br>CCE | Command CRC Error<br><br>Command CRC Error is generated in two cases.<br>• If a response is returned and the Command Timeout Error is set to 0, indicating no time-out, this bit is set when detecting a CRC error in the command response.<br>• The SDHC detects a CMD line conflict by monitoring the CMD line when a command is issued. If the SDHC drives the CMD line to 1, but detects 0 on the CMD line at the next SDCLK edge, then the SDHC shall abort the command (Stop driving CMD line) and set this bit to 1. The Command Timeout Error shall also be set to 1 to distinguish CMD line conflict.<br><br>0b No error.<br>1b CRC Error generated. |
| 16<br>CTOE | Command Timeout Error<br><br>Occurs only if no response is returned within 64 SDCLK cycles from the end bit of the command. If the SDHC detects a CMD line conflict, in which case a Command CRC Error shall also be set, this bit shall be set without waiting for 64 SDCLK cycles. This is because the command will be aborted by the SDHC.<br><br>0b No error.<br>1b Time out. |
| 15–9<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 8<br>CINT | Card Interrupt<br><br>This status bit is set when an interrupt signal is detected from the external card. In 1-bit mode, the SDHC will detect the Card Interrupt without the SD Clock to support wakeup. In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, so the interrupt from card can only be sampled during interrupt cycle, introducing some delay between the interrupt signal from the SDIO card and the interrupt to the host system. Writing this bit to 1 can clear this bit, but as the interrupt factor from the SDIO card does not clear, this bit is set again. To clear this bit, it is required to reset the interrupt factor from the external card followed by a writing 1 to this bit. |

*Table continues on the next page...*

## SDHC_IRQSTAT field descriptions (continued)

| Field | Description |
|---|---|
| | When this status has been set, and the host driver needs to service this interrupt, the Card Interrupt Signal Enable in the Interrupt Signal Enable register should be 0 to stop driving the interrupt signal to the host system. After completion of the card interrupt service (it must reset the interrupt factors in the SDIO card and the interrupt signal may not be asserted), write 1 to clear this bit, set the Card Interrupt Signal Enable to 1, and start sampling the interrupt signal again.<br><br>0b No Card Interrupt.<br>1b Generate Card Interrupt. |
| 7<br>CRM | Card Removal<br><br>This status bit is set if the Card Inserted bit in the Present State register changes from 1 to 0. When the host driver writes this bit to 1 to clear this status, the status of the Card Inserted in the Present State register must be confirmed. Because the card state may possibly be changed when the host driver clears this bit and the interrupt event may not be generated. When this bit is cleared, it will be set again if no card is inserted. To leave it cleared, clear the Card Removal Status Enable bit in Interrupt Status Enable register.<br><br>0b Card state unstable or inserted.<br>1b Card removed. |
| 6<br>CINS | Card Insertion<br><br>This status bit is set if the Card Inserted bit in the Present State register changes from 0 to 1. When the host driver writes this bit to 1 to clear this status, the status of the Card Inserted in the Present State register must be confirmed. Because the card state may possibly be changed when the host driver clears this bit and the interrupt event may not be generated. When this bit is cleared, it will be set again if a card is inserted. To leave it cleared, clear the Card Inserted Status Enable bit in Interrupt Status Enable register.<br><br>0b Card state unstable or removed.<br>1b Card inserted. |
| 5<br>BRR | Buffer Read Ready<br><br>This status bit is set if the Buffer Read Enable bit, in the Present State register, changes from 0 to 1. See the Buffer Read Enable bit in the Present State register for additional information.<br><br>0b Not ready to read buffer.<br>1b Ready to read buffer. |
| 4<br>BWR | Buffer Write Ready<br><br>This status bit is set if the Buffer Write Enable bit, in the Present State register, changes from 0 to 1. See the Buffer Write Enable bit in the Present State register for additional information.<br><br>0b Not ready to write buffer.<br>1b Ready to write buffer. |
| 3<br>DINT | DMA Interrupt<br><br>Occurs only when the internal DMA finishes the data transfer successfully. Whenever errors occur during data transfer, this bit will not be set. Instead, the DMAE bit will be set. Either Simple DMA or ADMA finishes data transferring, this bit will be set.<br><br>0b No DMA Interrupt.<br>1b DMA Interrupt is generated. |

*Table continues on the next page...*

**SDHC_IRQSTAT field descriptions (continued)**

| Field | Description |
|---|---|
| 2<br>BGE | Block Gap Event<br><br>If PROCTL[SABGREQ] is set, this bit is set when a read or write transaction is stopped at a block gap. If PROCTL[SABGREQ] is not set to 1, this bit is not set to 1.<br><br>In the case of a read transaction: This bit is set at the falling edge of the DAT line active status, when the transaction is stopped at SD Bus timing. The read wait must be supported in order to use this function.<br><br>In the case of write transaction: This bit is set at the falling edge of write transfer active status, after getting CRC status at SD bus timing.<br><br>0b No block gap event.<br>1b Transaction stopped at block gap. |
| 1<br>TC | Transfer Complete<br><br>This bit is set when a read or write transfer is completed.<br><br>In the case of a read transaction: This bit is set at the falling edge of the read transfer active status. There are two cases in which this interrupt is generated. The first is when a data transfer is completed as specified by the data length, after the last data has been read to the host system. The second is when data has stopped at the block gap and completed the data transfer by setting PROCTL[SABGREQ], after valid data has been read to the host system.<br><br>In the case of a write transaction: This bit is set at the falling edge of the DAT line active status. There are two cases in which this interrupt is generated. The first is when the last data is written to the SD card as specified by the data length and the busy signal is released. The second is when data transfers are stopped at the block gap, by setting PROCTL[SABGREQ], and the data transfers are completed,after valid data is written to the SD card and the busy signal released.<br><br>0b Transfer not complete.<br>1b Transfer complete. |
| 0<br>CC | Command Complete<br><br>This bit is set when you receive the end bit of the command response, except Auto CMD12. See PRSSTAT[CIHB].<br><br>0b Command not complete.<br>1b Command complete. |

## 52.4.14 Interrupt Status Enable register (SDHC_IRQSTATEN)

Setting the bits in this register to 1 enables the corresponding interrupt status to be set by the specified event. If any bit is cleared, the corresponding interrupt status bit is also cleared, that is, when the bit in this register is cleared, the corresponding bit in interrupt status register is always 0.

## NOTE

- Depending on PROCTL[IABG] bit setting, SDHC may be programmed to sample the card interrupt signal during the interrupt period and hold its value in the flip-flop. There will be some delays on the card interrupt, asserted from the card, to the time the host system is informed.
- To detect a CMD line conflict, the host driver must set both IRQSTATEN[CTOESEN] and IRQSTATEN[CCESEN] to 1.

Address: 400B_1000h base + 34h offset = 400B_1034h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | DMAESEN | 0 | | | AC12ESEN | 0 | DEBESEN | DCESEN | DTOESEN | CIESEN | CEBESEN | CCESEN | CTOESEN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | CINTSEN | CRMSEN | CINSEN | BRRSEN | BWRSEN | DINTSEN | BGESEN | TCSEN | CCSEN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

### SDHC_IRQSTATEN field descriptions

| Field | Description |
|---|---|
| 31–29<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 28<br>DMAESEN | DMA Error Status Enable<br><br>0b Masked<br>1b Enabled |
| 27–25<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 24<br>AC12ESEN | Auto CMD12 Error Status Enable<br><br>0b Masked<br>1b Enabled |
| 23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22<br>DEBESEN | Data End Bit Error Status Enable<br><br>0b Masked<br>1b Enabled |

*Table continues on the next page...*

## SDHC_IRQSTATEN field descriptions (continued)

| Field | Description |
|---|---|
| 21<br>DCESEN | Data CRC Error Status Enable<br><br>0b Masked<br>1b Enabled |
| 20<br>DTOESEN | Data Timeout Error Status Enable<br><br>0b Masked<br>1b Enabled |
| 19<br>CIESEN | Command Index Error Status Enable<br><br>0b Masked<br>1b Enabled |
| 18<br>CEBESEN | Command End Bit Error Status Enable<br><br>0b Masked<br>1b Enabled |
| 17<br>CCESEN | Command CRC Error Status Enable<br><br>0b Masked<br>1b Enabled |
| 16<br>CTOESEN | Command Timeout Error Status Enable<br><br>0b Masked<br>1b Enabled |
| 15–9<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 8<br>CINTSEN | Card Interrupt Status Enable<br><br>If this bit is set to 0, the SDHC will clear the interrupt request to the system. The card interrupt detection is stopped when this bit is cleared and restarted when this bit is set to 1. The host driver must clear the this bit before servicing the card interrupt and must set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts.<br><br>0b Masked<br>1b Enabled |
| 7<br>CRMSEN | Card Removal Status Enable<br><br>0b Masked<br>1b Enabled |
| 6<br>CINSEN | Card Insertion Status Enable<br><br>0b Masked<br>1b Enabled |
| 5<br>BRRSEN | Buffer Read Ready Status Enable<br><br>0b Masked<br>1b Enabled |
| 4<br>BWRSEN | Buffer Write Ready Status Enable |

*Table continues on the next page...*

**SDHC_IRQSTATEN field descriptions (continued)**

| Field | Description |
|---|---|
| | 0b Masked<br>1b Enabled |
| 3<br>DINTSEN | DMA Interrupt Status Enable<br><br>0b Masked<br>1b Enabled |
| 2<br>BGESEN | Block Gap Event Status Enable<br><br>0b Masked<br>1b Enabled |
| 1<br>TCSEN | Transfer Complete Status Enable<br><br>0b Masked<br>1b Enabled |
| 0<br>CCSEN | Command Complete Status Enable<br><br>0b Masked<br>1b Enabled |

## 52.4.15 Interrupt Signal Enable register (SDHC_IRQSIGEN)

This register is used to select which interrupt status is indicated to the host system as the interrupt. All of these status bits share the same interrupt line. Setting any of these bits to 1 enables interrupt generation. The corresponding status register bit will generate an interrupt when the corresponding interrupt signal enable bit is set.

Address: 400B_1000h base + 38h offset = 400B_1038h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | DMAEIEN | 0 | | | AC12EIEN | 0 | DEBEIEN | DCEIEN | DTOEIEN | CIEIEN | CEBEIEN | CCEIEN | CTOEIEN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | CINTIEN | CRMIEN | CINSIEN | BRRIEN | BWRIEN | DINTIEN | BGEIEN | TCIEN | CCIEN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## SDHC_IRQSIGEN field descriptions

| Field | Description |
|---|---|
| 31–29<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 28<br>DMAEIEN | DMA Error Interrupt Enable<br><br>0b Masked<br>1b Enabled |
| 27–25<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 24<br>AC12EIEN | Auto CMD12 Error Interrupt Enable<br><br>0b Masked<br>1b Enabled |
| 23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22<br>DEBEIEN | Data End Bit Error Interrupt Enable<br><br>0b Masked<br>1b Enabled |
| 21<br>DCEIEN | Data CRC Error Interrupt Enable<br><br>0b Masked<br>1b Enabled |
| 20<br>DTOEIEN | Data Timeout Error Interrupt Enable<br><br>0b Masked<br>1b Enabled |
| 19<br>CIEIEN | Command Index Error Interrupt Enable<br><br>0b Masked<br>1b Enabled |
| 18<br>CEBEIEN | Command End Bit Error Interrupt Enable<br><br>0b Masked<br>1b Enabled |
| 17<br>CCEIEN | Command CRC Error Interrupt Enable<br><br>0b Masked<br>1b Enabled |
| 16<br>CTOEIEN | Command Timeout Error Interrupt Enable<br><br>0b Masked<br>1b Enabled |
| 15–9<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 8<br>CINTIEN | Card Interrupt Enable<br><br>0b Masked<br>1b Enabled |

*Table continues on the next page...*

SDHC_IRQSIGEN field descriptions (continued)

| Field | Description |
|---|---|
| 7<br>CRMIEN | Card Removal Interrupt Enable<br><br>0b Masked<br>1b Enabled |
| 6<br>CINSIEN | Card Insertion Interrupt Enable<br><br>0b Masked<br>1b Enabled |
| 5<br>BRRIEN | Buffer Read Ready Interrupt Enable<br><br>0b Masked<br>1b Enabled |
| 4<br>BWRIEN | Buffer Write Ready Interrupt Enable<br><br>0b Masked<br>1b Enabled |
| 3<br>DINTIEN | DMA Interrupt Enable<br><br>0b Masked<br>1b Enabled |
| 2<br>BGEIEN | Block Gap Event Interrupt Enable<br><br>0b Masked<br>1b Enabled |
| 1<br>TCIEN | Transfer Complete Interrupt Enable<br><br>0b Masked<br>1b Enabled |
| 0<br>CCIEN | Command Complete Interrupt Enable<br><br>0b Masked<br>1b Enabled |

## 52.4.16 Auto CMD12 Error Status Register (SDHC_AC12ERR)

When the AC12ESEN bit in the Status register is set, the host driver shall check this register to identify what kind of error the Auto CMD12 indicated. This register is valid only when the Auto CMD12 Error status bit is set.

The following table shows the relationship between the Auto CMGD12 CRC error and the Auto CMD12 command timeout error.

**Table 52-25. Relationship between Command CRC Error and Command Timeout Error For Auto CMD12**

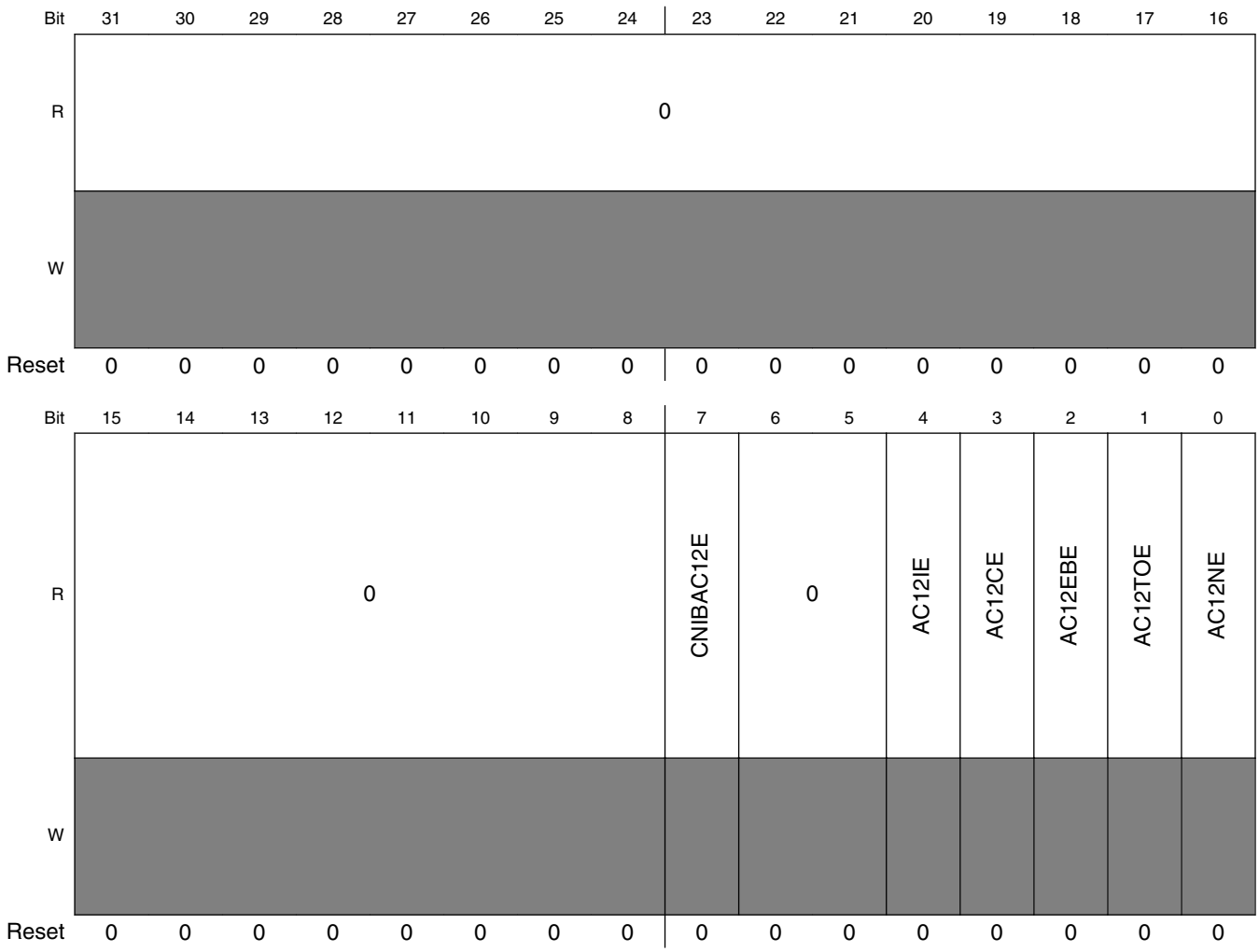
| Auto CMD12 CRC error | Auto CMD12 timeout error | Type of error |
|---|---|---|
| 0 | 0 | No error |
| 0 | 1 | Response timeout error |
| 1 | 0 | Response CRC error |
| 1 | 1 | CMD line conflict |

Changes in Auto CMD12 Error Status register can be classified in three scenarios:

1. When the SDHC is going to issue an Auto CMD12:
   - Set bit 0 to 1 if the Auto CMD12 can't be issued due to an error in the previous command.
   - Set bit 0 to 0 if the auto CMD12 is issued.
2. At the end bit of an auto CMD12 response:
   - Check errors corresponding to bits 1-4.
   - Set bits 1-4 corresponding to detected errors.
   - Clear bits 1-4 corresponding to detected errors.
3. Before reading the Auto CMD12 error status bit 7:
   - Set bit 7 to 1 if there is a command that can't be issued.
   - Clear bit 7 if there is no command to issue.

The timing for generating the auto CMD12 error and writing to the command register are asynchronous. After that, bit 7 shall be sampled when the driver is not writing to the command register. So it is suggested to read this register only when IRQSTAT[AC12E] is set. An Auto CMD12 error interrupt is generated when one of the error bits (0-4) is set to 1. The command not issued by auto CMD12 error does not generate an interrupt.

Address: 400B_1000h base + 3Ch offset = 400B_103Ch




## SDHC_AC12ERR field descriptions

| Field | Description |
|---|---|
| 31–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>CNIBAC12E | Command Not Issued By Auto CMD12 Error<br><br>Setting this bit to 1 means CMD_wo_DAT is not executed due to an auto CMD12 error (D04-D01) in this register.<br><br>0b No error.<br>1b Not issued. |
| 6–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>AC12IE | Auto CMD12 Index Error<br><br>Occurs if the command index error occurs in response to a command.<br><br>0b No error.<br>1b Error, the CMD index in response is not CMD12. |

*Table continues on the next page...*

## SDHC_AC12ERR field descriptions (continued)

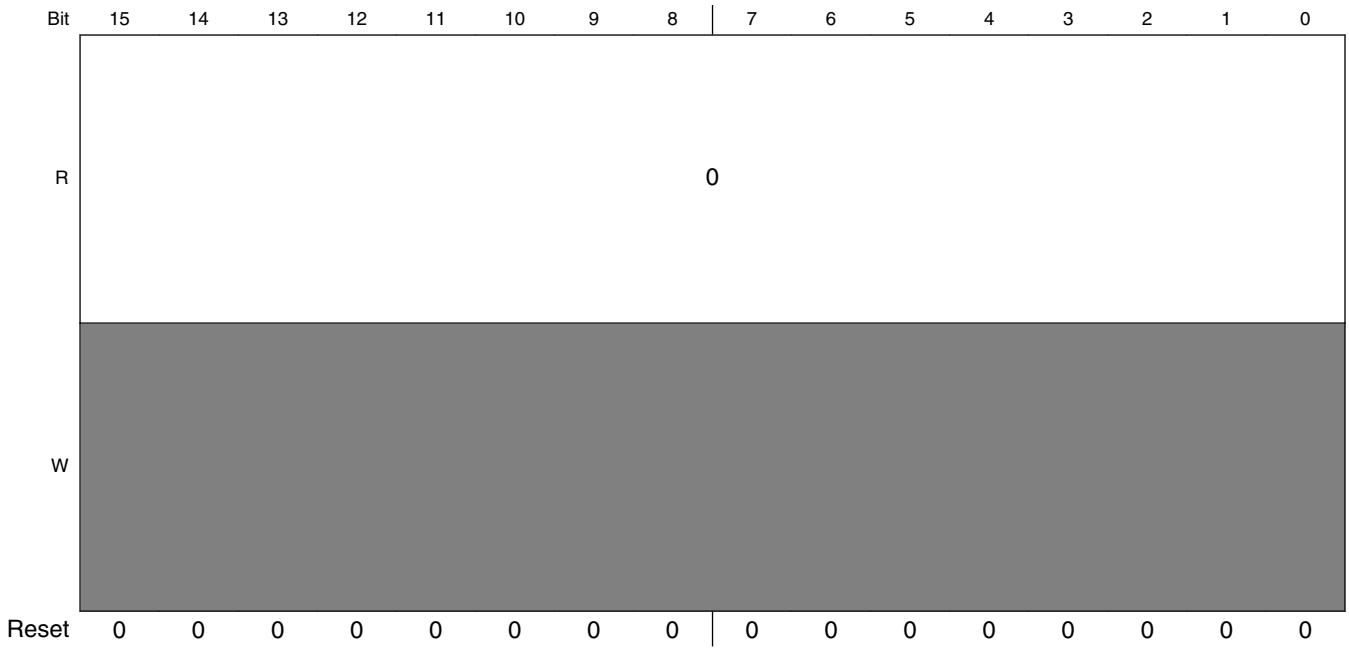
| Field | Description |
|---|---|
| 3<br>AC12CE | Auto CMD12 CRC Error<br><br>Occurs when detecting a CRC error in the command response.<br><br>0b No CRC error.<br>1b CRC error met in Auto CMD12 response. |
| 2<br>AC12EBE | Auto CMD12 End Bit Error<br><br>Occurs when detecting that the end bit of command response is 0 which must be 1.<br><br>0b No error.<br>1b End bit error generated. |
| 1<br>AC12TOE | Auto CMD12 Timeout Error<br><br>Occurs if no response is returned within 64 SDCLK cycles from the end bit of the command. If this bit is set to 1, the other error status bits (2-4) have no meaning.<br><br>0b No error.<br>1b Time out. |
| 0<br>AC12NE | Auto CMD12 Not Executed<br><br>If memory multiple block data transfer is not started, due to a command error, this bit is not set because it is not necessary to issue an auto CMD12. Setting this bit to 1 means the SDHC cannot issue the auto CMD12 to stop a memory multiple block data transfer due to some error. If this bit is set to 1, other error status bits (1-4) have no meaning.<br><br>0b Executed.<br>1b Not executed. |

## 52.4.17 Host Controller Capabilities (SDHC_HTCAPBLT)

This register provides the host driver with information specific to the SDHC implementation. The value in this register is the power-on-reset value, and does not change with a software reset. Any write to this register is ignored.

Address: 400B_1000h base + 40h offset = 400B_1040h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | Reserved | Reserved | VS33 | SRS | DMAS | HSS | ADMAS | 0 | MBL | | |
| W | | | | | | Reserved | Reserved | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |




## SDHC_HTCAPBLT field descriptions

| Field | Description |
|---|---|
| 31–27<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 26<br>Reserved | This field is reserved. |
| 25<br>Reserved | This field is reserved. |
| 24<br>VS33 | Voltage Support 3.3 V<br><br>This bit shall depend on the host system ability.<br><br>0b 3.3 V not supported.<br>1b 3.3 V supported. |
| 23<br>SRS | Suspend/Resume Support<br><br>This bit indicates whether the SDHC supports suspend / resume functionality. If this bit is 0, the suspend and resume mechanism, as well as the read Wwait, are not supported, and the host driver shall not issue either suspend or resume commands.<br><br>0b Not supported.<br>1b Supported. |
| 22<br>DMAS | DMA Support<br><br>This bit indicates whether the SDHC is capable of using the internal DMA to transfer data between system memory and the data buffer directly.<br><br>0b DMA not supported.<br>1b DMA supported. |
| 21<br>HSS | High Speed Support |

*Table continues on the next page...*

**SDHC_HTCAPBLT field descriptions (continued)**

| Field | Description |
|---|---|
| | This bit indicates whether the SDHC supports high speed mode and the host system can supply a SD Clock frequency from 25 MHz to 50 MHz.<br><br>0b High speed not supported.<br>1b High speed supported. |
| 20<br>ADMAS | ADMA Support<br><br>This bit indicates whether the SDHC supports the ADMA feature.<br><br>0b Advanced DMA not supported.<br>1b Advanced DMA supported. |
| 19<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 18–16<br>MBL | Max Block Length<br><br>This value indicates the maximum block size that the host driver can read and write to the buffer in the SDHC. The buffer shall transfer block size without wait cycles.<br><br>000b 512 bytes<br>001b 1024 bytes<br>010b 2048 bytes<br>011b 4096 bytes |
| 15–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 52.4.18 Watermark Level Register (SDHC_WML)

Both write and read watermark levels (FIFO threshold) are configurable. There value can range from 1 to 128 words. Both write and read burst lengths are also configurable. There value can range from 1 to 31 words.

Address: 400B_1000h base + 44h offset = 400B_1044h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | 0 | | | | | WRWML | | | | | | | | 0 | | | 0 | | | | | RDWML | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**SDHC_WML field descriptions**

| Field | Description |
|---|---|
| 31–29<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 28–24<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

*Table continues on the next page...*

**SDHC_WML field descriptions (continued)**

| Field | Description |
|---|---|
| 23–16<br>WRWML | Write Watermark Level<br><br>The number of words used as the watermark level (FIFO threshold) in a DMA write operation. Also the number of words as a sequence of write bursts in back-to-back mode. The maximum legal value for the write watermark level is 128. |
| 15–13<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 12–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7–0<br>RDWML | Read Watermark Level<br><br>The number of words used as the watermark level (FIFO threshold) in a DMA read operation. Also the number of words as a sequence of read bursts in back-to-back mode. The maximum legal value for the read water mark level is 128. |

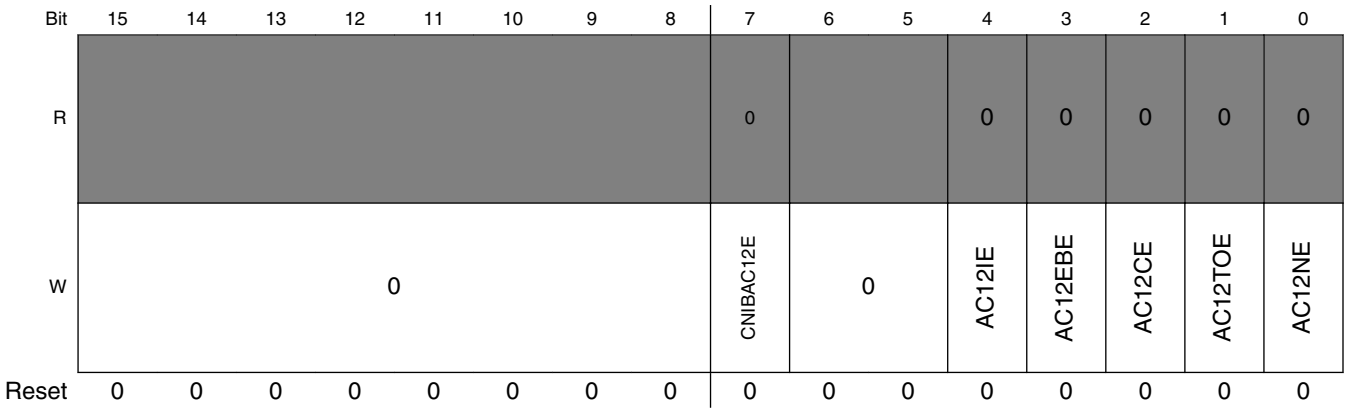
## 52.4.19 Force Event register (SDHC_FEVT)

The Force Event (FEVT) register is not a physically implemented register. Rather, it is an address at which the Interrupt Status register can be written if the corresponding bit of the Interrupt Status Enable register is set. This register is a write only register and writing 0 to it has no effect. Writing 1 to this register actually sets the corresponding bit of Interrupt Status register. A read from this register always results in 0's. To change the corresponding status bits in the interrupt status register, make sure to set SYSCTL[IPGEN] so that bus clock is always active.

Forcing a card interrupt will generate a short pulse on the DAT[1] line, and the driver may treat this interrupt as a normal interrupt. The interrupt service routine may skip polling the card interrupt factor as the interrupt is selfcleared.

Address: 400B_1000h base + 50h offset = 400B_1050h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | 0 | | | | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | CINT | 0 | | DMAE | 0 | | | AC12E | 0 | DEBE | DCE | DTOE | CIE | CEBE | CCE | CTOE |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |




## SDHC_FEVT field descriptions

| Field | Description |
|---|---|
| 31<br>CINT | Force Event Card Interrupt<br><br>Writing 1 to this bit generates a short low-level pulse on the internal DAT[1] line, as if a self-clearing interrupt was received from the external card. If enabled, the CINT bit will be set and the interrupt service routine may treat this interrupt as a normal interrupt from the external card. |
| 30–29<br>Reserved | This field is reserved. |
| 28<br>DMAE | Force Event DMA Error<br><br>Forces the DMAE bit of Interrupt Status Register to be set. |
| 27–25<br>Reserved | This field is reserved. |
| 24<br>AC12E | Force Event Auto Command 12 Error<br><br>Forces IRQSTAT[AC12E] to be set. |
| 23<br>Reserved | This field is reserved. |
| 22<br>DEBE | Force Event Data End Bit Error<br><br>Forces IRQSTAT[DEBE] to be set. |
| 21<br>DCE | Force Event Data CRC Error<br><br>Forces IRQSTAT[DCE] to be set. |
| 20<br>DTOE | Force Event Data Time Out Error<br><br>Forces IRQSTAT[DTOE] to be set. |
| 19<br>CIE | Force Event Command Index Error<br><br>Forces IRQSTAT[CCE] to be set. |
| 18<br>CEBE | Force Event Command End Bit Error<br><br>Forces IRQSTAT[CEBE] to be set. |
| 17<br>CCE | Force Event Command CRC Error<br><br>Forces IRQSTAT[CCE] to be set. |

*Table continues on the next page...*

**SDHC_FEVT field descriptions (continued)**

| Field | Description |
|---|---|
| 16<br>CTOE | Force Event Command Time Out Error<br><br>Forces IRQSTAT[CTOE] to be set. |
| 15–8<br>Reserved | This field is reserved. |
| 7<br>CNIBAC12E | Force Event Command Not Executed By Auto Command 12 Error<br><br>Forces AC12ERR[CNIBAC12E] to be set. |
| 6–5<br>Reserved | This field is reserved. |
| 4<br>AC12IE | Force Event Auto Command 12 Index Error<br><br>Forces AC12ERR[AC12IE] to be set. |
| 3<br>AC12EBE | Force Event Auto Command 12 End Bit Error<br><br>Forces AC12ERR[AC12EBE] to be set. |
| 2<br>AC12CE | Force Event Auto Command 12 CRC Error<br><br>Forces AC12ERR[AC12CE] to be set. |
| 1<br>AC12TOE | Force Event Auto Command 12 Time Out Error<br><br>Forces AC12ERR[AC12TOE] to be set. |
| 0<br>AC12NE | Force Event Auto Command 12 Not Executed<br><br>Forces AC12ERR[AC12NE] to be set. |

## 52.4.20 ADMA Error Status register (SDHC_ADMAES)

When an ADMA error interrupt has occurred, the ADMA Error States field in this register holds the ADMA state and the ADMA System Address register holds the address around the error descriptor.

For recovering from this error, the host driver requires the ADMA state to identify the error descriptor address as follows:

- ST_STOP: Previous location set in the ADMA System Address register is the error descriptor address.
- ST_FDS: Current location set in the ADMA System Address register is the error descriptor address.

- ST_CADR: This state is never set because it only increments the descriptor pointer and doesn't generate an ADMA error.
- ST_TFR: Previous location set in the ADMA System Address register is the error descriptor address.

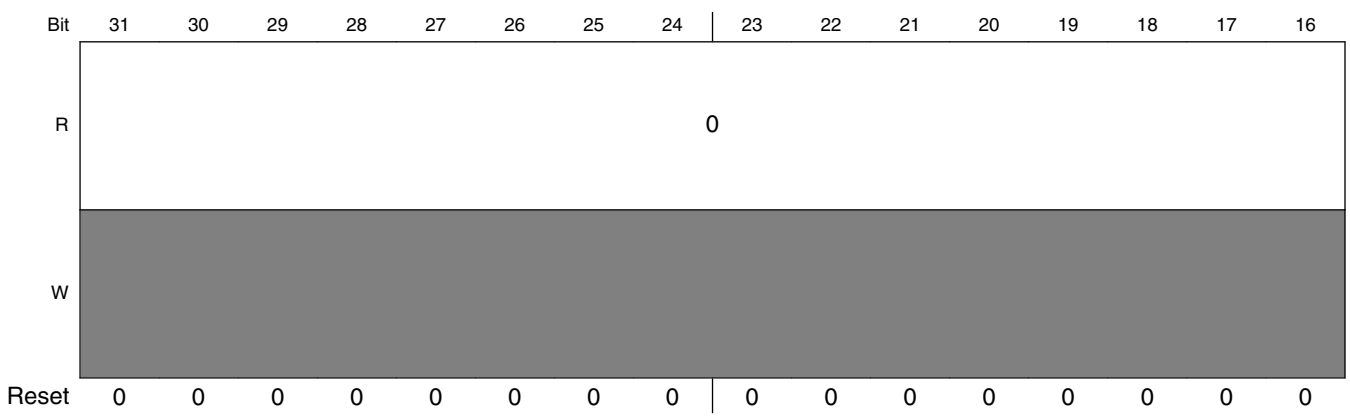
In case of a write operation, the host driver must use the ACMD22 to get the number of the written block, rather than using this information, because unwritten data may exist in the host controller.

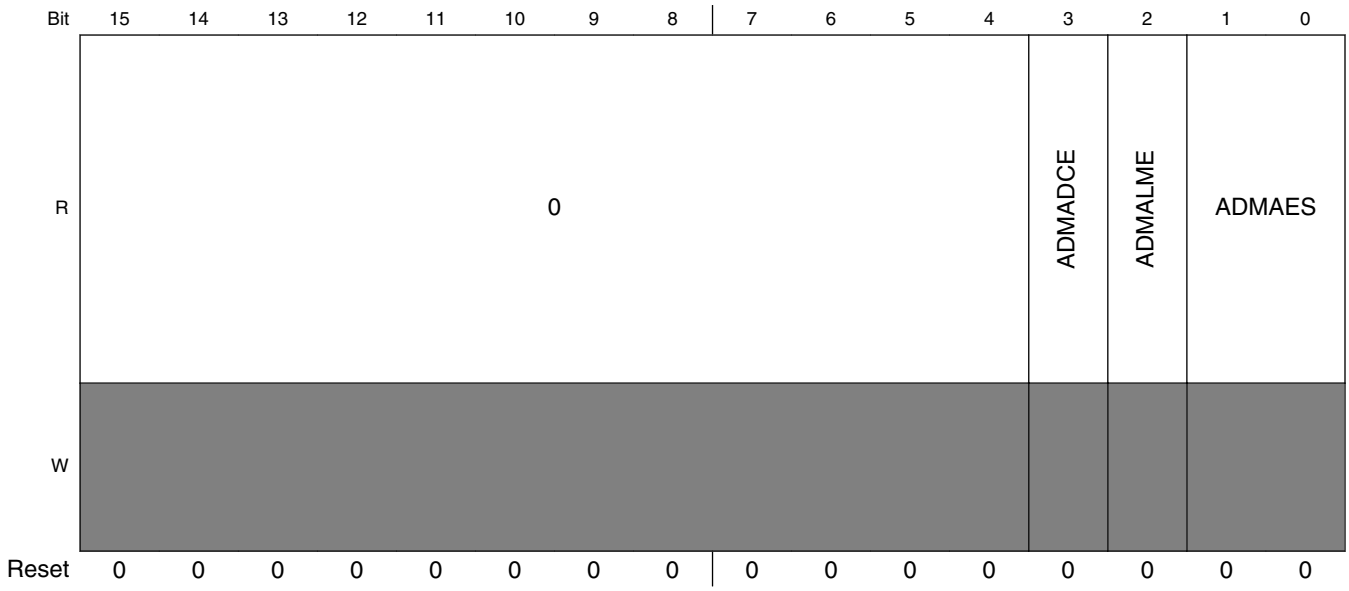
The host controller generates the ADMA error interrupt when it detects invalid descriptor data (valid = 0) in the ST_FDS state. The host driver can distinguish this error by reading the valid bit of the error descriptor.

**Table 52-30. ADMA Error State coding**

| D01-D00 | ADMA Error State when error has occurred | Contents of ADMA System Address register |
|---|---|---|
| 00 | ST_STOP (Stop DMA) | Holds the address of the next executable descriptor command |
| 01 | ST_FDS (fetch descriptor) | Holds the valid descriptor address |
| 10 | ST_CADR (change address) | No ADMA error is generated |
| 11 | ST_TFR (Transfer Data) | Holds the address of the next executable descriptor command |

Address: 400B_1000h base + 54h offset = 400B_1054h






## SDHC_ADMAES field descriptions

| Field | Description |
|---|---|
| 31–4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3<br>ADMADCE | ADMA Descriptor Error<br><br>This error occurs when an invalid descriptor is fetched by ADMA.<br><br>0b No error.<br>1b Error. |
| 2<br>ADMALME | ADMA Length Mismatch Error<br><br>This error occurs in the following 2 cases:<br>• While the block count enable is being set, the total data length specified by the descriptor table is different from that specified by the block count and block length.<br>• Total data length can not be divided by the block length.<br><br>0b No error.<br>1b Error. |
| 1–0<br>ADMAES | ADMA Error State (When ADMA Error Is Occurred.)<br><br>Indicates the state of the ADMA when an error has occurred during an ADMA data transfer. |

## 52.4.21 ADMA System Addressregister (SDHC_ADSADDR)

This register contains the physical system memory address used for ADMA transfers.

Address: 400B_1000h base + 58h offset = 400B_1058h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ADSADDR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

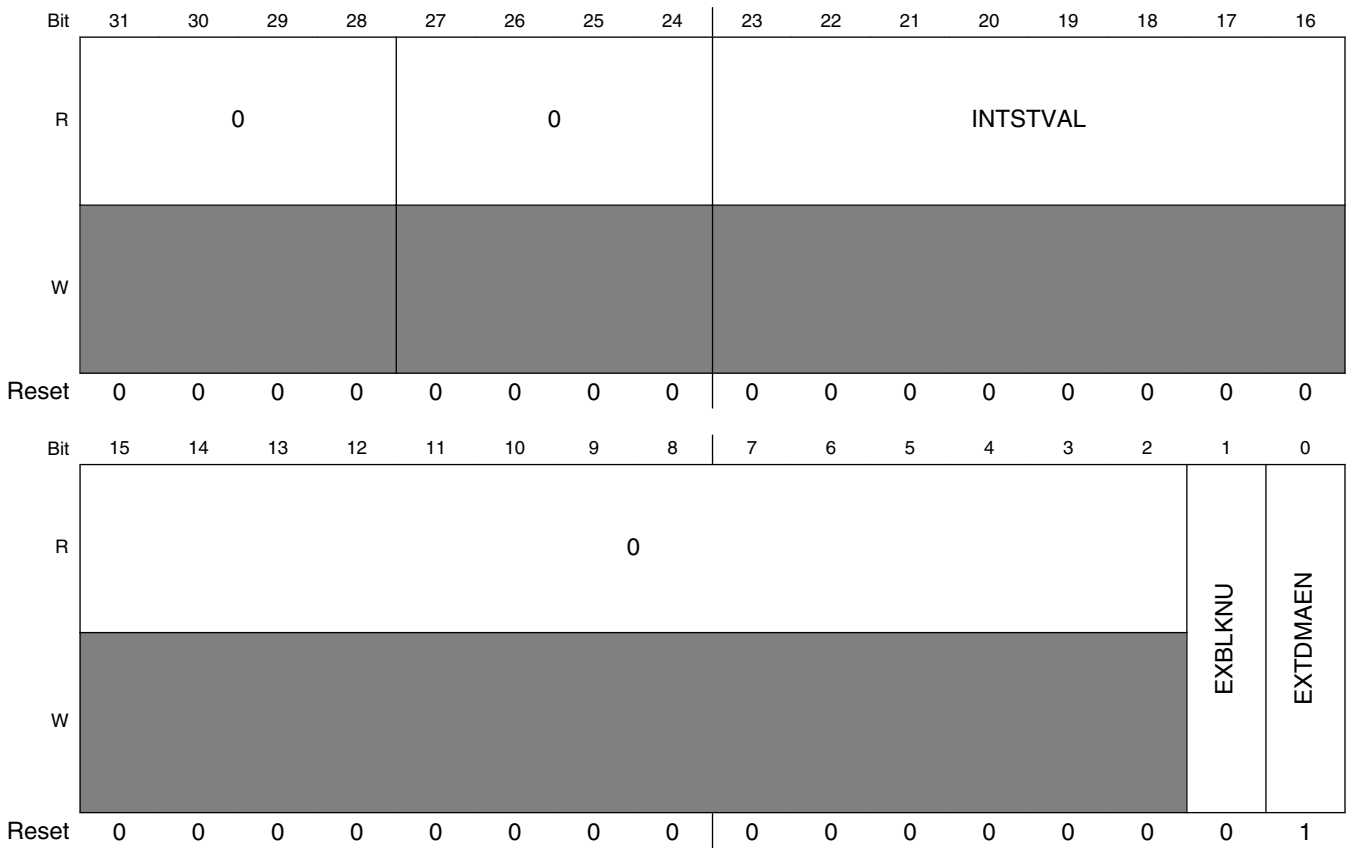
### SDHC_ADSADDR field descriptions

| Field | Description |
|---|---|
| 31–2<br>ADSADDR | ADMA System Address<br><br>Holds the word address of the executing command in the descriptor table. At the start of ADMA, the host driver shall set the start address of the Descriptor table. The ADMA engine increments this register address whenever fetching a descriptor command. When the ADMA is stopped at the block gap, this register indicates the address of the next executable descriptor command. When the ADMA error interrupt is generated, this register shall hold the valid descriptor address depending on the ADMA state. The lower 2 bits of this register is tied to '0' so the ADMA address is always word-aligned. Because this register supports dynamic address reflecting, when TC bit is set, it automatically alters the value of internal address counter, so SW cannot change this register when TC bit is set. |
| 1–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 52.4.22 Vendor Specific register (SDHC_VENDOR)

This register contains the vendor-specific control/status register.

Address: 400B_1000h base + C0h offset = 400B_10C0h




### SDHC_VENDOR field descriptions

| Field | Description |
|---|---|
| 31–28<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 27–24<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 23–16<br>INTSTVAL | Internal State Value<br><br>Internal state value, reflecting the corresponding state value selected by **Debug Select** field. This field is read-only and write to this field does not have effect. |
| 15–2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 1<br>EXBLKNU | Exact Block Number Block Read Enable For SDIO CMD53<br><br>This bit must be set before S/W issues CMD53 multi-block read with exact block number. This bit must not be set if the CMD53 multi-block read is not exact block number. |

*Table continues on the next page...*

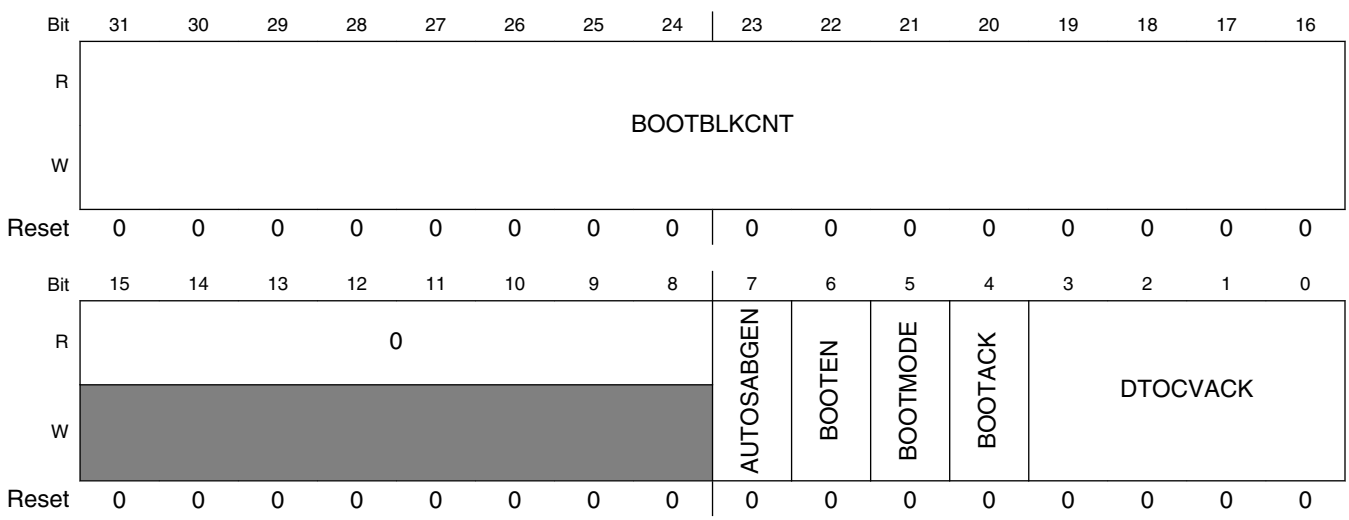
**SDHC_VENDOR field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 None exact block read.<br>1 Exact block read for SDIO CMD53. |
| 0<br>EXTDMAEN | External DMA Request Enable<br><br>Enables the request to external DMA. When the internal DMA (either simple DMA or advanced DMA) is not in use, and this bit is set, SDHC will send out DMA request when the internal buffer is ready. This bit is particularly useful when transferring data by CPU polling mode, and it is not allowed to send out the external DMA request. By default, this bit is set.<br><br>0 In any scenario, SDHC does not send out the external DMA request.<br>1 When internal DMA is not active, the external DMA request will be sent out. |

## 52.4.23 MMC Boot register (SDHC_MMCBOOT)

This register contains the MMC fast boot control register.

Address: 400B_1000h base + C4h offset = 400B_10C4h




**SDHC_MMCBOOT field descriptions**

| Field | Description |
|---|---|
| 31–16<br>BOOTBLKCNT | Defines the stop at block gap value of automatic mode. When received card block cnt is equal to BOOTBLKCNT and AUTOSABGEN is 1, then stop at block gap. |
| 15–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>AUTOSABGEN | When boot, enable auto stop at block gap function. This function will be triggered, and host will stop at block gap when received card block cnt is equal to BOOTBLKCNT. |
| 6<br>BOOTEN | Boot Mode Enable<br><br>0 Fast boot disable.<br>1 Fast boot enable. |

*Table continues on the next page...*

**SDHC_MMCBOOT field descriptions (continued)**

| Field | Description |
|---|---|
| 5<br>BOOTMODE | Boot Mode Select<br><br>0 Normal boot.<br>1 Alternative boot. |
| 4<br>BOOTACK | Boot Ack Mode Select<br><br>0 No ack.<br>1 Ack. |
| 3–0<br>DTOCVACK | Boot ACK Time Out Counter Value<br><br>0000b SDCLK x 2^8<br>0001b SDCLK x 2^9<br>0010b SDCLK x 2^10<br>0011b SDCLK x 2^11<br>0100b SDCLK x 2^12<br>0101b SDCLK x 2^13<br>0110b SDCLK x 2^14<br>0111b SDCLK x 2^15<br>...<br>1110b SDCLK x 2^22<br>1111b Reserved |

## 52.4.24 Host Controller Version (SDHC_HOSTVER)

This register contains the vendor host controller version information. All bits are read only and will read the same as the power-reset value.

Address: 400B_1000h base + FCh offset = 400B_10FCh

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | VVN | | | | | | | | SVN | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**SDHC_HOSTVER field descriptions**

| Field | Description |
|---|---|
| 31–16<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 15–8<br>VVN | Vendor Version Number<br><br>These status bits are reserved for the vendor version number. The host driver shall not use this status.<br><br>00h Freescale SDHC version 1.0<br>10h Freescale SDHC version 2.0<br>11h Freescale SDHC version 2.1 |

*Table continues on the next page...*

**SDHC_HOSTVER field descriptions (continued)**

| Field | Description |
|---|---|
| | 12h Freescale SDHC version 2.2<br>All others Reserved |
| 7–0<br>SVN | Specification Version Number<br><br>These status bits indicate the host controller specification version.<br><br>01h SD host specification version 2.0, supports test event register and ADMA.<br>All others Reserved |

## 52.5 Functional description

The following sections provide a brief functional description of the major system blocks, including the data buffer, DMA crossbar switch interface, dual-port memory wrapper, data/command controller, clock & reset manager, and clock generator.

### 52.5.1 Data buffer

The SDHC uses one configurable data buffer, so that data can be transferred between the system bus and the SD card, with an optimized manner to maximize throughput between the two clock domains (that is, the IP peripheral clock, and the master clock). The following diagram illustrates the buffer scheme. The buffer is used as temporary storage for data being transferred between the host system and the card. The watermark levels for read and write are both configurable, and can be any number from 1 to 128 words. The burst lengths for read and write are also configurable, and can be any number from 1 to 31 words.

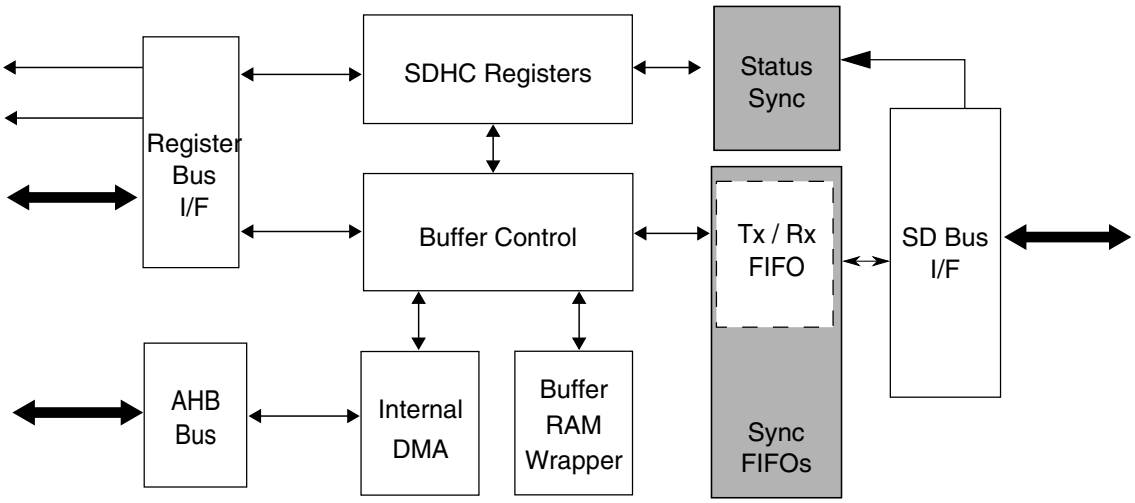



**Figure 52-27. SDHC buffer scheme**

There are 3 transfer modes to access the data buffer:

- CPU Polling mode:
  - For a host read operation, when the number of words received in the buffer meets or exceeds the RDWML watermark value, then by polling IRQSTAT[BRR] the host driver can read the DATPORT register to fetch the amount of words set in the WML register from the buffer. The write operation is similar.
- External DMA mode:
  - For a read operation, when there are more words received in the buffer than the amount set in the RDWML register, a DMA request is sent out to inform the external DMA to fetch the data. The request will be immediately deasserted when there is an access on the DATPORT register. If the number of words in the buffer after the current burst meets or exceeds RDWML value, then the DMA request is asserted again. For instance if there are twice as many words in the buffer than the RDWML value, there are two successive DMA requests with only one cycle of deassertion between. The write operation is similar.

    Note the accesses CPU Polling mode and External DMA mode both use the IP bus, and if the external DMA is enable, in both modes an external DMA request is sent out whenever the buffer is ready.
- Internal DMA mode (includes simple and advanced DMA access's):
  - The internal DMA access, either by simple or advanced DMA, is over the crossbar switch bus. For internal DMA access mode, the external DMA request will never be sent out.

For a read operation, when there are more words in the buffer than the amount set in WML, the internal DMA starts fetching data over the crossbar switch bus. Except INCR4 and INCR8, the burst type is always INCR mode and the burst length depends on the shortest of following factors:

1. Burst length configured in the burst length field of WML
2. Watermark level boundary
3. Block size boundary
4. Data boundary configured in the current descriptor if the ADMA is active
5. 1 KB address boundary

Write operation is similar.

Sequential and contiguous access is necessary to ensure the pointer address value is correct. Random or skipped access is not possible. The byte order, by reset, is little endian mode. The actually byte order is swapped inside the buffer, according to the endian mode configured by software, as illustrated in the following diagrams. For a host write operation, byte order is swapped after data is fetched from the buffer and ready to send to the SD bus. For a host read operation, byte order is swapped before the data is stored into the buffer.

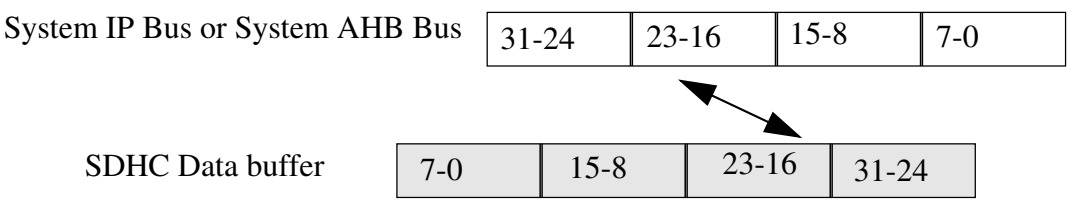



**Figure 52-28. Data swap between system bus and SDHC data buffer in byte little endian mode**

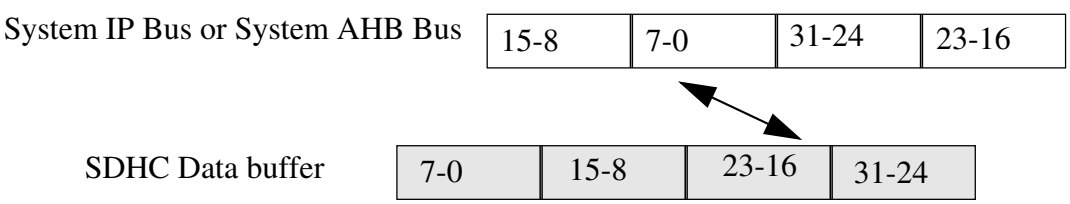



**Figure 52-29. Data swap between system bus and SDHC data buffer in half word big endian mode**

### 52.5.1.1 Write operation sequence

There are three ways to write data into the buffer when the user transfers data to the card:

1. Using external DMA through the SDHC DMA request signal.
2. Processor core polling through IRQSTAT[BWR] (interrupt or polling).
3. Using the internal DMA.

When the internal DMA is not used, that is, the XFERTYP[DMAEN] bit is not set when the command is sent, the SDHC asserts a DMA request when the amount of buffer space exceeds the value set in WML, and is ready for receiving new data. At the same time, the SDHC would set IRQSTAT[BWR]. The buffer write ready interrupt will be generated if it is enabled by software.

When internal DMA is used, the SDHC will not inform the system before all the required number of bytes are transferred if no error was encountered. When an error occurs during the data transfer, the SDHC will abort the data transfer and abandon the current block. The host driver must read the contents of the DSADDR to get the starting address of the abandoned data block. If the current data transfer is in multiblock mode, the SDHC will not automatically send CMD12, even though XFERTYP[AC12EN] is set. The host driver shall send CMD12 in this scenario and restart the write operation from that address. It is recommended that a software reset for data be applied before the transfer is restarted after error recovery.

The SDHC will not start data transmission until the number of words set in WML can be held in the buffer. If the buffer is empty and the host system does not write data in time, the SDHC will stop the SD_CLK to avoid the data buffer underrun situation.

#### 52.5.1.2 Read operation sequence

There are three ways to read data from the buffer when the user transfers data to the card:

1. Using the external DMA through the SDHC DMA request signal
2. Processor core polling through IRQSTAT[BRR] (interrupt or polling)
3. Using the internal DMA

When internal DMA is not used, that is, XFERTYP[DMAEN] is not set when the command is sent, the SDHC asserts a DMA request when the amount of data exceeds the value set in WML, that is available and ready for system fetching data. At the same time, the SDHC would set IRQSTAT[BRR]. The buffer read ready interrupt will be generated if it is enabled by software.

When internal DMA is used, the SDHC will not inform the system before all the required number of bytes are transferred if no error was encountered. When an error occurs during the data transfer, the SDHC will abort the data transfer and abandon the current block. The host driver must read the content of the DMA system address register to get the starting address of the abandoned data block. If the current data transfer is in multiblock mode, the SDHC will not automatically send CMD12, even though XFERTYP[AC12EN] is set. The host driver shall send CMD12 in this scenario and restart the read operation from that address. It is recommended that a software reset for data be applied before the transfer is restarted after error recovery.

For any write transfer mode, the SDHC will not start data transmission until the number of words set in WML are in the buffer. If the buffer is full and the Host System does not read data in time, the SDHC will stop the SDHC_DCLK to avoid the data buffer overrun situation.

#### 52.5.1.3 Data buffer and block size

The user needs to know the buffer size for the buffer operation during a data transfer to use it in the most optimized way. In the SDHC, the only data buffer can hold up to 128 words (32-bit), and the watermark levels for write and read can be configured respectively. For both read and write, the watermark level can be from 1 word to the maximum of 128 words. For both read and write, the burst length can be from 1 word to the maximum of 31 words. The host driver may configure the value according to the system situation and requirement.

During a multiblock data transfer, the block length may be set to any value between 1 and 4096 bytes inclusive, which satisfies the requirements of the external card. The only restriction is from the external card. It might not support that large of a block or it doesn't support a partial block access, which is not the integer times of 512 bytes.

For block size not times of 4, that is, not word aligned, SDHC requires stuff bytes at the end of each block, because the SDHC treats each block individually. For example, the block size is 7 bytes, there are 12 blocks to write, the system side must write 2 times for each block, and for each block, the ending byte would be abandoned by the SDHC because it sends only 7 bytes to the card and picks data from the following system write, so there would be 24 beats of write access in total.

#### 52.5.1.4 Dividing large data transfer

This SDIO command CMD53 definition, limits the maximum data size of data transfers according to the following formula:

Max data size = Block size x Block count

The length of a multiple block transfer needs to be in block size units. If the total data length can't be divided evenly into a multiple of the block size, then there are two ways to transfer the data which depend on the function and the card design. Option 1 is for the host driver to split the transaction. The remainder of the block size data is then transferred by using a single block command at the end. Option 2 is to add dummy data in the last block to fill the block size. For option 2, the card must manage the removal of the dummy data.

The following diagram illustrates the dividing of large data transfers. Assuming a kind of WLAN SDIO card supports block size only up to 64 bytes. Although the SDHC supports a block size of up to 4096 bytes, the SDIO can only accept a block size less than 64 bytes, so the data must be divided. See the example below.

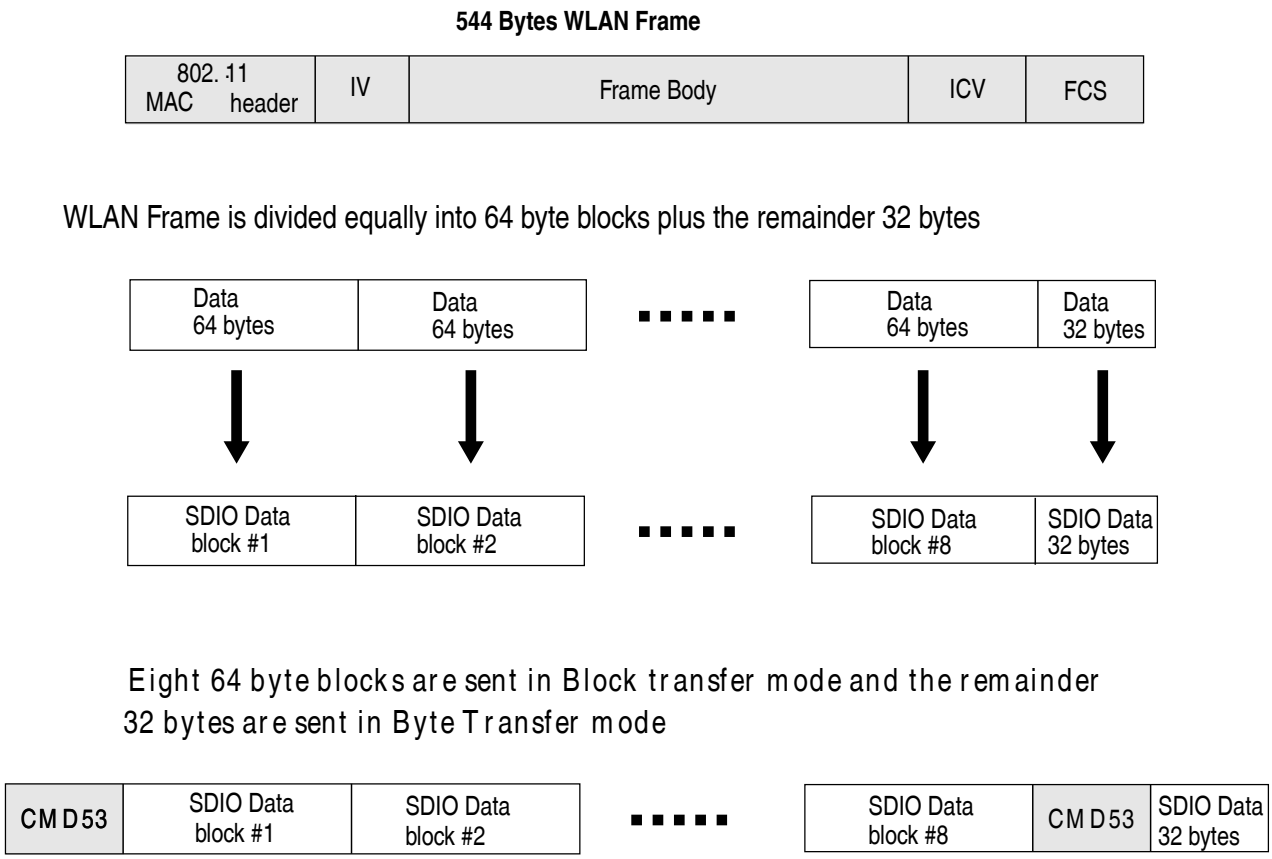



**Figure 52-30. Example for dividing large data transfers**

### 52.5.1.5 External DMA request

When the internal DMA is not in use, and external DMA request is enabled, the data buffer will generate a DMA request to the system. During a write operation, when the number of WRWML words can be held in the buffer free space, a DMA request is sent , informing the host system of a DMA write. IRQSTAT[BWR] is also set, as long as IRQSTATEN[BWRSEN] is set. The DMA request is immediately deasserted when an access to the DATPORT register is made. If the buffer's free space still meets the watermark condition, the DMA request is asserted again after a cycle.

On read operation, when the number of RDWML words are already in the buffer, a DMA request is sent, informing the host system for a DMA read. IRQSTAT[BRR] is also set, as long as IRQSTATEN[BRRSEN] is set. The DMA request is immediately deasserted when an access to the DATPORT register is made. If the buffer's data still meets the watermark condition, the DMA request is asserted again after a cycle.

Because the DMA burst length can't change during a data transfer for an external DMA transfer, the watermark level (read or write) must be a divisor of the block size. If it is not, transferring of the block may cause buffer underrun for read operation or overrun for write operation. For example, if the block size is 512 bytes, the watermark level of read (or write) must be a power of 2 between 1 and 128. For processor core polling access, as the last access in the block transfer can be controlled by software, there is no such issue. The watermark level can be any value, even larger than the block size but not greater than 128 words. This is because the actual number of bytes transferred by the software can be controlled and does not exceed the block size in each transfer.

The SDHC also supports non-word aligned block size, as long as the card supports that block size. In this case, the watermark level must be set as the number of words. For example, if the block size is 31 bytes, the watermark level can be set to any number of word. For this case, the BLKATTR[BLKSIZE] bits shall be set as 1fh. For the CPU polling access, the burst length can be 1 to 128 words, without restriction. This is because the software will transfer 8 words, and the SDHC will also set the IRQSTAT[BWR] or IRQSTAT[BRR] bits when the remaining data does not violate data buffer. See DMA burst length for more details about the dynamic watermark level of the data buffer. For the above example, even though 8 words are transferred via the DATPORT register, the SDHC will transfer only 31 bytes over the SD bus, as required by the BLKATTR[BLKSIZE] bits. In this data transfer, with non-word aligned block size, the endian mode must be set cautiously, or invalid data will be transferred to/from the card.

## 52.5.2 DMA crossbar switch interface

The internal DMA implements a DMA engine and the crossbar switch master. When the internal DMA is enabled (XFERTYP[DMAEN] is set), the interrupt status bits are set if they are enabled. To avoid setting them, clear IRQSTATEN[BWRSEN, BRRSEN]. The following diagram illustrates the DMA crossbar switch interface block.

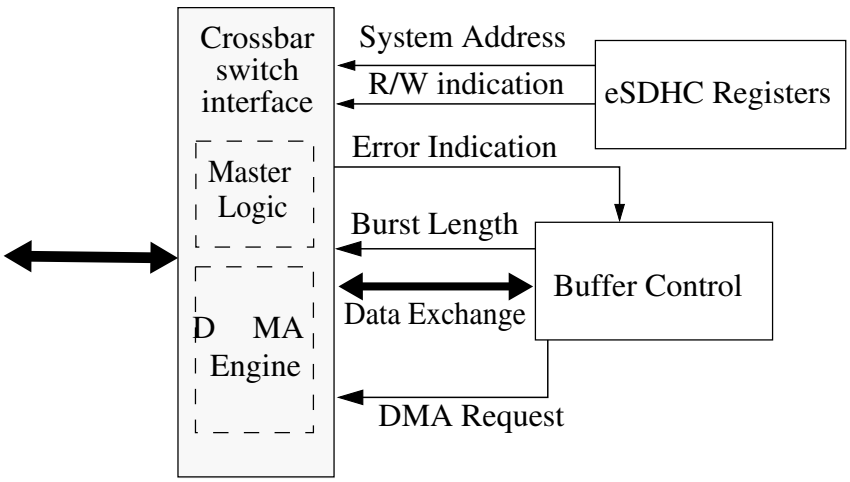



**Figure 52-31. DMA crossbar switch interface block**

### 52.5.2.1 Internal DMA request

If the watermark level requirement is met in data transfer, and the internal DMA is enabled, the data buffer block sends a DMA request to the crossbar switch interface. Meanwhile, the external DMA request signal is disabled. The delay in response from the internal DMA engine depends on the system bus loading and the priority assigned to the SDHC. The DMA engine does not respond to the request during its burst transfer, but is ready to serve as soon as the burst is over. The data buffer de-asserts the request once an access to the buffer is made. Upon access to the buffer by internal DMA, the data buffer updates its internal buffer pointer, and when the watermark level is satisfied, another DMA request is sent.

The data transfer is in the block unit, and the subsequent watermark level is always set as the remaining number of words. For instance, for a multiblock data read with each block size of 31 bytes, and the burst length set to 6 words. After the first burst transfer, if there are more than 2 words in the buffer, which might contain some data of the next block, another DMA request read is sent. This is because the remaining number of words to send for the current block is (31 - 6 * 4) / 4 = 2. The SDHC will read 2 words out of the buffer, with 7 valid bytes and 1 stuff byte.

### 52.5.2.2 DMA burst length

Just like a CPU polling access, the DMA burst length for the internal DMA engine can be from 1 to 16 words. The actual burst length for the DMA depends on the lesser of the configured burst length or the remaining words of the current block. Take the example in Internal DMA request again. The following burst length after 6 words are read will be 2 words, and the next burst length will be 6 words again. This is because the next block starts, which is 31 bytes, more than 6 words. The host driver writer may take this variable burst length into account. It is also acceptable to configure the burst length as the divisor of the block size, so that each time the burst length will be the same.

### 52.5.2.3 Crossbar switch master interface

It is possible that the internal DMA engine could fail during the data transfer. When this error occurs, the DMA engine stops the transfer and goes to the idle state as well as the internal data buffer stops accepting incoming data. IRQSTAT[DMAE] is set to inform the driver.

After the DMAE interrupt is received, the software shall send a CMD12 to abort the current transfer and read DSADDR[DSADDR] to get the starting address of the corrupted block. After the DMA error is fixed, the software must apply a data reset and restart the transfer from this address to recover the corrupted block.

### 52.5.2.4 ADMA engine

In the SDHC standard, the new DMA transfer algorithm called the advanced DMA (ADMA) is defined. For simple DMA, once the page boundary is reached, a DMA interrupt will be generated and the new system address shall be programmed by the host driver. The ADMA defines the programmable descriptor table in the system memory. The host driver can calculate the system address at the page boundary and program the descriptor table before executing ADMA. It reduces the frequency of interrupts to the host system. Therefore, higher speed DMA transfers could be realized because the host MCU intervention would not be needed during long DMA-based data transfers.

There are two types of ADMA: ADMA1 and ADMA2 in host controller. ADMA1 can support data transfer of 4 KB aligned data in system memory. ADMA2 improves the restriction so that data of any location and any size can be transferred in system memory. Their formats of descriptor table are different.

ADMA can recognize all kinds of descriptors defined in the SDHC standard, and if end flag is detected in the descriptor, ADMA will stop after this descriptor is processed.

#### 52.5.2.4.1 ADMA concept and descriptor format

For ADMA1, including the following descriptors:

- Valid/Invalid descriptor
- Nop descriptor
- Set data length descriptor
- Set data address descriptor
- Link descriptor
- Interrupt flag and end flag in descriptor

For ADMA2, including the following descriptors:

- Valid/Invalid descriptor
- Nop descriptor
- Rsv descriptor
- Set data length and address descriptor
- Link descriptor
- Interrupt flag and end flag in descriptor

ADMA2 deals with the lower 32-bit first, and then the higher 32-bit. If the Valid flag of descriptor is 0, it will ignore the high 32-bit. The address field shall be set on word aligned (lower 2-bit is always set to 0). Data length is in byte unit.

ADMA will start read/write operation after it reaches the tran state, using the data length and data address analyzed from most recent descriptor(s).

For ADMA1, the valid data length descriptor is the last set type descriptor before tran type descriptor. Every tran type will trigger a transfer, and the transfer data length is extracted from the most recent set type descriptor. If there is no set type descriptor after the previous trans descriptor, the data length will be the value for previous transfer, or 0 if no set descriptor is ever met.

For ADMA2, tran type descriptor contains both the data length and transfer data address, so only a tran type descriptor can start a data transfer.

## Table 52-35. Format of the ADMA1 descriptor table

| Address/page field | | Address/page field | | Attribute field | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 31 | 12 | 11 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Address or data length | | 000000 | | Act2 | Act1 | 0 | Int | End | Valid |

| Act2 | Act1 | Symbol | Comment | 31-28 | 27-12 |
|---|---|---|---|---|---|
| 0 | 0 | nop | No operation | Don't care | |
| 0 | 1 | set | Set data length | 0000 | Data length |
| 1 | 0 | tran | Transfer data | Data address | |
| 1 | 1 | link | Link descriptor | Descriptor address | |

| | |
|---|---|
| Valid | Valid = 1 indicates this line of descriptor is effective. If Valid = 0 generate ADMA error Interrupt and stop ADMA. |
| End | End = 1 indicates current descriptor is the ending one. |
| Int | Int = 1 generates DMA interrupt when this descriptor is processed. |

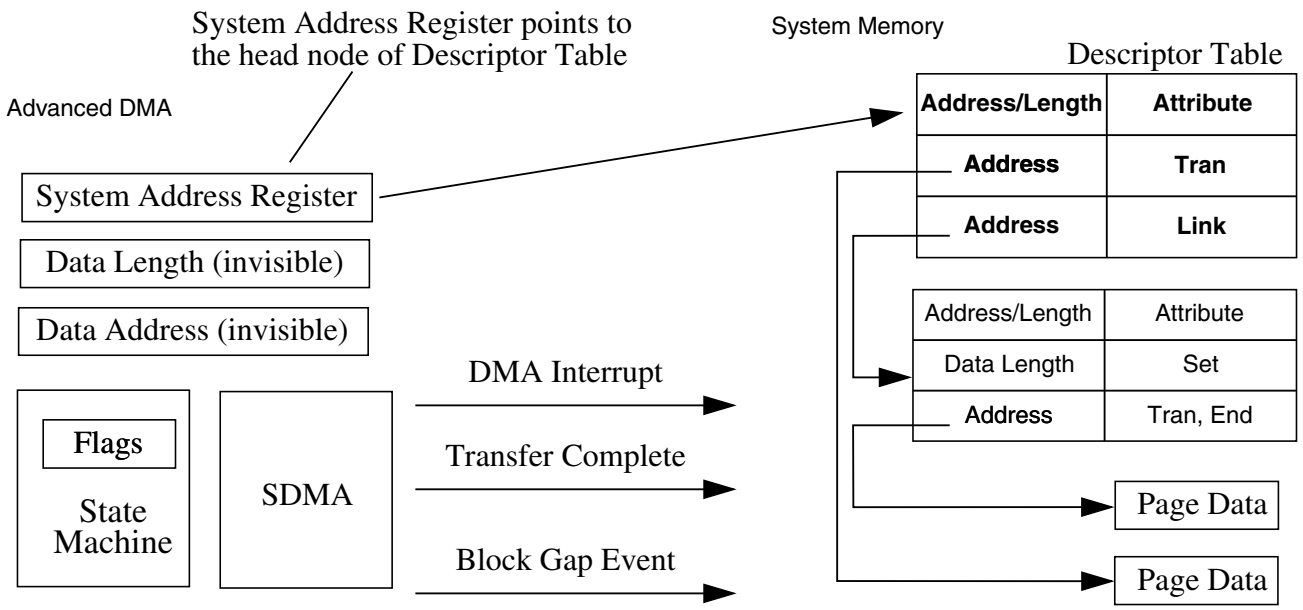



**Figure 52-32. Concept and access method of ADMA1 descriptor table**

## Table 52-36. Format of the ADMA2 descriptor table

| Address field | | Length | | Reserved | | | Attribute field | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 63 | 32 | 31 | 16 | 15 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| 32-bit address | | 16-bit length | | 0000000000 | | Act2 | Act1 | 0 | Int | End | Valid |

| Act2 | Act1 | Symbol | Comment | Operation |
|---|---|---|---|---|
| 0 | 0 | nop | No operation | Don't care |
| 0 | 1 | rsv | Reserved | Same as nop. Read this line and go to next one |
| 1 | 0 | tran | Transfer data | Transfer data with address and length set in this descriptor line |

*Table continues on the next page...*

**Table 52-36. Format of the ADMA2 descriptor table (continued)**

| | | | | |
|---|---|---|---|---|
| 1 | 1 | link | Link descriptor | Link to another descriptor |

| | |
|---|---|
| Valid | Valid = 1 indicates this line of descriptor is effective. If valid = 0 generate ADMA error interrupt and stop ADMA. |
| End | End = 1 indicates current descriptor is the ending one. |
| Int | Int = 1 generates DMA interrupt when this descriptor is done. |

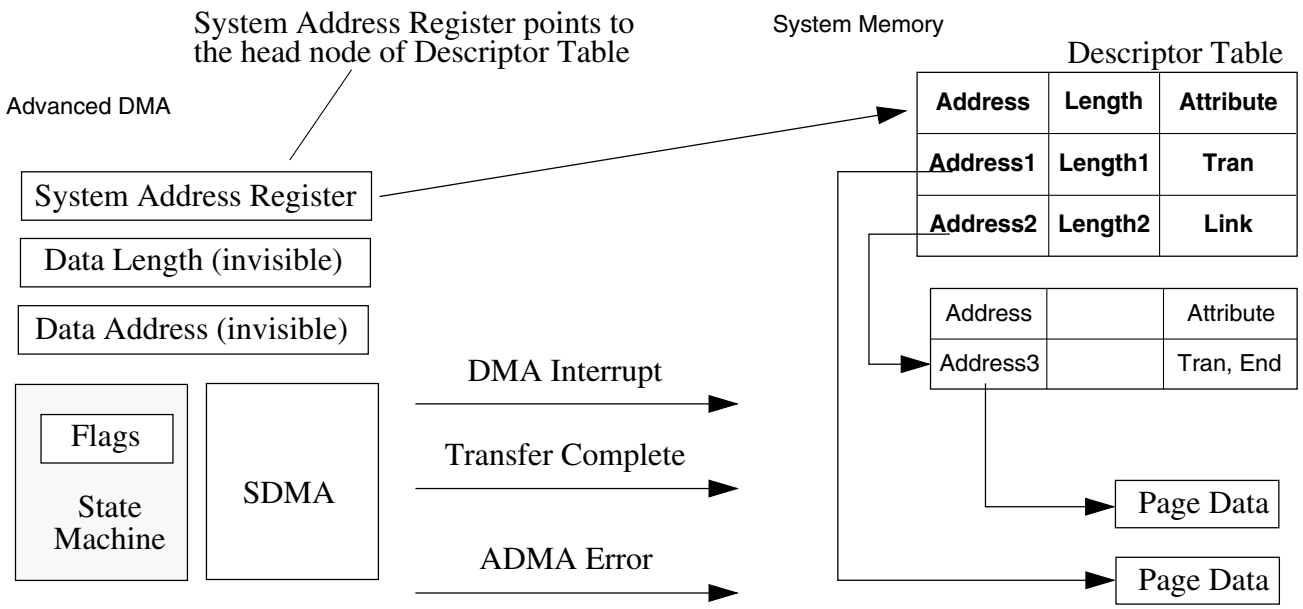



**Figure 52-33. Concept and access method of ADMA2 descriptor table**

### 52.5.2.4.2 ADMA interrupt

If the interrupt flag of descriptor is set, ADMA will generate an interrupt according to different type descriptor:

For ADMA1:

- Set type descriptor: interrupt is generated when data length is set.
- Tran type descriptor: interrupt is generated when this transfer is complete.
- Link type descriptor: interrupt is generated when new descriptor address is set.
- Nop type descriptor: interrupt is generated just after this descriptor is fetched.

For ADMA2:

- Tran type descriptor: interrupt is generated when this transfer is complete.

- Link type descriptor: interrupt is generated when new descriptor address is set.
- Nop/Rsv type descriptor: interrupt is generated just after fetch this descriptor.

#### 52.5.2.4.3 ADMA error

The ADMA stops whenever any error is encountered. These errors include:

- Fetching descriptor error
- Transfer error
- Data length mismatch error

ADMA descriptor error will be generated when it fails to detect the valid flag in the descriptor. If ADMA descriptor error occurs, the interrupt is not generated even if the interrupt flag of this descriptor is set.

When XFERTYP[BCEN] is set, data length set in buffer must be equal to the whole data length set in descriptor nodes, otherwise data length mismatch error will be generated.

If XFERTYP[BCEN] is not set, the whole data length set in descriptor must be times of block length, otherwise, when all data set in the descriptor nodes are done not at block boundary, the data mismatch error will occur.

### 52.5.3 SD protocol unit

The SD protocol unit deals with all SD protocol affairs.

The SD protocol unit performs the following functions:

- Acts as the bridge between the internal buffer and the SD bus
- Sends the command data as well as its argument serially
- Stores the serial response bit stream into the corresponding registers
- Detects the bus state on the DAT[0] line
- Monitors the interrupt from the SDIO card
- Asserts the read wait signal
- Gates off the SD clock when buffer is announcing danger status
- Detects the write protect state

The SD protocol unit consists of four submodules:

- SD transceiver
- SD clock and monitor
- Command agent
- Data agent

#### 52.5.3.1 SD transceiver

In the SD protocol unit, the transceiver is the main control module. It consists of a FSM and control module, from which the control signals for all other three modules are generated.

#### 52.5.3.2 SD clock & monitor

This module monitors the signal level on all 8 data lines, the command lines, and directly routes the level values into the register bank. The driver can use this for debug purposes.

The module also detects the card detection (CD) line as well as the DAT[3] line. The transceiver reports the card insertion state according to the CD state, or the signal level on the DAT[3] line, when PROCTL[D3CD] is set.

The module detects the write protect (WP) line. With the information of the WP state, the register bank will ignore the command, accompanied by a write operation, when the WP switch is on.

If the internal data buffer is in danger, and the SD clock must be gated off to avoid buffer over/under-run, this module will assert the gate of the output SD clock to shut the clock off. After the buffer danger has recovered, and when the system access of the buffer catches up, the clock gate of this module will open and the SD clock will be active again.

This module also drives the SDHC_LCTL output signal when PROCTL[LCTL] is set by the driver.

#### 52.5.3.3 Command agent

The command agent deals with the transactions on the CMD line. The following diagram illustrates the structure for the command CRC Shift Register.

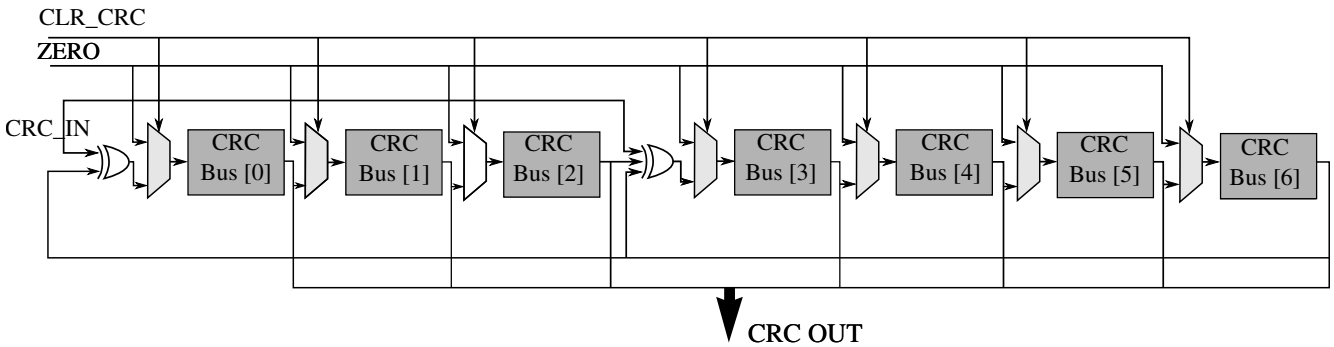



**Figure 52-34. Command CRC Shift Register**

The CRC polynomials for the CMD line are as follows:

Generator polynomial: $G(x) = x^7 + x^3 + 1$
$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + \ldots + (\text{last bit}) * x^0$
$CRC[6:0] = \text{Remainder } [(M(x) * x^7) / G(x)]$

### 52.5.3.4 Data agent

The data agent deals with the transactions on the eight data lines. Moreover, this module also detects the busy state on the DAT[0] line, and generates the read wait state by the request from the transceiver. The CRC polynomials for the DAT are as follows:

Generator polynomial: $G(x) = x^{16} + x^{12} + x^5 + 1$
$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + \ldots + (\text{last bit}) * x^0$
$CRC[15:0] = \text{Remainder } [(M(x) * x^{16}) / G(x)]$

## 52.5.4 Clock and reset manager

This module controls all the reset signals within the SDHC.

There are four kinds of reset signals within the SDHC:

- Hardware reset
- Software reset for all
- Software reset for the data part
- Software reset for the command part

All these signals are fed into this module and stable signals are generated inside the module to reset all other modules. The module also gates off all the inside signals.

There are three clocks inside the SDHC:

- Bus clock
- SDHC clock
- System clock

The module monitors the activities of all other modules, supplies the clocks for them, and when enabled, automatically gates off the corresponding clocks.

### 52.5.5 Clock generator

The clock generator generates the SDHC_CLK by peripheral source clock in two stages. The following diagram illustrates the structure of the divider. The term "base" represents the frequency of peripheral source clock.

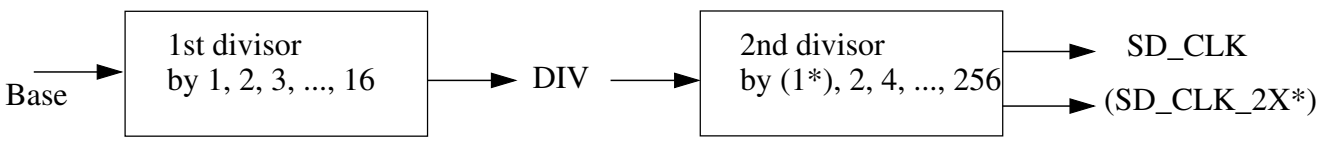



**Figure 52-35. Two stages of the clock divider**

The first stage outputs an intermediate clock (DIV), which can be base, base/2, base/3, ..., or base/16.

The second stage is a prescaler, and outputs the actual clock (SDHC_CLK). This clock is the driving clock for all submodules of the SD protocol unit, and the sync FIFOs to synchronize with the data rate from the internal data buffer. The frequency of the clock output from this stage, can be DIV, DIV/2, DIV/4,..., or DIV/256. Thus, the highest frequency of the SDHC_CLK is base, and the next highest is base/2, while the lowest frequency is base/4096. If the base clock is of equal duty ratio (usually true), the duty cycle of SDHC_CLK is also 50%, even when the compound divisor is an odd value.

### 52.5.6 SDIO card interrupt

This section discusses SDIO interrupt handling.

#### 52.5.6.1 Interrupts in 1-bit mode

In this case, the DAT[1] pin is dedicated to providing the interrupt function. An interrupt is asserted by pulling the DAT[1] low from the SDIO card, until the interrupt service is finished to clear the interrupt.

### 52.5.6.2 Interrupt in 4-bit mode

Because the interrupt and data line 1 share Pin 8 in 4-bit mode, an interrupt will be sent by the card and recognized by the host only during a specific time. This is known as the interrupt period. The SDHC will only sample the level on pin 8 during the interrupt period. At all other times, the host will ignore the level on pin 8, and treat it as the data signal. The definition of the interrupt period is different for operations with single block and multiple block data transfers.

In the case of normal single data block transmissions, the interrupt period becomes active two clock cycles after the completion of a data packet. This interrupt period lasts until after the card receives the end bit of the next command that has a data block transfer associated with it.

For multiple block data transfers in 4-bit mode, there is only a limited period of time that the interrupt period can be active due to the limited period of data line availability between the multiple blocks of data. This requires a more strict definition of the interrupt period. In this case, the interrupt period is limited to two clock cycles. This begins two clocks after the end bit of the previous data block. During this 2-clock cycle interrupt period, if an interrupt is pending, the SDHC_D1 line will be held low for one clock cycle with the last clock cycle pulling SDHC_D1 high. On completion of the Interrupt Period, the card releases the SDHC_D1 line into the high Z state. The SDHC samples the SDHC_D1] during the interrupt period when PROCTL[IABG] is set.

See the SDIO Card Specification v1.10f for further information about the SDIO card interrupt.

### 52.5.6.3 Card interrupt handling

When IRQSIGEN[CINTIEN] is set to 0, the SDHC clears the interrupt request to the host system. The host driver must clear this bit before servicing the SDIO Interrupt and must set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts.

The SDIO status bit is cleared by resetting the SDIO interrupt. Writing to this bit would have no effects. In 1-bit mode, the SDHC will detect the SDIO interrupt with or without the SD clock (to support wakeup). In 4-bit mode, the interrupt signal is sampled during the interrupt period, so there are some sample delays between the interrupt signal from the SDIO card and the interrupt to the host system interrupt controller. When the SDIO status has been set, and the host driver needs to service this interrupt, so the SDIO bit in the interrupt control register of the SDIO card will be cleared. This is required to clear the

SDIO interrupt status latched in the SDHC and to stop driving the interrupt signal to the system interrupt controller. The host driver must issue a CMD52 to clear the card interrupt. After completion of the card interrupt service, the SDIO Interrupt Enable bit is set to 1, and the SDHC starts sampling the interrupt signal again.

The following diagram illustrates the SDIO card interrupt scheme and for the sequences of software and hardware events that take place during a card interrupt handling procedure.

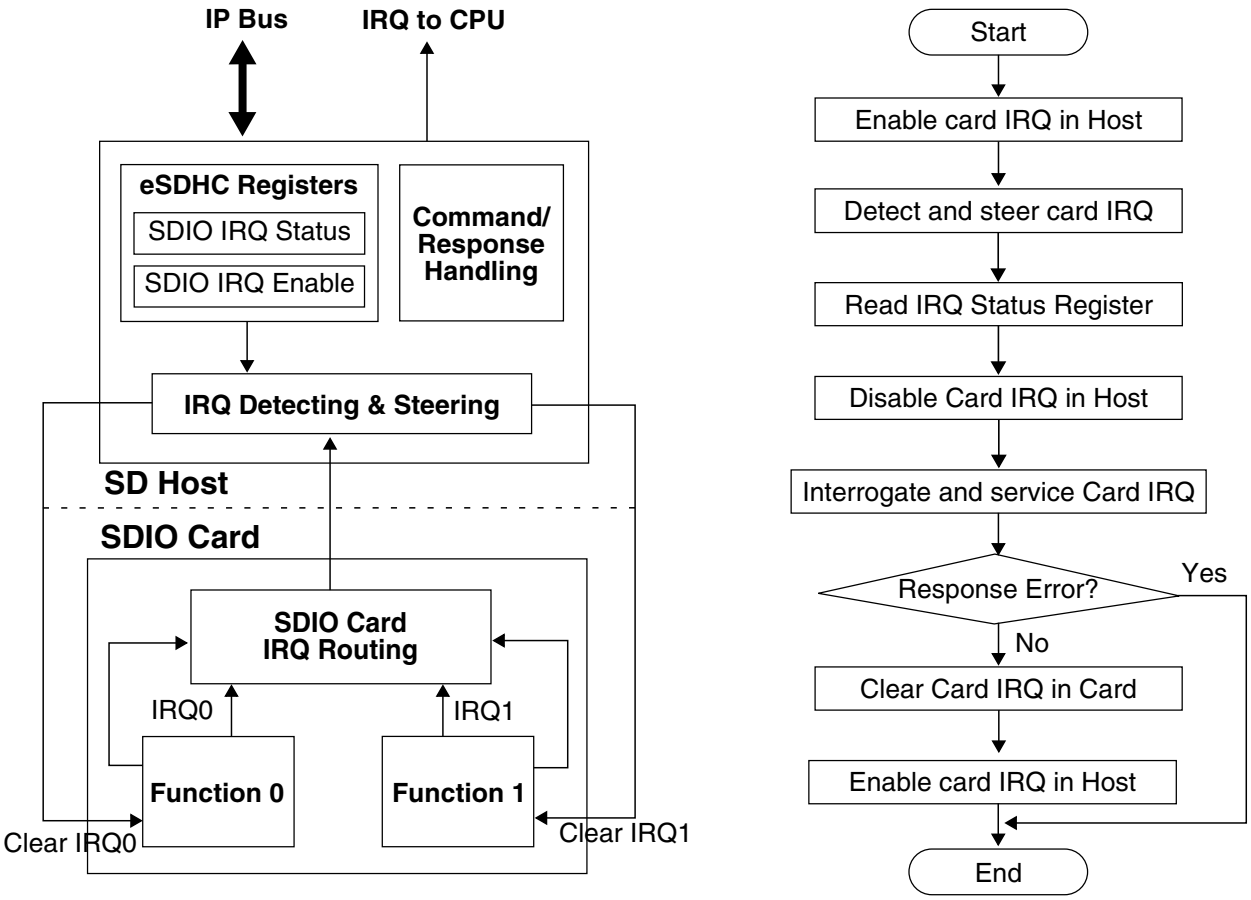



**Figure 52-36. Card interrupt scheme and card interrupt detection and handling procedure**

## 52.5.7 Card insertion and removal detection

The SDHC uses either the DAT[3] pin or the CD pin to detect card insertion or removal. When there is no card on the MMC/SD bus, the DAT[3] will be pulled to a low voltage level by default. When any card is inserted to or removed from the socket, the SDHC detects the logic value changes on the DAT[3] pin and generates an interrupt. When the DAT[3] pin is not used for card detection (for example, it is implemented in GPIO), the CD pin must be connected for card detection. Whether DAT[3] is configured for card

detection or not, the CD pin is always a reference for card detection. Whether the DAT[3] pin or the CD pin is used to detect card insertion, the SDHC will send an interrupt (if enabled) to inform the Host system that a card is inserted.

### 52.5.8 Power management and wakeup events

When there is no operation between the SDHC and the card through the SD bus, the user can completely disable the bus clock and the SDHC clock in the chip-level clock control module to save power. When the user needs to use the SDHC to communicate with the card, it can enable the clock and start the operation.

In some circumstances, when the clocks to the SDHC are disabled, for instance, when the system is in low-power mode, there are some events for which the user needs to enable the clock and handle the event. These events are called wakeup interrupts. The SDHC can generate these interrupt even when there are no clocks enabled.

The three interrupts which can be used as wakeup events are:

- Card removal interrupt
- Card insertion interrupt
- Interrupt from SDIO card

The SDHC offers a power management feature. By clearing the clock enabled bits in the system control register, the clocks are gated in the low position to the SDHC. For maximum power saving, the user can disable all the clocks to the SDHC when there is no operation in progress.

These three wakeup events, or wakeup interrupts, can also be used to wake up the system from low-power modes.

**Note**

To make the interrupt a wakeup event, when all the clocks to the SDHC are disabled or when the whole system is in low-power mode, the corresponding wakeup enabled bit needs to be set. See Protocol Control register for more information.

#### 52.5.8.1 Setting wakeup events

For the SDHC to respond to a wakeup event, the software must set the respective wakeup enable bit before the CPU enters Sleep mode. Before the software disables the host clock, it must ensure that all of the following conditions have been met:

- No read or write transfer is active
- Data and command lines are not active
- No interrupts are pending
- Internal data buffer is empty

### 52.5.9 MMC fast boot

In Embedded MultiMediaCard(eMMC4.3) spec, add fast boot feature needs hardware support.

In Boot Operation mode, the master (multimediacard host) can read boot data from the slave (MMC device) by keeping CMD line low after power-on, or sending CMD0 with argument + 0xFFFFFFFA (optional for slave), before issuing CMD1.

There are two types of Fast Boot mode, boot operation, and Alternative boot operation in eMMC4.3 spec. Each type also has with acknowledge and without acknowledge modes.

**Note**

For the eMMC4.3 card setting, please see the eMMC4.3 specification.

#### 52.5.9.1 Boot operation

**Note**

In this block guide, this fast boot is called Normal Fast Boot mode.

If the CMD line is held low for 74 clock cycles and more after power-up before the first command is issued, the slave recognizes that Boot mode is being initiated and starts preparing boot data internally.

Within 1 second after the CMD line goes low, the slave starts to send the first boot data to the master on the DAT line(s). The master must keep the CMD line low to read all of the boot data.

If boot acknowledge is enabled, the slave has to send acknowledge pattern 010 to the master within 50 ms after the CMD line goes low. If boot acknowledge is disabled, the slave will not send out acknowledge pattern '010'.

The master can terminate Boot mode with the CMD line high.

Boot operation will be terminated when all contents of the enabled boot data are sent to the master. After boot operation is executed, the slave shall be ready for CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.

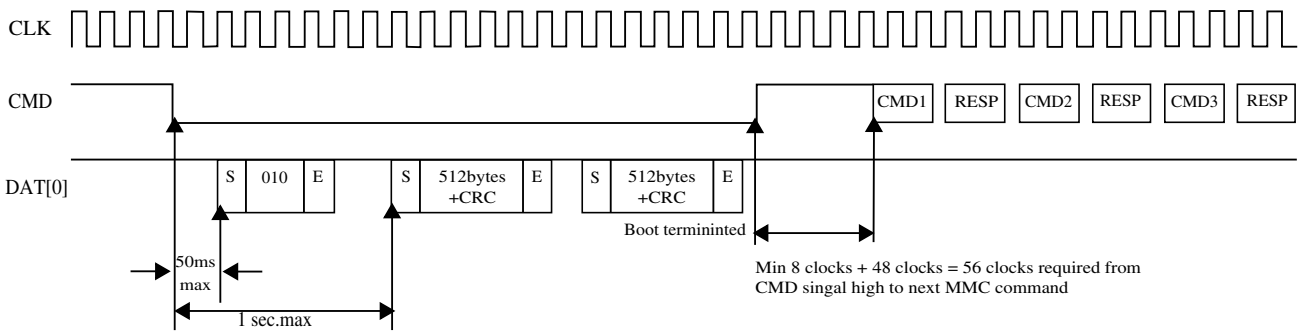



**Figure 52-37. Multimediacard state diagram in Normal Boot mode**

### 52.5.9.2 Alternative boot operation

This boot function is optional for the device. If bit 0 in the extended CSD byte[228] is set to 1, the device supports the alternative boot operation.

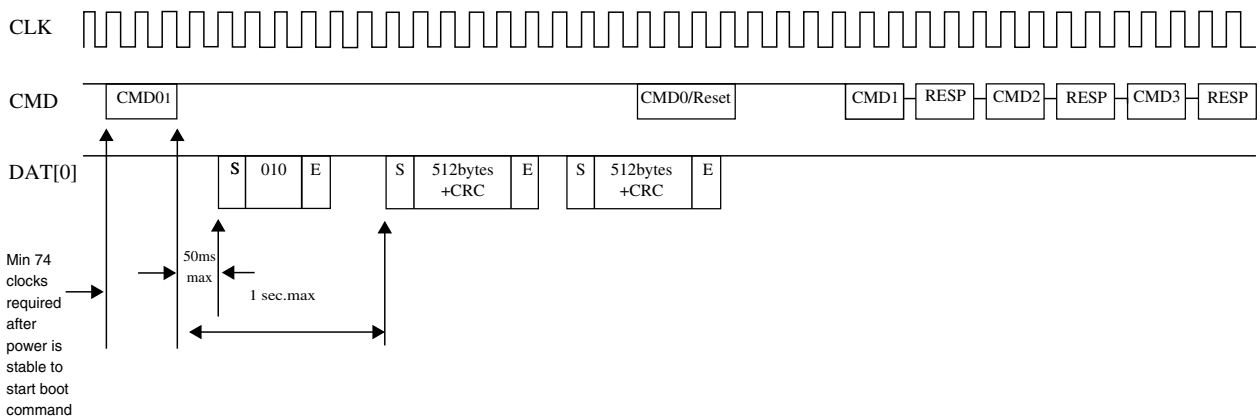
After power-up, if the host issues CMD0 with the argument of 0xFFFFFFFA after 74 clock cycles, before CMD1 is issued, or the CMD line goes low, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within 1 second after CMD0 with the argument of 0xFFFFFFFA is issued, the slave starts to send the first boot data to the master on the DAT line(s).

If boot acknowledge is enabled, the slave has to send the acknowledge pattern '010' to the master within 50ms after the CMD0 with the argument of 0xFFFFFFFA is received. If boot acknowledge is disabled, theslave will not send out acknowledge pattern '010'.

The master can terminate boot mode by issuing CMD0 (Reset).

Boot operation will be terminated when all contents of the enabled boot data are sent to the master. After boot operation is executed, the slave shall be ready for CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.




NOTE 1. CMD0 with argument 0xFFFFFFFA

**Figure 52-38. MultiMediaCard state diagram in Alternative Boot mode**

## 52.6 Initialization/application of SDHC

All communication between system and cards are controlled by the host. The host sends commands of two types:

- Broadcast
- Addressed point-to-point

Broadcast commands are intended for all cards, such as GO_IDLE_STATE, SEND_OP_COND, ALL_SEND_CID, and so on. In Broadcast mode, all cards are in the Open-Drain mode to avoid bus contention. See Commands for MMC/SD/SDIO/CE-ATA for the commands of bc and bcr categories.

After the broadcast command CMD3 is issued, the cards enter Standby mode. Addressed type commands are used from this point. In this mode, the CMD/DAT I/O pads will turn to Push-Pull mode, to have the driving capability for maximum frequency operation. See Commands for MMC/SD/SDIO/CE-ATA for the commands of ac and adtc categories.

### 52.6.1 Command send and response receive basic operation

Assuming the data type WORD is an unsigned 32-bit integer, the following flow is a guideline for sending a command to the card(s):

```
send_command(cmd_index, cmd_arg, other requirements)
{
WORD wCmd; // 32-bit integer to make up the data to write into Transfer Type register, it is
recommended to implement in a bit-field manner
wCmd = (<cmd_index> & 0x3f) >> 24; // set the first 8 bits as '00'+<cmd_index>
set CMDTYP, DPSEL, CICEN, CCCEN, RSTTYP, DTDSEL accorind to the command index;
if (internal DMA is used) wCmd |= 0x1;
```

```
if (multi-block transfer) {
 set MSBSEL bit;
 if (finite block number) {
  set BCEN bit;
  if (auto12 command is to use) set AC12EN bit;
 }
}
write_reg(CMDARG, <cmd_arg>); // configure the command argument
write_reg(XFERTYP, wCmd); // set Transfer Type register as wCmd value to issue the command
}
wait_for_response(cmd_index)
{
while (CC bit in IRQ Status register is not set); // wait until Command Complete bit is set
read IRQ Status register and check if any error bits about Command are set
if (any error bits are set) report error;
write 1 to clear CC bit and all Command Error bits;
}
```

For the sake of simplicity, the function wait_for_response is implemented here by means of polling. For an effective and formal way, the response is usually checked after the command complete interrupt is received. By doing this, make sure the corresponding interrupt status bits are enabled.

In some scenarios, the response time-out is expected. For instance, after all cards respond to CMD3 and go to the standby state, no response to the host when CMD2 is sent. The host driver shall deal with 'fake' errors like this with caution.

## 52.6.2 Card Identification mode

When a card is inserted to the socket or the card was reset by the host, the host needs to validate the operation voltage range, identify the cards, request the cards to publish the relative card address (RCA) or to set the RCA for the MMC cards. For CE-ATA, the device is connected in a point-to-point manner, and no RCA is needed. All data communications in the Card Identification mode use the command line (CMD) only. See CE-ATA Digital Protocol, Revision 1.1 for more details.

### 52.6.2.1 Card detect

The following diagram illustrates the detection of MMC, SD, and SDIO cards using the SDHC.

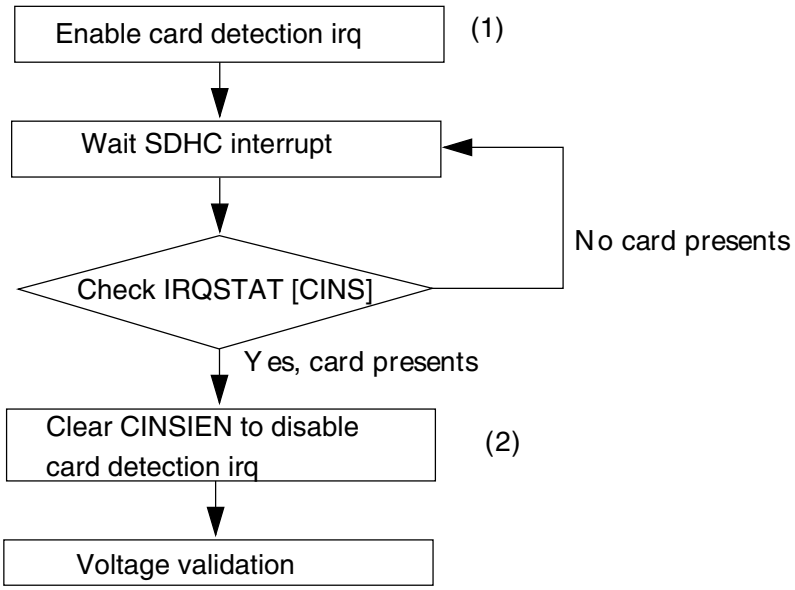



**Figure 52-39. Flow diagram for card detection**

Here is the card detection sequence:

- Set the CINSIEN bit to enable card detection interrupt
- When an interrupt from the SDHC is received, check IRQSTAT[CINS] in the Interrupt Status register to see whether it was caused by card insertion
- Clear the CINSIEN bit to disable the card detection interrupt and ignore all card insertion interrupts afterwards

To detect a CE-ATA device, after completing the normal MMC reset and initialization procedures, the host driver shall issue CMD 60 to check for a CE-ATA signature. If the device responds to the command with the CE-ATA signature, a CE-ATA device has been found. Then the driver must query EXT_CSD register byte 504 (S_CMD_SET) in the MMC register space. If the ATA bit (bit 4) is set, then the MMC device is an ATA device. If the device indicates that it is an ATA device, the Driver must set the ATA bit (bit 4) of the EXT_CSD register byte 191 (CMD_SET) to activate the ATA command set for use. To choose the command set, the driver shall issue CMD6. It is possible that the CE-ATA device does not support the ATA mode, so the driver shall not issue ATA command to the device.

### 52.6.2.2 Reset

The host consists of three types of resets:

- Hardware reset (card and host) which is driven by power on reset (POR)

- Software reset (host only) is proceed by the write operation on the SYSCTL[RSTD], SYSCTL[RSTC], or SYSCTL[RSTA] bits to reset the data part, command part, or all parts of the host controller, respectively
- Card reset (card only). The command, Go_Idle_State (CMD0), is the software reset command for all types of MMC cards, SD Memory cards, and CE-ATA cards. This command sets each card into the idle state regardless of the current card state. For an SD I/O Card, CMD52 is used to write an I/O reset in the CCCR. The cards are initialized with a default relative card address (RCA = 0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

After the card is reset, the host needs to validate the voltage range of the card. The following diagram illustrates the software flow to reset both the SDHC and the card.

For CE-ATA device that supports ATA mode, before issuing CMD0 to reset the MMC layer, two CMD39 should be issued back-to-back to the ATA control register. The first CMD39 shall have the SRST bit set to 1. The second CMD39 command shall have the SRST bit cleared to 0.

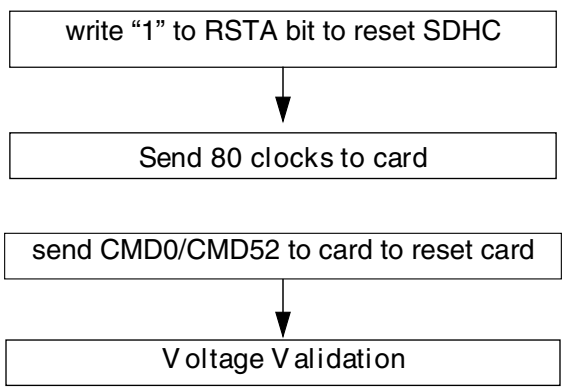



**Figure 52-40. Flow chart for reset of the SDHC and SD I/O card**

```
software_reset()
{
set_bit(SYSCTRL, RSTA); // software reset the Host
set DTOCV and SDCLKFS bit fields to get the SD_CLK of frequency around 400kHz
configure IO pad to set the power voltage of external card to around 3.0V
poll bits CIHB and CDIHB bits of PRSSTAT to wait both bits are cleared
set_bit(SYSCTRL, INTIA); // send 80 clock ticks for card to power up
send_command(CMD_GO_IDLE_STATE, <other parameters>); // reset the card with CMD0
or send_command(CMD_IO_RW_DIRECT, <other parameters>);
}
```

### 52.6.2.3 Voltage validation

All cards should be able to establish communication with the host using any operation voltage in the maximum allowed voltage range specified in the card specification. However, the supported minimum and maximum values for $V_{DD}$ are defined in the Operation Conditions Register (OCR) and may not cover the whole range. Cards that store the CID and CSD data in the preload memory are able to communicate this information only under data transfer $V_{DD}$ conditions. This means if the host and card have noncommon $V_{DD}$ ranges, the card will not be able to complete the identification cycle, nor will it be able to send CSD data.

Therefore, a special command Send_Op_Cont (CMD1 for MMC), SD_Send_Op_Cont (ACMD41 for SD Memory) and IO_Send_Op_Cont (CMD5 for SD I/O) is used. For a CE-ATA card, the process is the same as that of an MMC card. The voltage validation procedure is designed to provide a mechanism to identify and reject cards which do not match the $V_{DD}$ range(s) desired by the host. This is accomplished by the host sending the desired $V_{DD}$ voltage window as the operand of this command. Cards that can't perform the data transfer in the specified range must discard themselves from further bus operations and go into the Inactive state. By omitting the voltage range in the command, the host can query each card and determine the common voltage range before sending out-of-range cards into the inactive state. This query should be used if the host is able to select a common voltage range or if a notification shall be sent to the system when a nonusable card in the stack is detected.

The following steps show how to perform voltage validation when a card is inserted:

```
voltage_validation(voltage_range_arguement)
{
label the card as UNKNOWN;
send_command(IO_SEND_OP_COND, 0x0, <other parameters are omitted>); // CMD5, check SDIO
operation voltage, command argument is zero
if (RESP_TIMEOUT != wait_for_response(IO_SEND_OP_COND)) { // SDIO command is accepted
 if (0 < number of IO functions) {
  label the card as SDIO;
  IORDY = 0;
  while (!(IORDY in IO OCR response)) { // set voltage range for each IO function
   send_command(IO_SEND_OP_COND, <voltage range>, <other parameter>);
   wait_for_response(IO_SEND_OP_COND);
  } // end of while ...
 } // end of if (0 < ...
 if (memory part is present inside SDIO card) Label the card as SDCombo; // this is an
SD-Combo card
} // end of if (RESP_TIMEOUT ...
if (the card is labelled as SDIO card) return; // card type is identified and voltage range
is
set, so exit the function;
send_command(APP_CMD, 0x0, <other parameters are omitted>); // CMD55, Application specific
CMD
prefix
if (no error calling wait_for_response(APP_CMD, <...>) { // CMD55 is accepted
 send_command(SD_APP_OP_COND, <voltage range>, <...>); // ACMD41, to set voltage range
for memory part or SD card
 wait_for_response(SD_APP_OP_COND); // voltage range is set
 if (card type is UNKNOWN) label the card as SD;
```

```
 return; //
} // end of if (no error ...
else if (errors other than time-out occur) { // command/response pair is corrupted
 deal with it by program specific manner;
} // of else if (response time-out
else { // CMD55 is refuse, it must be MMC card or CE-ATA card
 if (card is already labelled as SDCombo) { // change label
  re-label the card as SDIO;
  ignore the error or report it;
  return; // card is identified as SDIO card
 } // of if (card is ...
 send_command(SEND_OP_COND, <voltage range>, <...>);
 if (RESP_TIMEOUT == wait_for_response(SEND_OP_COND)) { // CMD1 is not accepted, either
  label the card as UNKNOWN;
  return;
 } // of if (RESP_TIMEOUT ...
 if (check for CE-ATA signature succeeded) { // the card is CE-ATA
  store CE-ATA specific info from the signature;
  label the card as CE-ATA;
 } // of if (check for CE-ATA ...
 else label the card as MMC;
} // of else
}
```

### 52.6.2.4 Card registry

Card registry for the MMC and SD/SDIO/SD combo cards are different. For CE-ATA, it enters the tran state after reset is completed.

For the SD card, the identification process starts at a clock rate lower than 400 kHz and the power voltage higher than 2.7 V, as defined by the card specification. At this time, the CMD line output drives are push-pull drivers instead of open-drain. After the bus is activated, the host will request the card to send their valid operation conditions. The response to ACMD41 is the operation condition register of the card. The same command shall be send to all of the new cards in the system. Incompatible cards are put into the inactive state. The host then issues the command, All_Send_CID (CMD2), to each card to get its unique card identification (CID) number. Cards that are currently unidentified, in the ready state, send their CID number as the response. After the CID is sent by the card, the card goes into the Identification state.

The host then issues Send_Relative_Addr (CMD3), requesting the card to publish a new relative card address (RCA) that is shorter than the CID. This RCA will be used to address the card for future data transfer operations. After the RCA is received, the card changes its state to the Standby state. At this point, if the host wants the card to have an alternative RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card.

The host repeats the identification process with CMD2 and CMD3 for each card in the system until the last CMD2 gets no response from any of the cards in system.

For MMC operation, the host starts the card identification process in Open-Drain mode with the identification clock rate lower than 400 kHz and the power voltage higher than 2.7 V. The open drain driver stages on the CMD line allow parallel card operation during card identification. After the bus is activated, the host will request the cards to send their valid operation conditions (CMD1). The response to CMD1 is the wired OR operation on the condition restrictions of all cards in the system. Incompatible cards are sent into the Inactive state.

The host then issues the broadcast command All_Send_CID (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards, the cards in ready state, simultaneously start sending their CID numbers serially, while bit-wise monitoring their outgoing bit stream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods, stop sending their CID immediately and must wait for the next identification cycle. Because the CID is unique for each card, only one card can successfully send its full CID to the host. This card then goes into the Identification state.

Thereafter, the host issues Set_Relative_Addr (CMD3) to assign to the card a relative card address (RCA). When the RCA is received the card state changes to the standby state, and the card does not react in further identification cycles, and its output driver switches from open-drain to push-pull. The host repeats the process, mainly CMD2 and CMD3, until the host receives a time-out condition to recognize the completion of the identification process.

For CE-ATA operation (same interface as MMC cards):

```
card_registry()
{
do { // decide RCA for each card until response time-out
 if(card is labelled as SDCombo or SDIO) { // for SDIO card like device
  send_command(SET_RELATIVE_ADDR, 0x00, <...>); // ask SDIO card to publish its
RCA
  retrieve RCA from response;
 } // end if (card is labelled as SDCombo ...
 else if (card is labelled as SD) { // for SD card
  send_command(ALL_SEND_CID, <...>);
  if (RESP_TIMEOUT == wait_for_response(ALL_SEND_CID)) break;
  send_command(SET_RELATIVE_ADDR, <...>);
  retrieve RCA from response;
 } // else if (card is labelled as SD ...
 else if (card is labelled as MMC or CE-ATA) { // treat CE-ATA as MMC
  send_command(ALL_SEND_CID, <...>);
  rca = 0x1; // arbitrarily set RCA, 1 here for example, this RCA is also the
relative address to access the CE-ATA card
  send_command(SET_RELATIVE_ADDR, 0x1 << 16, <...>); // send RCA at upper 16
bits
 } // end of else if (card is labelled as MMC ...
} while (response is not time-out);
}
```

## 52.6.3 Card access

This section discusses the various card access methods.

### 52.6.3.1 Block write

This section discusses the block write access methods.

#### 52.6.3.1.1 Normal write

During a block write (CMD24 - 27, CMD60, CMD61), one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. If the CRC fails, the card shall indicate the failure on the dat line. The transferred data will be discarded and not written, and all further transmitted blocks in Multiple Block Write mode will be ignored.

If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set, and the CE-ATA card does not support partial block write, either), the card detects the block misalignment error and aborts the programming before the beginning of the first misaligned block. The card sets the ADDRESS_ERROR error bit in the status register, and while ignoring all further data transfer, waits in the Receive-Data-state for a stop command. For a CE-ATA card, check the CE-ATA card specification for its behavior in block misalignment. The write operation is also aborted if the host tries to write over a write protected area.

For MMC and SD cards, programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card will report an error and not change any register contents.

For all types of cards, some may require long and unpredictable periods of time to write a block of data. After receiving a block of data and completing the CRC check, the card will begin writing and hold the DAT line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) or other means for SDIO and CE-ATA cards at any time, and the card will respond with its status. The responded status indicates whether the card can accept new data or whether the write process is still in progress. The host may deselect the card by issuing a CMD7 to select a different card to place the card into the Standby state and release the DAT line without interrupting the

write operation. When reselecting the card, it will reactivate the busy indication by pulling DAT to low if the programming is still in progress and the write buffer is unavailable.

The software flow to write to a card incorporates the internal DMA and the write operation is a multi-block write with the Auto CMD12 enabled. For the other two methods, by means of external DMA or CPU polling status, with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to write to a card is:

1. Check the card status, wait until the card is ready for data.
2. Set the card block length/size:
    a. For SD/MMC cards, use SET_BLOCKLEN (CMD16)
    b. For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT (CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
    c. For CE-ATA cards, configure bits 1~0 in the scrControl register
3. Set the eSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the eSDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer write ready interrupt, configure the DMA settings, and enable the eSDHC DMA when sending the command with data transfer. AC12EN should also be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see whether a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

#### 52.6.3.1.2 Write with pause

The write operation can be paused during the transfer. Instead of stopping the SD_CLK at any time to pause all the operations, which is also inaccessible to the host driver, the driver can set PROCTL[SABGREQ] to pause the transfer between the data blocks. As there is no time-out condition in a write operation during the data blocks, a write to all types of cards can be paused in this way, and if the DAT0 line is not required to deassert to release the busy state, no suspend command is needed.

Like the flow described in Normal write, the write with pause is shown with the same kind of write operation:

1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:
   a. For SD/MMC, use SET_BLOCKLEN (CMD16)
   b. For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
   c. For CE-ATA cards, configure bits 1~0 in the scrControl register
3. Set the SDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the SDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer write ready interrupt, configure the DMA settings, and enable the SDHC DMA when sending the command with data transfer. The XFERTYP[AC12EN] bit should also be set.
6. Set PROCTL[SABGREQ].
7. Wait for the transfer complete interrupt.
8. Clear PROCTL[SABGREQ].
9. Check the status bit to see whether a write CRC error occurred.
10. Set PROCTL[CREQ] to continue the write operation.
11. Wait for the transfer complete interrupt.
12. Check the status bit to see whether a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

The number of blocks left during the data transfer is accessible by reading the contents of the BLKATTR[BLKCNT] . As the data transfer and the setting of the PROCTL[SABGREQ] bit are concurrent, and the delay of register read and the register setting, the actual number of blocks left may not be exactly the value read earlier. The driver shall read the value of BLKATTR[BLKCNT] after the transfer is paused and the transfer complete interrupt is received.

It is also possible that the last block has begun when the stop at block gap request is sent to the buffer. In this case, the next block gap is actually the end of the transfer. These types of requests are ignored and the Driver should treat this as a non-pause transfer and deal with it as a common write operation.

When the write operation is paused, the data transfer inside the host system is not stopped, and the transfer is active until the data buffer is full. Because of this (if not needed), it is recommended to avoid using the suspend command for the SDIO card. This is because when such a command is sent, the SDHC thinks the system will switch to another function on the SDIO card, and flush the data buffer. The SDHC takes the resume command as a normal command with data transfer, and it is left for the driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, XFERTYP[MSBSEL] and XFERTYP[BCEN] are set as well as XFERTYP[AC12EN]. However, the SDHC will automatically send a CMD12 to mark the end of the multiblock transfer.

### 52.6.3.2 Block read

This section discusses the block read access methods.

#### 52.6.3.2.1 Normal read

For block reads, the basic unit of data transfer is a block whose maximum size is stored in areas defined by the corresponding card specification. A CRC is appended to the end of each block, ensuring data transfer integrity. The CMD17, CMD18, CMD53, CMD60, CMD61, and so on, can initiate a block read. After completing the transfer, the card returns to the Transfer state. For multi blocks read, data blocks will be continuously transferred until a stop command is issued.

The software flow to read from a card incorporates the internal DMA and the read operation is a multi-block read with the Auto CMD12 enabled. For the other two methods (by means of external DMA or CPU polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to read from a card is:

1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:
    a. For SD/MMC, use SET_BLOCKLEN (CMD16)

b. For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O block size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)

c. For CE-ATA cards, configure bits 1~0 in the scrControl register

3. Set the SDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the SDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer read ready interrupt, configure the DMA settings and enable the SDHC DMA when sending the command with data transfer. XFERTYP[AC12EN] must also be set.
6. Wait for the transfer complete interrupt.
7. Check the status bit to see whether a read CRC error occurred, or some another error, occurred during the auto12 command sending and response receiving.

#### 52.6.3.2.2 Read with pause

The read operation is not generally able to pause. Only the SDIO card and SDCombo card working under I/O mode supporting the read and wait feature can pause during the read operation. If the SDIO card support read wait (SRW bit in CCCR register is 1), the Driver can set the SABGREQ bit in the Protocol Control register to pause the transfer between the data blocks. Before setting the SABGREQ bit, make sure the RWCTL bit in the Protocol Control register is set, otherwise the eSDHC will not assert the Read Wait signal during the block gap and data corruption occurs. Set the RWCTL bit after the Read Wait capability of the SDIO card is recognized.

Like the flow described in Normal read, the read with pause is shown with the same kind of read operation:

1. Check the SRW bit in the CCR register on the SDIO card to confirm the card supports Read Wait.
2. Set RWCTL.
3. Check the card status and wait until the card is ready for data.
4. Set the card block length/size:

a. For SD/MMC, use SET_BLOCKLEN (CMD16)

b. For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)

c. For CE-ATA cards, configure bits 1~0 in the scrControl register

5. Set the SDHC block length register to be the same as the block length set for the card in Step 2.
6. Set the SDHC number block register (NOB), nob is 5 (for instance).
7. Disable the buffer read ready interrupt, configure the DMA setting and enable the eSDHC DMA when sending the command with data transfer. AC12EN must also be set.
8. Set SABGREQ.
9. Wait for the Transfer Complete interrupt.
10. Clear SABGREQ.
11. Check the status bit to see whether read CRC error occurred.
12. Set CREQ to continue the read operation.
13. Wait for the Transfer Complete interrupt.
14. Check the status bit to see whether a read CRC error occurred, or some another error, occurred during the auto12 command sending and response receiving.

Like the write operation, it is possible to meet the ending block of the transfer when paused. In this case, the SDHC will ignore the Stop At Block Gap Request and treat it as a command read operation.

Unlike the write operation, there is no remaining data inside the buffer when the transfer is paused. All data received before the pause will be transferred to the Host System. Whether the Suspend Command is sent or not, the internal data buffer is not flushed.

If the Suspend Command is sent and the transfer is later resumed by means of a Resume Command, the SDHC takes the command as a normal one accompanied with data transfer. It is left for the Driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set, as well as the AC12EN bit. However, the SDHC will automatically send the CMD12 to mark the end of multi-block transfer.

### 52.6.3.3 Suspend resume

The SDHC supports the suspend resume operations of SDIO cards, although slightly different than the suggested implementation of Suspend in the SDIO card specification.

#### 52.6.3.3.1 Suspend

After setting the PROCTL[SABGREQ] bit, the host driver may send a suspend command to switch to another function of the SDIO card. The SDHC does not monitor the content of the response, therefore it doesn't know whether the suspend command succeeded or not. Accordingly, it doesn't deassert read wait for read pause. To solve this problem, the driver shall not mark the suspend command as a suspend, that is, setting the XFERTYP[CMDTYP] bits to 01. Instead, the driver shall send this command as if it were a normal command, and only when the command succeeds, and the BS bit is set in the response, can the driver send another command marked as suspend to inform the SDHC that the current transfer is suspended. As shown in the following sequence for suspend operation:

1. Set PROCTL[SABGREQ] to pause the current data transfer at block gap.
2. After IRQSTAT[BGE] is set, send the suspend command to suspend the active function. XFERTYP[CMDTYP] field must be 2'b00.
3. Check the BS bit of the CCCR in the response. If it is 1, repeat this step until the BS bit is cleared or abandon the suspend operation according to the Driver strategy.
4. Send another normal I/O command to the suspended function. The XFERTYP[CMDTYP] of this command must be 2'b01, so the SDHC can detect this special setting and be informed that the paused operation has successfully suspended. If the paused transfer is a read operation, the SDHC stops driving DAT2 and goes to the Idle state.
5. Save the context registers in the system memory for later use, including the DMA system address register for internal DMA operation, and the block attribute register.
6. Begin operation for another function on the SDIO card.

#### 52.6.3.3.2 Resume

To resume the data transfer, a resume command shall be issued:

1. To resume the suspended function, restore the context register with the saved value in step 5 of the suspend operation above.

2. Send the resume command. In the transfer type register, all fields are set to the value as if this were another ordinary data transfer, instead of a transfer resume, except the CMDTYP is set to 2'b10.
3. If the resume command has responded, the data transfer will be resumed.

### 52.6.3.4 ADMA usage

To use the ADMA in a data transfer, the host driver must prepare the correct descriptor chain prior to sending the read/write command:

1. Create a descriptor to set the data length that the current descriptor group is about to transfer. The data length should be even numbers of the block size.
2. Create another descriptor to transfer the data from the address setting in this descriptor. The data address must be at a page boundary (4 KB address aligned).
3. If necessary, create a link descriptor containing the address of the next descriptor. The descriptor group is created in steps 1–3.
4. Repeat steps 1–3 until all descriptors are created.
5. In the last descriptor, set the end flag to 1 and make sure the total length of all descriptors match the product of the block size and block number configured in the BLKATTR register.
6. Set the DSADDR register to the address of the first descriptor and set the PROCTL[DMAS] field to 01 to select the ADMA.
7. Issue a write or read command with XFERTYP[DMAEN] set to 1.

Steps 1–5 are independent of step 6, so step 6 can finish before steps 1–5. Regarding the descriptor configuration, do not to use the link descriptor, as it requires extra system memory access.

### 52.6.3.5 Transfer error

This section discusses the handling of transfer errors.

#### 52.6.3.5.1 CRC error

It is possible at the end of a block transfer that a write CRC status error or read CRC error occurs. For this type of error, the latest block received shall be discarded. This is because the integrity of the data block is not guaranteed. Discard the following data blocks and retransfer the block from the corrupted one.

For a multi-block transfer, the host driver shall issue a CMD12 to abort the current process and start the transfer by a new data command. In this scenario, even when the XFERTYP[AC12EN] and BCEND bits are set, the SDHC does not automatically send a CMD12 because the last block is not transferred. On the other hand, if it is within the last block that the CRC error occurs, an auto CMD12 will be sent by the SDHC. In this case, the driver shall re-send or re-obtain the last block with a single block transfer.

#### 52.6.3.5.2 Internal DMA error

During the data transfer with internal simple DMA, if the DMA engine encounters some error on the system bus, the DMA operation is aborted and DMA error interrupt is sent to the host system. When acknowledged by such an interrupt, the driver shall calculate the start address of data block in which the error occurs. The start address can be calculated by either:

1. Reading the DMA system address register. The error occurs during the previous burst. Taking the block size, the previous burst length and the start address of the next burst transfer into account, it is straight forward to obtain the start address of the corrupted block.
2. Reading the BLKCNT field of the block attribute register. By the number of blocks left, the total number to transfer, the start address of transfer, and the size of each block, the start address of corrupted block can be determined. When the BCEN bit is not set, the contents of the block attribute register does not change, so this method does not work.

When a DMA error occurs, abort the current transfer by means of a CMD12 (for multi block transfer), apply a reset for data, and restart the transfer from the corrupted block to recover from the error.

#### 52.6.3.5.3 ADMA error

There are three kinds of possible ADMA errors: transfer, invalid descriptor, and data-length mismatch. Whenever these errors occur, the DMA transfer stops and the corresponding error status bit is set. For acknowledging the status, the host driver should recover the error as shown below and retransfer from the place of interruption.

- Transfer error: May occur during data transfer or descriptor fetch. For either scenario, it is recommended to retrieve the transfer context, reset for the data part and re-transfer the block that was corrupted, or the next block if no block is corrupted.
- Invalid descriptor error: For such errors, retrieve the transfer context, reset for the data part, and recreate the descriptor chain from the invalid descriptor, and issue a new transfer. As the data to transfer now may be less than the previous setting, the data length configured in the new descriptor chain should match the new value.
- Data-length mismatch error: It is similar to recover from this error. The host driver polls relating registers to retrieve the transfer context, apply a reset for the data part, configure a new descriptor chain, and make another transfer if there is data left. Like the previous scenario of the invalid descriptor error, the data length must match the new transfer.

#### 52.6.3.5.4 Auto CMD12 error

After the last block of the multi-block transfer is sent or received, and XFERTYP[AC12EN] is set when the data transfer is initiated by the data command, the SDHC automatically sends a CMD12 to the card to stop the transfer. When errors with this command occur, the driver can deal with the situations in the following manner:

1. Auto CMD12 response time-out: It is not certain whether the command is accepted by the card or not. The driver should clear the IRQSTAT[AC12E] bits and resend the CMD12 until it is accepted by the card.
2. Auto CMD12 response CRC error: Because the card responds to the CMD12, the card will abort the transfer. The driver may ignore the error and clear the IRQSTAT[AC12E] bit.
3. Auto CMD12 conflict error or not sent: The command is not sent, so the driver shall send a CMD12 manually.

### 52.6.3.6 Card interrupt

The external cards can inform the host controller by means of some special signals. For the SDIO card, it can be the low level on the DAT[1] line during some special period. For the CE-ATA card, it can be a pulse on the CMD line to inform the host controller that the command and its response is finished, and it is possible that some additional external interrupt behaviors are defined. The SDHC only monitors the DAT[1] line and supports the SDIO interrupt.

When the SDIO interrupt is captured by the SDHC, and the host system is informed by the SDHC asserting the SDHC interrupt line, the interrupt service from the host driver is called.

Because the interrupt factor is controlled by the external card, the interrupt from the SDIO card must be served before IRQSTAT[CINT] is cleared by written 1. See Card interrupt handling for the card interrupt handling flow.

### 52.6.4 Switch function

MMC cards transferring data at bus widths other than 1-bit is a new feature added to the MMC specifications. The high-speed timing mode for all card devices was also recently defined in various card specifications. To enable these new features, a switch command shall be issued by the host driver.

For SDIO cards, the high-speed mode is enabled by writing the EHS bit in the CCCR register after the SHS bit is confirmed. For SD cards, the high speed mode is queried and enabled by a CMD6, with the mnemonic symbol as SWITCH_FUNC. For MMC cards (and CE-ATA over MMC interface), the high-speed mode is queried by a CMD8 and enabled by a CMD6, with the mnemonic symbol as SWITCH.

The 4-bit and 8-bit bus width of the MMC is also enabled by the SWITCH command, but with a different argument.

These new functions can also be disabled by a software reset. For SDIO cards it can be done by setting the RES bit in the CCCR register. For other cards, it can be accomplished by issuing a CMD0. This method of restoring to the normal mode is not recommended because a complete identification process is needed before the card is ready for data transfer.

For the sake of simplicity, the following flowcharts do not show current capability check, which is recommended in the function switch process.

#### 52.6.4.1 Query, enabl,e and disable SDIO high-speed mode

```
enable_sdio_high_speed_mode(void)
{
send CMD52 to query bit SHS at address 0x13;
if (SHS bit is '0') report the SDIO card does not support high speed mode and return;
send CMD52 to set bit EHS at address 0x13 and read after write to confirm EHS bit is set;
change clock divisor value or configure the system clock feeding into eSDHC to generate the
card_clk of around 50MHz;
(data transactions like normal peers)
}
disable_sdio_high_speed_mode(void)
{
```

```
send CMD52 to clear bit EHS at address 0x13 and read after write to confirm EHS bit is
cleared;
change clock divisor value or configure the system clock feeding into eSDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

### 52.6.4.2 Query, enable, and disable SD high-speed mode

```
enable_sd_high_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0xFFFFF1 and read 64 bytes of data accompanying the R1 response;
wait data transfer done bit is set;
check if the bit 401 of received 512 bit is set;
if (bit 401 is '0') report the SD card does not support high speed mode and return;
send CMD6, with argument 0x80FFFFF1 and read 64 bytes of data accompanying the R1 response;
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into eSDHC to generate the
card_clk of around 50MHz;
(data transactions like normal peers)
}
disable_sd_high_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0x80FFFFF0 and read 64 bytes of data accompanying the R1 response;
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into eSDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

### 52.6.4.3 Query, enable, and disable MMC high-speed mode

```
enable_mmc_high_speed_mode(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support high speed mode and
return;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
extract the value of CARD_TYPE field to check the 'high speed mode' in this MMC is 26MHz or
52MHz;
send CMD6 with argument 0x1B90100;
send CMD13 to wait card ready (busy line released);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 1;
if (HS_TIMING is not 1) report MMC switching to high speed mode failed and return;
change clock divisor value or configure the system clock feeding into eSDHC to generate the
card_clk of around 26MHz or 52MHz according to the CARD_TYPE;
(data transactions like normal peers)
}
disable_mmc_high_speed_mode(void)
{
send CMD6 with argument 0x2B90100;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 0;
```

```
if (HS_TIMING is not 0) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into eSDHC to generate the
card_clk of the desired value below 20MHz;
(data transactions like normal peers)
}
```

#### 52.6.4.4 Set MMC bus width

```
change_mmc_bus_width(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support multiple bit width and
return;
send CMD6 with argument 0x3B70x00; (8-bit, x=2; 4-bit, x=1; 1-bit, x=0)
send CMD13 to wait card ready (busy line released);
(data transactions like normal peers)
}
```

### 52.6.5 ADMA operation

This section presents code examples for ADMA operation.

#### 52.6.5.1 ADMA1 operation

```
Set_adma1_descriptor
{
      if (to start data transfer) {
          // Make sure the address is 4KB align.
          Set 'Set' type descriptor;
          {
              Set Act bits to 01;
              Set [31:12] bits data length (byte unit);
          }
          Set 'Tran' type descriptor;
          {
              Set Act bits to 10;
              Set [31:12] bits address (4KB align);
          }
      }
      else if (to fetch descriptor at non-continuous address) {
          Set Act bits to 11;
          Set [31:12] bits the next descriptor address (4KB align);
      }
      else { // other types of descriptor
          Set Act bits accordingly
      }
      if (this descriptor is the last one) {
          Set End bit to 1;
      }
      if (to generate interrupt for this descriptor) {
          Set Int bit to 1;
      }
      Set Valid bit to 1;
}
```

#### 52.6.5.2 ADMA2 operation

```
Set_adma2_descriptor
{
      if (to start data transfer) {
          // Make sure the address is 32-bit boundary (lower 2-bit are always '00').
          Set higher 32-bit of descriptor for this data transfer initial address;
          Set [31:16] bits data length (byte unit);
          Set Act bits to '10';
      }
      else if (to fetch descriptor at non-continuous address) {
          Set Act bits to '11';
          // Make sure the address is 32-bit boundary (lower 2-bit are always set to '00').
          Set higher 32-bit of descriptor for the next descriptor address;
      }
      else { // other types of descriptor
          Set Act bits accordingly
      }
      if (this descriptor is the last one) {
          Set 'End' bit '1';
      }
      if (to generate interrupt for this descriptor) {
          Set 'Int' bit '1';
      }
      Set the 'Valid' bit to '1';
}
```

### 52.6.6 Fast boot operation

This section discusses fast boot operations.

#### 52.6.6.1 Normal fast boot flow

1. Software needs to configure SYSCTL[INITA] to make sure 74 card clocks are finished.
2. Software needs to configure MMCBOOT[BOOTEN] to 1, MMCBOOT[BOOTMODE] to 0, and MMCBOOT[BOOTACK] to select the ack mode or not. If sending through DMA mode, software needs to configure MMCBOOT[AUTOSABGEN] to enable automatically stop at block gap feature, and MMCBOOT[DTOCVACK] to select the ack timeout value according to the sd clk frequence.
3. Software then needs to configure BLKATTR register to set block size/no.
4. Software needs to configure PROCTL[DTW].
5. Software needs to configure CMDARG to set argument if needed (no need in normal fast boot).

6. Software needs to configure XFERTYP register to start the boot process. In Normal Boot mode, XFERTYP[CMDINX], XFERTYP[CMDTYP], XFERTYP[RSPTYP], XFERTYP[CICEN], XFERTYP[CCCEN], XFERTYP[AC12EN], XFERTYP[BCEN] and XFERTYP[DMAEN] are kept default value. XFERTYP[DPSEL] bit is set to 1, XFERTYP[DTDSEL] is set to 1, XFERTYP[MSBSEL] is set to 1. Note XFERTYP[DMAEN] must be configured as 0 in polling mode. And if XFERTYP[BCEN] is configured as 1, better to configure BLKATTR[BLKSIZE] to the max value.
7. When the step 6 is configured, boot process will begin. Software needs to poll the data buffer ready status to read the data from buffer in time. If boot time-out happened (ack time out or the first data read time out), Interrupt will be triggered, and software need to configure MMCBOOT[]BOOTEN] to 0 to disable boot. Thus will make CMD high, and then after at least 56 clocks, it is ready to begin normal initialization process.
8. If no time out, software needs to decide the data read is finished and then configure MMCBOOT[]BOOTEN] to 0 to disable boot. This will make CMD line high and command completed asserted. After at least 56 clocks, it is ready to begin normal initialization process.
9. Reset the host and then can begin the normal process.

#### 52.6.6.2 Alternative fast boot flow

1. Software needs to configure init_active bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software needs to configure MMCBOOT [BOOTEN] to 1, MMCBOOT [BOOTMODE] to 1, and MMCBOOT [BOOTACK] to select the ack mode or not. If sending through DMA mode, it also needs to configure MMCBOOT [AUTOSABGEN] to enable automatically stop at block gap feature. And needs to configure MMCBOOT [DTOCVACK] to select the ack timeout value according to the sd clk frequence.
3. Software then needs to configure BLKATTR register to set block size/no.
4. Software needs to configure PROCTL[DTW].
5. Software needs to configure CMDARG register to set argument to 0xFFFFFFFA.
6. Software needs to configure XFERTYP register to start the boot process by CMD0 with 0xFFFFFFFA argument . In alternative boot, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN, and DMAEN are kept default value. DPSEL bit

is set to 1, DTDSEL is set to 1, MSBSEL is set to 1. Note DMAEN should be configured as 0 in polling mode. And if BCEN is configured as 1 in Polling mode, it is better to configure blk no in Bock Attributes Register to the max value.

7. When step 6 is configured, boot process will begin. Software needs to poll the data buffer ready status to read the data from buffer in time. If boot time out (ack data time out in 50ms or data time out in 1s), host will send out the interrupt and software need to send CMD0 with reset and then configure boot enable bit to 0 to stop this process. After command completed, configure MMCBOOT[BOOTEN] to 0 to disable boot. After at least 8 clocks from command completed, card is ready for identification step.
8. If no time out, software needs to decide when to stop the boot process, and send out the CMD0 with reset and then after command completed, configure MMCBOOT[BOOTEN] to stop the process. After 8 clocks from command completed, slave(card) is ready for identification step.
9. Reset the host and then begin the normal process.

### 52.6.6.3 Fast boot application case in DMA mode

In the boot application case, because the image destination and the image size are contained in the beginning of the image, switching DMA parameters on the fly during MMC fast boot is required.

In fast boot, host can use ADMA2 (advanced DMA2) with two destinations.

The detail flow:

1. Software needs to configure init_active bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software needs to configure MMCBOOT[BOOTEN] to 1; and MMCBOOT[BOOTMODE] to 0 (normal fast boot), to 1(alternative boot); and MMCBOOT[BOOTACK] to select the ack mode or not. In DMA mode, configure MMCBOOT[AUTOSABGEN] to 1 for enable automatically stop at block gap feature. Also configure MMCBOOT[BOOTBLKCNT] to set the VAULE1 (value of block count that need to trans first time), so that host will stop at block gap when card block counter is equal to this value. And it needs to configure MMCBOOT[DTOCVACK] to select the ack timeout value according to the sd clk frequence.

3. Software then needs to configure BLKATTR register to set block size/no. In DMA mode, it is better to set block number to the max value(16'hffff).
4. Software needs to configure PROCTL[DTW].
5. Software enables ADMA2 by configuring PROCTL[DMAS].
6. Software need to set at least three pairs ADMA2 descriptor in boot memory (that is, in IRAM, at least 6 words). The first pair descriptor define the start address (IRAM) and data length(512byte*VALUE1) of first part boot code. Software also need to set the second pair descriptor, the second start address (any value that is writable), data length is suggest to set 1~2 word (record as VAULE2). Note: the second couple desc also transfer useful data even at lease 1 word. Because our ADMA2 can't support 0 data_length data transfer descriptor.
7. Software needs to configure CMDARG register to set argument to 0xFFFFFFFA in alternative fast boot, and doesn't need to be set in normal fast boot.
8. Software needs to configure XFERTYP register to start the boot process . XFERTYP[CMDINX], XFERTYP[CMDTYP], XFERTYP[RSPTYP], XFERTYP[CICEN], XFERTYP[CCCEN], XFERTYP[AC12EN], XFERTYP[BCEN], and XFERTYP[DMAEN] are kept at default value. XFERTYP[DPSEL] bit is set to 1, XFERTYP[DTDSEL] is set to 1, XFERTYP[MSBSEL] is set to 1. XFERTYP[DMAEN] is configured as 1 in DMA mode. And if XFERTYP[BCEN] is configured as 1, better to configure blk no in BLKATTR register to the max value.
9. When step 8 is configured, boot process will begin, the first VAULE1 block number data has transfer. Software needs to poll IRQSTAT[TC] bit to determine first transfer is end. Also software needs to poll IRQSTAT[BGE] bit to determine if first transfer stop at block gap.
10. When IRQSTAT[TC] and IRQSTAT[BGE] bits are 1, . SW can analyzes the first code of VAULE1 block, initializes the new memory device, if required, and sets the third pair of descriptors to define the start address and length of the remaining part of boot code (VAULE3 the remain boot code block). Remember set the last descriptor with END.
11. Software needs to configure MMCBOOT register (offset 0xc4) again. Set MMCBOOT[BOOTEN] bit to 1; and MMCBOOT[BOOTMODE] bit to 0 (normal fast boot), to 1 (alternative boot); and MMCBOOT[BOOTACK] bit to select the ack mode or not. In DMA mode, configure MMCBOOT[AUTOSABGEN] bit to 1 for enable automatically stop at block gap feature. Also configure MMCBOOT[BOOTBLKCNT] bit to set the (VAULE1+1+VAULE3), so that host

will stop at block gap when card block counter is equal to this value. And need to configure MMCBOOT[DTOCVACK] bit to select the ack timeout value according to the sd clk frequence.

12. Software needs to clear IRQSTAT[TC] and IRQSTAT[BGE] bit. And software need to clear PROCTL[SABGREQ], and set PROCTL[CREQ] to 1 to resume the data transfer. Host will transfer the VALUE2 and VAULE3 data to the destination that is set by descriptor.
13. Software needs to poll IRQSTAT[BGE] bit to determine if the fast boot is over.

**Note**

1. When ADMA boot flow is started, for SDHC, it is like a normal ADMA read operation. So setting ADMA2 descriptor as the normal ADMA2 transfer.
2. Need a few words length memory to keep descriptor.
3. For the 1~2 words data in second descriptor setting, it is the useful data, so software need to deal the data due to the application case.

## 52.6.7 Commands for MMC/SD/SDIO/CE-ATA

The following table lists the commands for the MMC/SD/SDIO/CE-ATA cards.

See the corresponding specifications for more details about the command information.

There are four kinds of commands defined to control the Multimediacard:

- Broadcast commands (bc), no response
- Broadcast commands with response (bcr), response from all cards simultaneously
- Addressed (point-to-point) commands (ac), no data transfer on the DAT
- Addressed (point-to-point) data transfer commands (adtc)

**Table 52-37. Commands for MMC/SD/SDIO/CE-ATA cards**

| CMD INDEX | Type | Argument | Resp | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD0 | bc | [31:0] stuff bits | - | GO_IDLE_STATE | Resets all MMC and SD memory cards to idle state. |

*Table continues on the next page...*

## Table 52-37. Commands for MMC/SD/SDIO/CE-ATA cards (continued)

| CMD INDEX | Type | Argument | Resp | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD1 | bcr | [31:0] OCR without busy | R3 | SEND_OP_COND | Asks all MMC and SD memory cards in idle state to send their operation conditions register contents in the response on the CMD line. |
| CMD2 | bcr | [31:0] stuff bits | R2 | ALL_SEND_CID | Asks all cards to send their CID numbers on the CMD line. |
| CMD3[1] | ac | [31:6] RCA<br>[15:0] stuff bits | R1<br>R6 (SDIO) | SET/ SEND_RELATIVE_ADDR | Assigns relative address to the card. |
| CMD4 | bc | [31:0] DSR<br>[15:0] stuff bits | - | SET_DSR | Programs the DSR of all cards. |
| CMD5 | bc | [31:0] OCR without busy | R4 | IO_SEND_OP_COND | Asks all SDIO cards in idle state to send their operation conditions register contents in the response on the CMD line. |
| CMD6[2] | adtc | [31] Mode<br>0: Check function<br>1: Switch function [30:8] Reserved for function groups 6 ~ 3 (All 0 or 0xFFFF)<br>[7:4] Function group1 for command system<br>[3:0] Function group2 for access mode | R1 | SWITCH_FUNC | Checks switch ability (mode 0) and switch card function (mode 1). Refer to "SD Physical Specification V1.1" for more details. |
| CMD6[3] | ac | [31:26] Set to 0<br>[25:24] Access<br>[23:16] Index<br>[15:8] Value<br>[7:3] Set to 0<br>[2:0] Cmd Set | R1b | SWITCH | Switches the mode of operation of the selected card or modifies the EXT_CSD registers. Refer to "The MultiMediaCard System Specification Version 4.0 Final draft 2" for more details. |
| CMD7 | ac | [31:6] RCA<br>[15:0] stuff bits | R1b | SELECT/ DESELECT_CARD | Toggles a card between the stand-by and transfer states or between the programming and disconnect states. In both cases, the card is selected by its own relative address and gets deselected by any other address. Address 0 deselects all. |
| CMD8 | adtc | [31:0] stuff bits | R1 | SEND_EXT_CSD | The card sends its EXT_CSD register as a block of data, with a block size of 512 bytes. |

*Table continues on the next page...*

## Table 52-37. Commands for MMC/SD/SDIO/CE-ATA cards (continued)

| CMD INDEX | Type | Argument | Resp | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD9 | ac | [31:6] RCA<br>[15:0] stuff bits | R2 | SEND_CSD | Addressed card sends its card-specific data (CSD) on the CMD line. |
| CMD10 | ac | [31:6] RCA<br>[15:0] stuff bits | R2 | SEND_CID | Addressed card sends its card-identification (CID) on the CMD line. |
| CMD11 | adtc | [31:0] data address | R1 | READ_DAT_UNTIL_STOP | Reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows. |
| CMD12 | ac | [31:0] stuff bits | R1b | STOP_TRANSMISSION | Forces the card to stop transmission. |
| CMD13 | ac | [31:6] RCA<br>[15:0] stuff bits | R1 | SEND_STATUS | Addressed card sends its status register. |
| CMD14 | Reserved | | | | |
| CMD15 | ac | [31:6] RCA<br>[15:0] stuff bits | - | GO_INACTIVE_STATE | Sets the card to inactive state to protect the card stack against communication breakdowns. |
| CMD16 | ac | [31:0] block length | R1 | SET_BLOCKLEN | Sets the block length (in bytes) for all following block commands (read and write). Default block length is specified in the CSD. |
| CMD17 | adtc | [31:0] data address | R1 | READ_SINGLE_BLOCK | Reads a block of the size selected by the SET_BLOCKLEN command. |
| CMD18 | adtc | [31:0] data address | R1 | READ_MULTIPLE_BLOCK | Continuously transfers data blocks from card to host until interrupted by a stop command. |
| CMD19 | Reserved | | | | |
| CMD20 | adtc | [31:0] data address | R1 | WRITE_DAT_UNTIL_STOP | Writes data stream from the host, starting at the given address, until a STOP_TRANSMISION follows. |
| CMD21-23 | Reserved | | | | |
| CMD24 | adtc | [31:0] data address | R1 | WRITE_BLOCK | Writes a block of the size selected by the SET_BLOCKLEN command. |
| CMD25 | adtc | [31:0] data address | R1 | WRITE_MULTIPLE_BLOCK | Continuously writes blocks of data until a STOP_TRANSMISSION follows. |

*Table continues on the next page...*

## Table 52-37. Commands for MMC/SD/SDIO/CE-ATA cards (continued)

| CMD INDEX | Type | Argument | Resp | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD26 | adtc | [31:0] stuff bits | R1 | PROGRAM_CID | Programming of the card identification register. This command shall be issued only once per card. The card contains hardware to prevent this operation after the first programming. Normally this command is reserved for the manufacturer. |
| CMD27 | adtc | [31:0] stuff bits | R1 | PROGRAM_CSD | Programming of the programmable bits of the CSD. |
| CMD28 | ac | [31:0] data address | R1b | SET_WRITE_PROT | If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE). |
| CMD29 | ac | [31:0] data address | R1b | CLR_WRITE_PROT | If the card provides write protection features, this command clears the write protection bit of the addressed group. |
| CMD30 | adtc | [31:0] write protect data address | R1 | SEND_WRITE_PROT | If the card provides write protection features, this command asks the card to send the status of the write protection bits. |
| CMD31 | Reserved | | | | |
| CMD32 | ac | [31:0] data address | R1 | TAG_SECTOR_START | Sets the address of the first sector of the erase group. |
| CMD33 | ac | [31:0] data address | R1 | TAG_SECTOR_END | Sets the address of the last sector in a continuous range within the selection of a single sector to be selected for erase. |
| CMD34 | ac | [31:0] data address | R1 | UNTAG_SECTOR | Removes one previously selected sector from the erase selection. |
| CMD35 | ac | [31:0] data address | R1 | TAG_ERASE_GROUP_START | Sets the address of the first erase group within a range to be selected for erase. |
| CMD36 | ac | [31:0] data address | R1 | TAG_ERASE_GROUP_END | Sets the address of the last erase group within a continuous range to be selected for erase. |
| CMD37 | ac | [31:0] data address | R1 | UNTAG_ERASE_GROUP | Removes one previously selected erase group from the erase selection. |

*Table continues on the next page...*

## Table 52-37. Commands for MMC/SD/SDIO/CE-ATA cards (continued)

| CMD INDEX | Type | Argument | Resp | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD38 | ac | [31:0] stuff bits | R1b | ERASE | Erase all previously selected sectors. |
| CMD39 | ac | [31:0] RCA<br>[15] register write flag<br>[14:8] register address<br>[7:0] register data | R4 | FAST_IO | Used to write and read 8-bit (register) data fields. The command addresses a card, and a register, and provides the data for writing if the write flag is set. The R4 response contains data read from the address register. This command accesses application dependent registers which are not defined in the MMC standard. |
| CMD40 | bcr | [31:0] stuff bits | R5 | GO_IRQ_STATE | Sets the system into interrupt mode. |
| CMD41 | Reserved | | | | |
| CDM42 | adtc | [31:0] stuff bits | R1b | LOCK_UNLOCK | Used to set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command. |
| CMD43~51 | Reserved | | | | |
| CMD52 | ac | [31:0] stuff bits | R5 | IO_RW_DIRECT | Access a single register within the total 128k of register space in any I/O function. |
| CMD53 | ac | [31:0] stuff bits | R5 | IO_RW_EXTENDED | Accesses a multiple I/O register with a single command. Allows the reading or writing of a large number of I/O registers. |
| CMD54 | Reserved | | | | |
| CMD55 | ac | [31:16] RCA<br>[15:0] stuff bits | R1 | APP_CMD | Indicates to the card that the next command is an application specific command rather that a standard command. |
| CMD56 | adtc | [31:1] stuff bits<br>[0]: RD/WR | R1b | GEN_CMD | Used either to transfer a data block to the card or to get a data block from the card for general purpose / application specific commands. The size of the data block is set by the SET_BLOCK_LEN command. |
| CMD57~59 | Reserved | | | | |

*Table continues on the next page...*

## Table 52-37. Commands for MMC/SD/SDIO/CE-ATA cards (continued)

| CMD INDEX | Type | Argument | Resp | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD60 | adtc | [31] WR<br>[30:24] stuff bits<br>[23:16] address<br>[15:8] stuff bits<br>[7:0] byte count | R1b | RW_MULTIPLE_REGISTER | CE-ATA devices contain a set of Status and Control registers that begin at register offset 80h.These registers are used to control the behavior of the device and to retrieve status information regarding the operation of the device. All Status and Control registers are WORD (32-bit) in size and are WORD aligned. CMD60 shall be used to read and write these registers. |
| CMD61 | adtc | [31] WR<br>[30:16] stuff bits<br>[15:0] data unit count | R1b | RW_MULTIPLE_BLOCK | The host issues a RW_MULTIPLE_BLOCK (CMD61) to begin the data transfer for the ATA command. |
| CMD62~63 | Reserved | | | | |
| ACMD6[4] | ac | [31:2] stuff bits [1:0] bus width | R1 | SET_BUS_WIDTH | Defines the data bus width ('00'=1bit or '10'=4bit bus) to be used for data transfer. The allowed data bus widths are given in SCR register. |
| ACMD13[5] | adtc | [31:0] stuff bits | R1 | SD_STATUS | Send the SD memory card status. |
| ACMD22[6] | adtc | [31:0] stuff bits | R1 | SEND_NUM_WR_SECTORS | Send the number of the written sectors (without errors). Responds with 32-bit plus the CRC data block. |
| ACMD23[7] | ac | | R1 | SET_WR_BLK_ERASE_COUNT | - |
| ACMD41[8] | bcr | [31:0] OCR | R3 | SD_APP_OP_COND | Asks the accessed card to send its operating condition register (OCR) contents in the response on the CMD line. |
| ACMD42[9] | ac | | R1 | SET_CLR_CARD_DETECT | - |
| ACMD51[10] | adtc | [31:0] stuff bits | R1 | SEND_SCR | Reads the SD Configuration Register (SCR). |

1. CMD3 differs for MMC and SD cards. For MMC cards, it is referred to as SET_RELATIVE_ADDR, with a response type of R1. For SD cards, it is referred to as SEND_RELATIVE_ADDR, with a response type of R6 (with RCA inside).
2. CMD6 differs completely between high speed MMC cards and high speed SD cards. Command SWITCH_FUNC is for high speed SD cards.
3. Command SWITCH is for high speed MMC cards as well as for CE-ATA cards over the MMC interface. The Index field can contain any value from 0-255, but only values 0-191 are valid. If the Index value is in the 192-255 range the card does not perform any modification and the SWITCH_ERROR status bit in the EXT_CSD register is set. The Access Bits are shown in Table 52-38.
4. ACMDs shall be preceded with the APP_CMD command. (Commands listed are used for SD only, other SD commands not listed are not supported on this module).

5. ACMDs shall be preceded with the APP_CMD command. (Commands listed are used for SD only, other SD commands not listed are not supported on this module).
6. ACMDs shall be preceded with the APP_CMD command. (Commands listed are used for SD only, other SD commands not listed are not supported on this module).
7. ACMDs shall be preceded with the APP_CMD command. (Commands listed are used for SD only, other SD commands not listed are not supported on this module).
8. ACMDs shall be preceded with the APP_CMD command. (Commands listed are used for SD only, other SD commands not listed are not supported on this module).
9. ACMDs shall be preceded with the APP_CMD command. (Commands listed are used for SD only, other SD commands not listed are not supported on this module).
10. ACMDs shall be preceded with the APP_CMD command. (Commands listed are used for SD only, other SD commands not listed are not supported on this module).

The Access Bits for the EXT_CSD Access Modes are shown in the following table.

**Table 52-38. EXT_CSD Access Modes**

| Bits | Access Name | Operation |
|---|---|---|
| 00 | Command Set | The command set is changed according to the Cmd Set field of the argument |
| 01 | Set Bits | The bits in the pointed byte are set, according to the 1 bits in the Value field. |
| 10 | Clear Bits | The bits in the pointed byte are cleared, according to the 1 bits in the Value field. |
| 11 | Write Byte | The Value field is written into the pointed byte. |

## 52.7 Software restrictions

Software for the SDHC must observe the following restrictions.

### 52.7.1 Initialization active

The driver cannot set SYSCTL[INITA] when any of the command line or data lines is active, so the driver must ensure both PRSSTAT[CDIHB] and PRSSTAT[CIHB] bits are cleared. To auto clear SYSCTL[INITA], SYSCTL[SDCLKEN] must be 1, otherwise no clocks can go out to the card and SYSCTL[INITA] will never clear.

### 52.7.2 Software polling procedure

For polling read or write, when the software begins a buffer read or write, it must access exactly the number of times as the values set in the watermark level register. Moreover, if the block size is not the times of the value in watermark level register, read and write respectively, the software must access exactly the remained number of words at the end of each block.

For example, for read operation, if the WML[RDWML] is 4, indicating the watermark level is 16 bytes, block size is 40 bytes, and the block number is 2, then the access times for the burst sequence in the whole transfer process must be 4, 4, 2, 4, 4, 2.

### 52.7.3 Suspend operation

To suspend the data transfer, the software must inform SDHC that the suspend command is successfully accepted. To achieve this, after the suspend command is accepted by the SDIO card, software must send another normal command marked as suspend command (XFERTYP[CMDTYP] bits set as 01) to inform SDHC that the transfer is suspended.

If software needs resume the suspended transfer, it should read the value in BLKATTR[BLKCNT] to save the remained number of blocks before sending the normal command marked as suspend, otherwise on sending such suspend command, SDHC will regard the current transfer as aborted and change BLKATTR[BLKCNT] to its original value, instead of keeping the remained number of blocks.

### 52.7.4 Data length setting

For either ADMA (ADMA1 or ADMA2) transfer, the data in the data buffer must be word aligned, so the data length set in the descriptor must be times of 4.

### 52.7.5 (A)DMA address setting

To configure ADMA1/ADMA2/DMA address register, when TC[IRQSTAT] is set, the register will always update itself with the internal address value to support dynamic address synchronization, so software must ensure that TC[IRQSTAT] is cleared prior to configuring ADMA1/ADMA2/DMA address register.

### 52.7.6 Data port access

Data port does not support parallel access. For example, during an external DMA access, it is not allowed to write any data to the data port by CPU; or during a CPU read operation, it is also prohibited to write any data to the data port, by either CPU or external DMA. Otherwise the data would be corrupted inside the SDHC buffer.

### 52.7.7 Change clock frequency

SDHC does not automatically gates off the card clock when the host driver changes the clock frequency. To remove possible glitch on the card clock, clear SYSCTL[SDCLKEN] when changing clock divisor value and set SYSCTL[SDCLKEN] to 1 after PRSSTAT[SDSTB] is 1 again.

### 52.7.8 Multi-block read

For predefined multi-block read operation, that is, the number of blocks to read has been defined by previous CMD23 for MMC, or predefined number of blocks in CMD53 for SDIO/SDCombo, or whatever multi-block read without abort command at card side, an abort command, either automatic or manual CMD12/CMD52, is still required by SDHC after the pre-defined number of blocks are done, to drive the internal start response timeout. Manually sending an abort command with XFERTYP[RSPTYP] both bits cleared is recommended.

# Chapter 53
# Integrated Interchip Sound (I2S) / Synchronous Audio Interface (SAI)

## 53.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The $I^2S$ (or I2S) module provides a synchronous audio interface (SAI) that supports full-duplex serial interfaces with frame synchronization such as $I^2S$, AC97, TDM, and codec/DSP interfaces.

### 53.1.1 Features

- Transmitter with independent bit clock and frame sync supporting 2 data channels
- Receiver with independent bit clock and frame sync supporting 2 data channels
- Maximum Frame Size of 32 words
- Word size of between 8-bits and 32-bits
- Word size configured separately for first word and remaining words in frame
- Asynchronous 8 × 32-bit FIFO for each transmit and receive channel
- Supports graceful restart after FIFO error

### 53.1.2 Block diagram

The following block diagram also shows the module clocks.

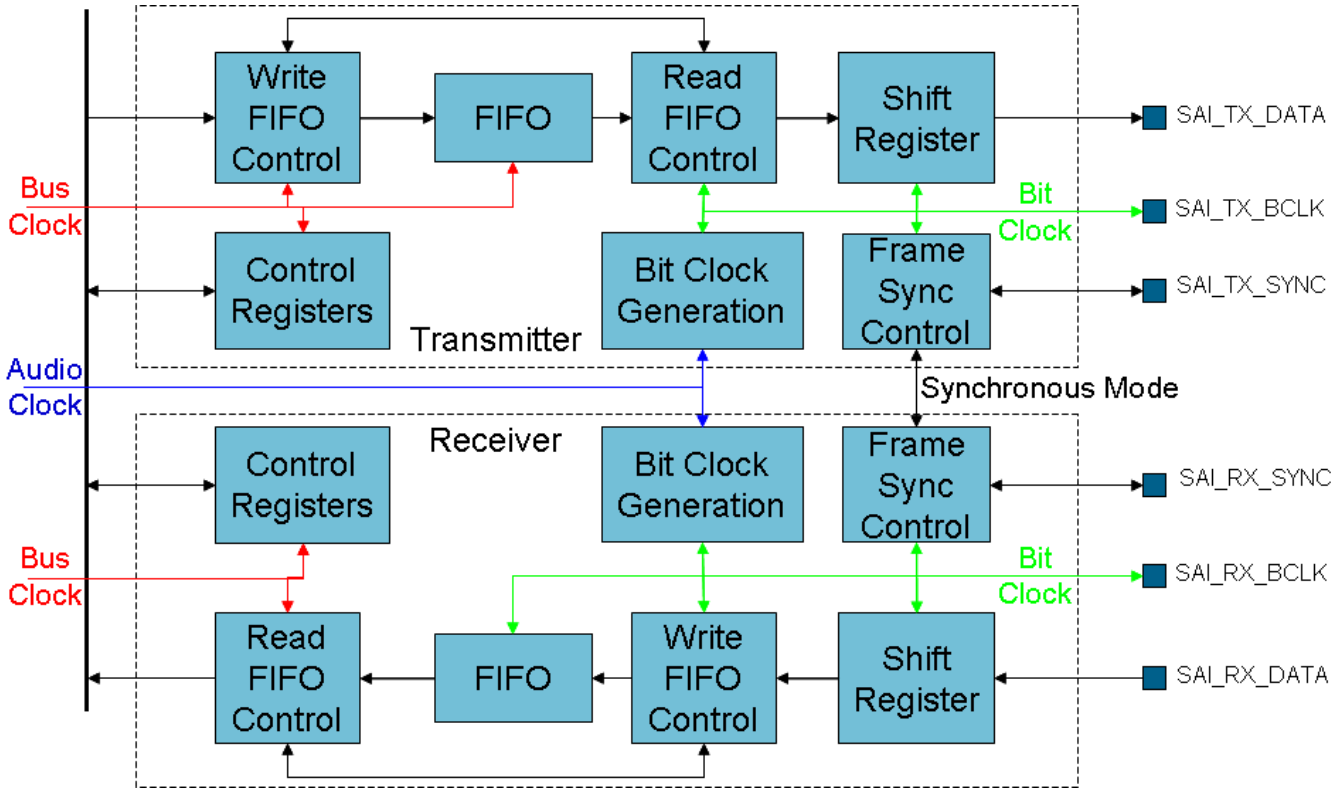



**Figure 53-1. I²S/SAI block diagram**

## 53.1.3 Modes of operation

The module operates in these MCU power modes: Run mode, stop modes, low-leakage modes, and Debug mode.

### 53.1.3.1 Run mode

In Run mode, the SAI transmitter and receiver operate normally.

### 53.1.3.2 Stop modes

In Stop mode, the SAI transmitter and/or receiver can continue operating provided the appropriate Stop Enable bit is set (TCSR[STOPE] and/or RCSR[STOPE], respectively), and provided the transmitter and/or receiver is/are using an externally generated bit clock or an Audio Master Clock that remains operating in Stop mode. The SAI transmitter and/or receiver can generate an asynchronous interrupt to wake the CPU from Stop mode.

In Stop mode, if the Transmitter Stop Enable (TCSR[STOPE]) bit is clear, the transmitter is disabled after completing the current transmit frame, and, if the Receiver Stop Enable (RCSR[STOPE]) bit is clear, the receiver is disabled after completing the current receive frame. Entry into Stop mode is prevented–not acknowledged–while waiting for the transmitter and receiver to be disabled at the end of the current frame.

#### 53.1.3.3 Low-leakage modes

When entering low-leakage modes, the Stop Enable (TCSR[STOPE] and RCSR[STOPE]) bits are ignored and the SAI is disabled after completing the current transmit and receive Frames. Entry into stop mode is prevented (not acknowledged) while waiting for the transmitter and receiver to be disabled at the end of the current frame.

#### 53.1.3.4 Debug mode

In Debug mode, the SAI transmitter and/or receiver can continue operating provided the Debug Enable bit is set. When TCSR[DBGE] or RCSR[DBGE] bit is clear and Debug mode is entered, the SAI is disabled after completing the current transmit or receive frame. The transmitter and receiver bit clocks are not affected by Debug mode.

## 53.2 External signals

| Name | Function | I/O | Reset | Pull |
|---|---|---|---|---|
| SAI_TX_BCLK | Transmit Bit Clock | I/O | 0 | — |
| SAI_TX_SYNC | Transmit Frame Sync | I/O | 0 | — |
| SAI_TX_DATA[1:0] | Transmit Data | O | 0 | — |
| SAI_RX_BCLK | Receive Bit Clock | I/O | 0 | — |
| SAI_RX_SYNC | Receive Frame Sync | I/O | 0 | — |
| SAI_RX_DATA[1:0] | Receive Data | I | 0 | — |
| SAI_MCLK | Audio Master Clock | I/O | 0 | — |

## 53.3 Memory map and register definition

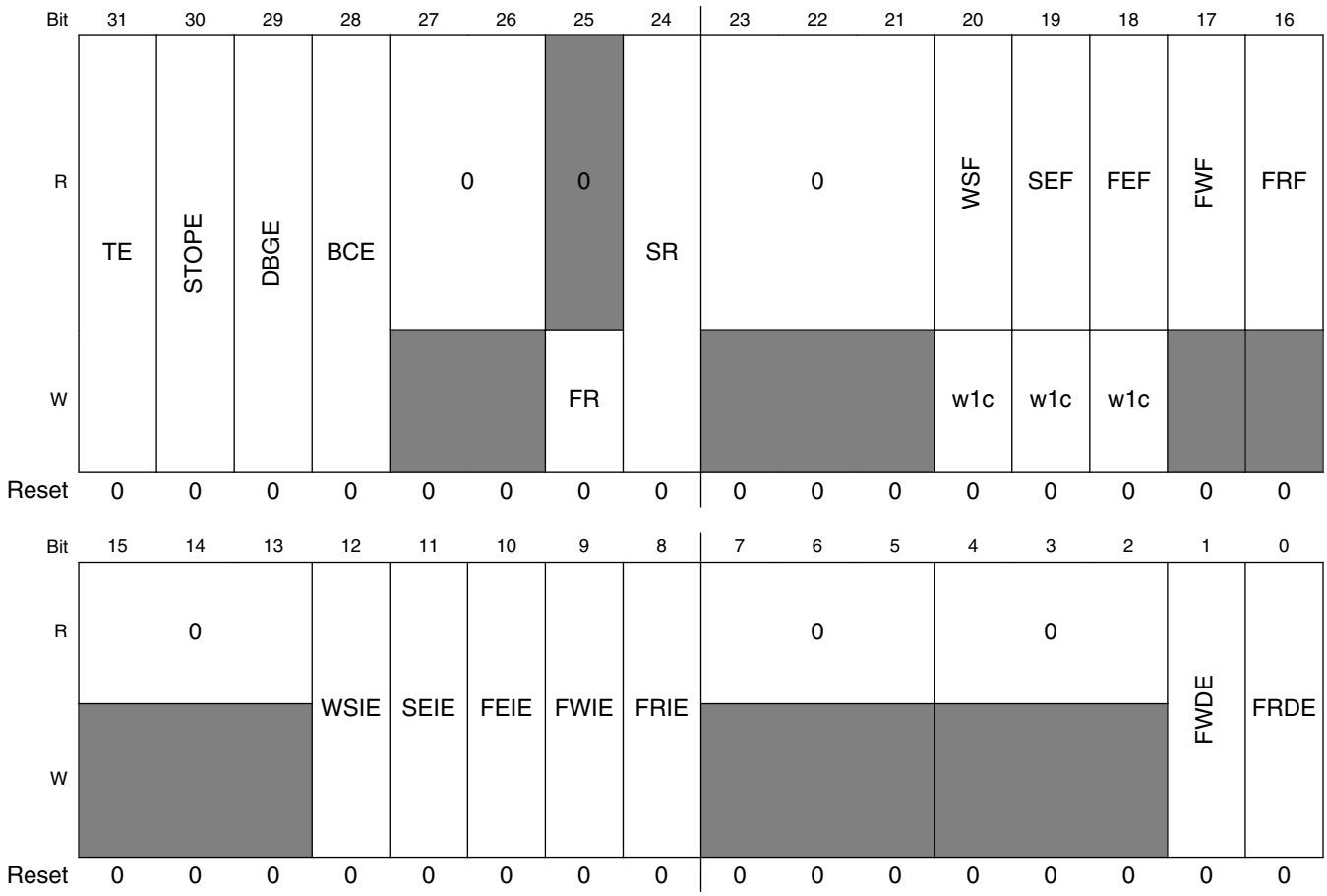
A read or write access to an address after the last register will result in a bus error.

## I2S memory map

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4002_F000 | SAI Transmit Control Register (I2S0_TCSR) | 32 | R/W | 0000_0000h | 53.3.1/1561 |
| 4002_F004 | SAI Transmit Configuration 1 Register (I2S0_TCR1) | 32 | R/W | 0000_0000h | 53.3.2/1564 |
| 4002_F008 | SAI Transmit Configuration 2 Register (I2S0_TCR2) | 32 | R/W | 0000_0000h | 53.3.3/1564 |
| 4002_F00C | SAI Transmit Configuration 3 Register (I2S0_TCR3) | 32 | R/W | 0000_0000h | 53.3.4/1566 |
| 4002_F010 | SAI Transmit Configuration 4 Register (I2S0_TCR4) | 32 | R/W | 0000_0000h | 53.3.5/1567 |
| 4002_F014 | SAI Transmit Configuration 5 Register (I2S0_TCR5) | 32 | R/W | 0000_0000h | 53.3.6/1568 |
| 4002_F020 | SAI Transmit Data Register (I2S0_TDR0) | 32 | W (always reads 0) | 0000_0000h | 53.3.7/1569 |
| 4002_F024 | SAI Transmit Data Register (I2S0_TDR1) | 32 | W (always reads 0) | 0000_0000h | 53.3.7/1569 |
| 4002_F040 | SAI Transmit FIFO Register (I2S0_TFR0) | 32 | R | 0000_0000h | 53.3.8/1570 |
| 4002_F044 | SAI Transmit FIFO Register (I2S0_TFR1) | 32 | R | 0000_0000h | 53.3.8/1570 |
| 4002_F060 | SAI Transmit Mask Register (I2S0_TMR) | 32 | R/W | 0000_0000h | 53.3.9/1570 |
| 4002_F080 | SAI Receive Control Register (I2S0_RCSR) | 32 | R/W | 0000_0000h | 53.3.10/ 1571 |
| 4002_F084 | SAI Receive Configuration 1 Register (I2S0_RCR1) | 32 | R/W | 0000_0000h | 53.3.11/ 1574 |
| 4002_F088 | SAI Receive Configuration 2 Register (I2S0_RCR2) | 32 | R/W | 0000_0000h | 53.3.12/ 1575 |
| 4002_F08C | SAI Receive Configuration 3 Register (I2S0_RCR3) | 32 | R/W | 0000_0000h | 53.3.13/ 1576 |
| 4002_F090 | SAI Receive Configuration 4 Register (I2S0_RCR4) | 32 | R/W | 0000_0000h | 53.3.14/ 1577 |
| 4002_F094 | SAI Receive Configuration 5 Register (I2S0_RCR5) | 32 | R/W | 0000_0000h | 53.3.15/ 1578 |
| 4002_F0A0 | SAI Receive Data Register (I2S0_RDR0) | 32 | R | 0000_0000h | 53.3.16/ 1579 |
| 4002_F0A4 | SAI Receive Data Register (I2S0_RDR1) | 32 | R | 0000_0000h | 53.3.16/ 1579 |
| 4002_F0C0 | SAI Receive FIFO Register (I2S0_RFR0) | 32 | R | 0000_0000h | 53.3.17/ 1580 |
| 4002_F0C4 | SAI Receive FIFO Register (I2S0_RFR1) | 32 | R | 0000_0000h | 53.3.17/ 1580 |
| 4002_F0E0 | SAI Receive Mask Register (I2S0_RMR) | 32 | R/W | 0000_0000h | 53.3.18/ 1580 |
| 4002_F100 | SAI MCLK Control Register (I2S0_MCR) | 32 | R/W | 0000_0000h | 53.3.19/ 1581 |
| 4002_F104 | SAI MCLK Divide Register (I2S0_MDR) | 32 | R/W | 0000_0000h | 53.3.20/ 1582 |

## 53.3.1 SAI Transmit Control Register (I2Sx_TCSR)

Address: 4002_F000h base + 0h offset = 4002_F000h




### I2Sx_TCSR field descriptions

| Field | Description |
|---|---|
| 31<br>TE | Transmitter Enable<br><br>Enables/disables the transmitter. When software clears this field, the transmitter remains enabled, and this bit remains set, until the end of the current frame.<br><br>0 Transmitter is disabled.<br>1 Transmitter is enabled, or transmitter has been disabled and has not yet reached end of frame. |
| 30<br>STOPE | Stop Enable<br><br>Configures transmitter operation in Stop mode. This field is ignored and the transmitter is disabled in all low-leakage stop modes.<br><br>0 Transmitter disabled in Stop mode.<br>1 Transmitter enabled in Stop mode. |
| 29<br>DBGE | Debug Enable |

*Table continues on the next page...*

## I2Sx_TCSR field descriptions (continued)

| Field | Description |
|---|---|
| | Enables/disables transmitter operation in Debug mode. The transmit bit clock is not affected by debug mode.<br><br>0 Transmitter is disabled in Debug mode, after completing the current frame.<br>1 Transmitter is enabled in Debug mode. |
| 28<br>BCE | Bit Clock Enable<br><br>Enables the transmit bit clock, separately from the TE. This field is automatically set whenever TE is set. When software clears this field, the transmit bit clock remains enabled, and this bit remains set, until the end of the current frame.<br><br>0 Transmit bit clock is disabled.<br>1 Transmit bit clock is enabled. |
| 27–26<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 25<br>FR | FIFO Reset<br><br>Resets the FIFO pointers. Reading this field will always return zero. FIFO pointers should only be reset when the transmitter is disabled or the FIFO error flag is set.<br><br>0 No effect.<br>1 FIFO reset. |
| 24<br>SR | Software Reset<br><br>When set, resets the internal transmitter logic including the FIFO pointers. Software-visible registers are not affected, except for the status registers.<br><br>0 No effect.<br>1 Software reset. |
| 23–21<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 20<br>WSF | Word Start Flag<br><br>Indicates that the start of the configured word has been detected. Write a logic 1 to this field to clear this flag.<br><br>0 Start of word not detected.<br>1 Start of word detected. |
| 19<br>SEF | Sync Error Flag<br><br>Indicates that an error in the externally-generated frame sync has been detected. Write a logic 1 to this field to clear this flag.<br><br>0 Sync error not detected.<br>1 Frame sync error detected. |
| 18<br>FEF | FIFO Error Flag<br><br>Indicates that an enabled transmit FIFO has underrun. Write a logic 1 to this field to clear this flag.<br><br>0 Transmit underrun not detected.<br>1 Transmit underrun detected. |

*Table continues on the next page...*

## I2Sx_TCSR field descriptions (continued)

| Field | Description |
|---|---|
| 17<br>FWF | FIFO Warning Flag<br><br>Indicates that an enabled transmit FIFO is empty.<br><br>0 No enabled transmit FIFO is empty.<br>1 Enabled transmit FIFO is empty. |
| 16<br>FRF | FIFO Request Flag<br><br>Indicates that the number of words in an enabled transmit channel FIFO is less than or equal to the transmit FIFO watermark.<br><br>0 Transmit FIFO watermark has not been reached.<br>1 Transmit FIFO watermark has been reached. |
| 15–13<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 12<br>WSIE | Word Start Interrupt Enable<br><br>Enables/disables word start interrupts.<br><br>0 Disables interrupt.<br>1 Enables interrupt. |
| 11<br>SEIE | Sync Error Interrupt Enable<br><br>Enables/disables sync error interrupts.<br><br>0 Disables interrupt.<br>1 Enables interrupt. |
| 10<br>FEIE | FIFO Error Interrupt Enable<br><br>Enables/disables FIFO error interrupts.<br><br>0 Disables the interrupt.<br>1 Enables the interrupt. |
| 9<br>FWIE | FIFO Warning Interrupt Enable<br><br>Enables/disables FIFO warning interrupts.<br><br>0 Disables the interrupt.<br>1 Enables the interrupt. |
| 8<br>FRIE | FIFO Request Interrupt Enable<br><br>Enables/disables FIFO request interrupts.<br><br>0 Disables the interrupt.<br>1 Enables the interrupt. |
| 7–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4–2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 1<br>FWDE | FIFO Warning DMA Enable |

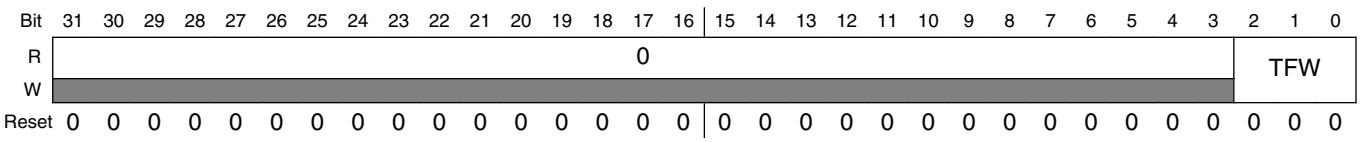
*Table continues on the next page...*

I2Sx_TCSR field descriptions (continued)

| Field | Description |
|---|---|
| | Enables/disables DMA requests.<br><br>0 Disables the DMA request.<br>1 Enables the DMA request. |
| 0<br>FRDE | FIFO Request DMA Enable<br><br>Enables/disables DMA requests.<br><br>0 Disables the DMA request.<br>1 Enables the DMA request. |

## 53.3.2 SAI Transmit Configuration 1 Register (I2Sx_TCR1)

Address: 4002_F000h base + 4h offset = 4002_F004h



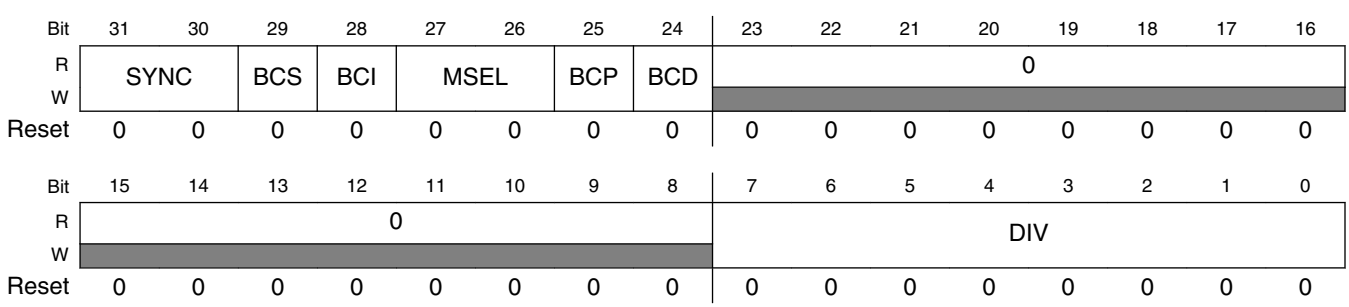

I2Sx_TCR1 field descriptions

| Field | Description |
|---|---|
| 31–3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2–0<br>TFW | Transmit FIFO Watermark<br><br>Configures the watermark level for all enabled transmit channels. |

## 53.3.3 SAI Transmit Configuration 2 Register (I2Sx_TCR2)

This register must not be altered when TCSR[TE] is set.

Address: 4002_F000h base + 8h offset = 4002_F008h



## I2Sx_TCR2 field descriptions

| Field | Description |
|---|---|
| 31–30<br>SYNC | Synchronous Mode<br><br>Configures between asynchronous and synchronous modes of operation. When configured for a synchronous mode of operation, the receiver must be configured for asynchronous operation.<br><br>00 Asynchronous mode.<br>01 Synchronous with receiver.<br>10 Synchronous with another SAI transmitter.<br>11 Synchronous with another SAI receiver. |
| 29<br>BCS | Bit Clock Swap<br><br>This field swaps the bit clock used by the transmitter. When the transmitter is configured in asynchronous mode and this bit is set, the transmitter is clocked by the receiver bit clock (SAI_RX_BCLK). This allows the transmitter and receiver to share the same bit clock, but the transmitter continues to use the transmit frame sync (SAI_TX_SYNC).<br><br>When the transmitter is configured in synchronous mode, the transmitter BCS field and receiver BCS field must be set to the same value. When both are set, the transmitter and receiver are both clocked by the transmitter bit clock (SAI_TX_BCLK) but use the receiver frame sync (SAI_RX_SYNC).<br><br>0 Use the normal bit clock source.<br>1 Swap the bit clock source. |
| 28<br>BCI | Bit Clock Input<br><br>When this field is set and using an internally generated bit clock in either synchronous or asynchronous mode, the bit clock actually used by the transmitter is delayed by the pad output delay (the transmitter is clocked by the pad input as if the clock was externally generated). This has the effect of decreasing the data input setup time, but increasing the data output valid time.<br><br>The slave mode timing from the datasheet should be used for the transmitter when this bit is set. In synchronous mode, this bit allows the transmitter to use the slave mode timing from the datasheet, while the receiver uses the master mode timing. This field has no effect when configured for an externally generated bit clock .<br><br>0 No effect.<br>1 Internal logic is clocked as if bit clock was externally generated. |
| 27–26<br>MSEL | MCLK Select<br><br>Selects the audio Master Clock option used to generate an internally generated bit clock. This field has no effect when configured for an externally generated bit clock.<br><br>**NOTE:** Depending on the device, some Master Clock options might not be available. See the chip configuration details for the availability and chip-specific meaning of each option.<br><br>00 Bus Clock selected.<br>01 Master Clock (MCLK) 1 option selected.<br>10 Master Clock (MCLK) 2 option selected.<br>11 Master Clock (MCLK) 3 option selected. |
| 25<br>BCP | Bit Clock Polarity<br><br>Configures the polarity of the bit clock.<br><br>0 Bit clock is active high with drive outputs on rising edge and sample inputs on falling edge.<br>1 Bit clock is active low with drive outputs on falling edge and sample inputs on rising edge. |

*Table continues on the next page...*

**I2Sx_TCR2 field descriptions (continued)**

| Field | Description |
|---|---|
| 24<br>BCD | Bit Clock Direction<br><br>Configures the direction of the bit clock.<br><br>0 Bit clock is generated externally in Slave mode.<br>1 Bit clock is generated internally in Master mode. |
| 23–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7–0<br>DIV | Bit Clock Divide<br><br>Divides down the audio master clock to generate the bit clock when configured for an internal bit clock. The division value is (DIV + 1) * 2. |

## 53.3.4 SAI Transmit Configuration 3 Register (I2Sx_TCR3)

This register must not be altered when TCSR[TE] is set.

Address: 4002_F000h base + Ch offset = 4002_F00Ch

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | | | | | | TCE | | 0 | | | | | | | | | | | WDFL | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**I2Sx_TCR3 field descriptions**

| Field | Description |
|---|---|
| 31–24<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 23–18<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 17–16<br>TCE | Transmit Channel Enable<br><br>Enables the corresponding data channel for transmit operation. A channel must be enabled before its FIFO is accessed.<br><br>0 Transmit data channel N is disabled.<br>1 Transmit data channel N is enabled. |
| 15–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4–0<br>WDFL | Word Flag Configuration<br><br>Configures which word sets the start of word flag. The value written must be one less than the word number. For example, writing 0 configures the first word in the frame. When configured to a value greater than TCR4[FRSZ], then the start of word flag is never set. |

## 53.3.5 SAI Transmit Configuration 4 Register (I2Sx_TCR4)

This register must not be altered when TCSR[TE] is set.

Address: 4002_F000h base + 10h offset = 4002_F010h

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

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | SYWD | | | | | 0 | | | MF | FSE | 0 | FSP | FSD |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### I2Sx_TCR4 field descriptions

| Field | Description |
|---|---|
| 31–29<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 28<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 27–26<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 25–24<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 23–21<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 20–16<br>FRSZ | Frame size<br><br>Configures the number of words in each frame. The value written must be one less than the number of words in the frame. For example, write 0 for one word per frame. The maximum supported frame size is 32 words. |
| 15–13<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 12–8<br>SYWD | Sync Width<br><br>Configures the length of the frame sync in number of bit clocks. The value written must be one less than the number of bit clocks. For example, write 0 for the frame sync to assert for one bit clock only. The sync width cannot be configured longer than the first word of the frame. |
| 7–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>MF | MSB First<br><br>Configures whether the LSB or the MSB is transmitted first.<br><br>0 LSB is transmitted first.<br>1 MSB is transmitted first. |

*Table continues on the next page...*

**I2Sx_TCR4 field descriptions (continued)**

| Field | Description |
|---|---|
| 3<br>FSE | Frame Sync Early<br><br>0 Frame sync asserts with the first bit of the frame.<br>1 Frame sync asserts one bit before the first bit of the frame. |
| 2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 1<br>FSP | Frame Sync Polarity<br><br>Configures the polarity of the frame sync.<br><br>0 Frame sync is active high.<br>1 Frame sync is active low. |
| 0<br>FSD | Frame Sync Direction<br><br>Configures the direction of the frame sync.<br><br>0 Frame sync is generated externally in Slave mode.<br>1 Frame sync is generated internally in Master mode. |

## 53.3.6 SAI Transmit Configuration 5 Register (I2Sx_TCR5)

This register must not be altered when TCSR[TE] is set.

Address: 4002_F000h base + 14h offset = 4002_F014h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | WNW | | | | | 0 | | | W0W | | | | | 0 | | | FBT | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**I2Sx_TCR5 field descriptions**

| Field | Description |
|---|---|
| 31–29<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 28–24<br>WNW | Word N Width<br><br>Configures the number of bits in each word, for each word except the first in the frame. The value written must be one less than the number of bits per word. Word width of less than 8 bits is not supported. |
| 23–21<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 20–16<br>W0W | Word 0 Width<br><br>Configures the number of bits in the first word in each frame. The value written must be one less than the number of bits in the first word. Word width of less than 8 bits is not supported if there is only one word per frame. |
| 15–13<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

*Table continues on the next page...*

### I2Sx_TCR5 field descriptions (continued)

| Field | Description |
|---|---|
| 12–8<br>FBT | First Bit Shifted<br><br>Configures the bit index for the first bit transmitted for each word in the frame. If configured for MSB First, the index of the next bit transmitted is one less than the current bit transmitted. If configured for LSB First, the index of the next bit transmitted is one more than the current bit transmitted. The value written must be greater than or equal to the word width when configured for MSB First. The value written must be less than or equal to 31-word width when configured for LSB First. |
| 7–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 53.3.7 SAI Transmit Data Register (I2Sx_TDR*n*)

Address: 4002_F000h base + 20h offset + (4d × i), where i=0d to 1d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | TDR[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### I2Sx_TDR*n* field descriptions

| Field | Description |
|---|---|
| 31–0<br>TDR[31:0] | Transmit Data Register<br><br>The corresponding TCR3[TCE] bit must be set before accessing the channel's transmit data register. Writes to this register when the transmit FIFO is not full will push the data written into the transmit data FIFO. Writes to this register when the transmit FIFO is full are ignored. |

## 53.3.8 SAI Transmit FIFO Register (I2Sx_TFR*n*)

The MSB of the read and write pointers is used to distinguish between FIFO full and empty conditions. If the read and write pointers are identical, then the FIFO is empty. If the read and write pointers are identical except for the MSB, then the FIFO is full.

Address: 4002_F000h base + 40h offset + (4d × i), where i=0d to 1d

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

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

**I2Sx_TFR*n* field descriptions**

| Field | Description |
|---|---|
| 31<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 30–20<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 19–16<br>WFP | Write FIFO Pointer<br><br>FIFO write pointer for transmit data channel. |
| 15–4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3–0<br>RFP | Read FIFO Pointer<br><br>FIFO read pointer for transmit data channel. |

## 53.3.9 SAI Transmit Mask Register (I2Sx_TMR)

This register is double-buffered and updates:

1. When TCSR[TE] is first set
2. At the end of each frame.

This allows the masked words in each frame to change from frame to frame.

Address: 4002_F000h base + 60h offset = 4002_F060h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TWM | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### I2Sx_TMR field descriptions

| Field | Description |
|---|---|
| 31–0<br>TWM | Transmit Word Mask<br><br>Configures whether the transmit word is masked (transmit data pin tristated and transmit data not read from FIFO) for the corresponding word in the frame.<br><br>0 Word N is enabled.<br>1 Word N is masked. The transmit data pins are tri-stated when masked. |

## 53.3.10 SAI Receive Control Register (I2Sx_RCSR)

Address: 4002_F000h base + 80h offset = 4002_F080h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RE | STOPE | DBGE | BCE | 0 | | 0 | SR | 0 | | | WSF | SEF | FEF | FWF | FRF |
| W | | | | | | | FR | | | | | w1c | w1c | w1c | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | WSIE | SEIE | FEIE | FWIE | FRIE | 0 | | | 0 | | | FWDE | FRDE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## I2Sx_RCSR field descriptions

| Field | Description |
|---|---|
| 31<br>RE | Receiver Enable<br><br>Enables/disables the receiver. When software clears this field, the receiver remains enabled, and this bit remains set, until the end of the current frame.<br><br>0 Receiver is disabled.<br>1 Receiver is enabled, or receiver has been disabled and has not yet reached end of frame. |
| 30<br>STOPE | Stop Enable<br><br>Configures receiver operation in Stop mode. This bit is ignored and the receiver is disabled in all low-leakage stop modes.<br><br>0 Receiver disabled in Stop mode.<br>1 Receiver enabled in Stop mode. |
| 29<br>DBGE | Debug Enable<br><br>Enables/disables receiver operation in Debug mode. The receive bit clock is not affected by Debug mode.<br><br>0 Receiver is disabled in Debug mode, after completing the current frame.<br>1 Receiver is enabled in Debug mode. |
| 28<br>BCE | Bit Clock Enable<br><br>Enables the receive bit clock, separately from RE. This field is automatically set whenever RE is set. When software clears this field, the receive bit clock remains enabled, and this field remains set, until the end of the current frame.<br><br>0 Receive bit clock is disabled.<br>1 Receive bit clock is enabled. |
| 27–26<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 25<br>FR | FIFO Reset<br><br>Resets the FIFO pointers. Reading this field will always return zero. FIFO pointers should only be reset when the receiver is disabled or the FIFO error flag is set.<br><br>0 No effect.<br>1 FIFO reset. |
| 24<br>SR | Software Reset<br><br>Resets the internal receiver logic including the FIFO pointers. Software-visible registers are not affected, except for the status registers.<br><br>0 No effect.<br>1 Software reset. |
| 23–21<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 20<br>WSF | Word Start Flag<br><br>Indicates that the start of the configured word has been detected. Write a logic 1 to this field to clear this flag. |

*Table continues on the next page...*

## I2Sx_RCSR field descriptions (continued)

| Field | Description |
|---|---|
| | 0 Start of word not detected.<br>1 Start of word detected. |
| 19<br>SEF | Sync Error Flag<br><br>Indicates that an error in the externally-generated frame sync has been detected. Write a logic 1 to this field to clear this flag.<br><br>0 Sync error not detected.<br>1 Frame sync error detected. |
| 18<br>FEF | FIFO Error Flag<br><br>Indicates that an enabled receive FIFO has overflowed. Write a logic 1 to this field to clear this flag.<br><br>0 Receive overflow not detected.<br>1 Receive overflow detected. |
| 17<br>FWF | FIFO Warning Flag<br><br>Indicates that an enabled receive FIFO is full.<br><br>0 No enabled receive FIFO is full.<br>1 Enabled receive FIFO is full. |
| 16<br>FRF | FIFO Request Flag<br><br>Indicates that the number of words in an enabled receive channel FIFO is greater than the receive FIFO watermark.<br><br>0 Receive FIFO watermark not reached.<br>1 Receive FIFO watermark has been reached. |
| 15–13<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 12<br>WSIE | Word Start Interrupt Enable<br><br>Enables/disables word start interrupts.<br><br>0 Disables interrupt.<br>1 Enables interrupt. |
| 11<br>SEIE | Sync Error Interrupt Enable<br><br>Enables/disables sync error interrupts.<br><br>0 Disables interrupt.<br>1 Enables interrupt. |
| 10<br>FEIE | FIFO Error Interrupt Enable<br><br>Enables/disables FIFO error interrupts.<br><br>0 Disables the interrupt.<br>1 Enables the interrupt. |
| 9<br>FWIE | FIFO Warning Interrupt Enable<br><br>Enables/disables FIFO warning interrupts. |

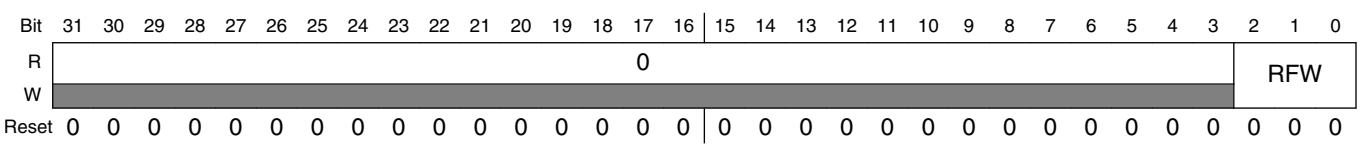
*Table continues on the next page...*

I2Sx_RCSR field descriptions (continued)

| Field | Description |
|---|---|
| | 0 Disables the interrupt.<br>1 Enables the interrupt. |
| 8<br>FRIE | FIFO Request Interrupt Enable<br><br>Enables/disables FIFO request interrupts.<br><br>0 Disables the interrupt.<br>1 Enables the interrupt. |
| 7–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4–2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 1<br>FWDE | FIFO Warning DMA Enable<br><br>Enables/disables DMA requests.<br><br>0 Disables the DMA request.<br>1 Enables the DMA request. |
| 0<br>FRDE | FIFO Request DMA Enable<br><br>Enables/disables DMA requests.<br><br>0 Disables the DMA request.<br>1 Enables the DMA request. |

## 53.3.11 SAI Receive Configuration 1 Register (I2Sx_RCR1)

Address: 4002_F000h base + 84h offset = 4002_F084h




I2Sx_RCR1 field descriptions

| Field | Description |
|---|---|
| 31–3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 2–0<br>RFW | Receive FIFO Watermark<br><br>Configures the watermark level for all enabled receiver channels. |

## 53.3.12 SAI Receive Configuration 2 Register (I2Sx_RCR2)

This register must not be altered when RCSR[RE] is set.

Address: 4002_F000h base + 88h offset = 4002_F088h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SYNC | | BCS | BCI | MSEL | | BCP | BCD | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

### I2Sx_RCR2 field descriptions

| Field | Description |
|---|---|
| 31–30<br>SYNC | Synchronous Mode<br><br>Configures between asynchronous and synchronous modes of operation. When configured for a synchronous mode of operation, the transmitter must be configured for asynchronous operation.<br><br>00 Asynchronous mode.<br>01 Synchronous with transmitter.<br>10 Synchronous with another SAI receiver.<br>11 Synchronous with another SAI transmitter. |
| 29<br>BCS | Bit Clock Swap<br><br>This field swaps the bit clock used by the receiver. When the receiver is configured in asynchronous mode and this bit is set, the receiver is clocked by the transmitter bit clock (SAI_TX_BCLK). This allows the transmitter and receiver to share the same bit clock, but the receiver continues to use the receiver frame sync (SAI_RX_SYNC).<br><br>When the receiver is configured in synchronous mode, the transmitter BCS field and receiver BCS field must be set to the same value. When both are set, the transmitter and receiver are both clocked by the receiver bit clock (SAI_RX_BCLK) but use the transmitter frame sync (SAI_TX_SYNC).<br><br>0 Use the normal bit clock source.<br>1 Swap the bit clock source. |
| 28<br>BCI | Bit Clock Input<br><br>When this field is set and using an internally generated bit clock in either synchronous or asynchronous mode, the bit clock actually used by the receiver is delayed by the pad output delay (the receiver is clocked by the pad input as if the clock was externally generated). This has the effect of decreasing the data input setup time, but increasing the data output valid time.<br><br>The slave mode timing from the datasheet should be used for the receiver when this bit is set. In synchronous mode, this bit allows the receiver to use the slave mode timing from the datasheet, while the transmitter uses the master mode timing. This field has no effect when configured for an externally generated bit clock .<br><br>0 No effect.<br>1 Internal logic is clocked as if bit clock was externally generated. |

*Table continues on the next page...*

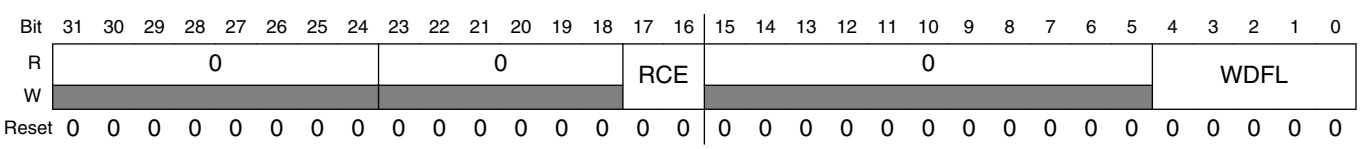
**I2Sx_RCR2 field descriptions (continued)**

| Field | Description |
|---|---|
| 27–26<br>MSEL | MCLK Select<br><br>Selects the audio Master Clock option used to generate an internally generated bit clock. This field has no effect when configured for an externally generated bit clock.<br><br>**NOTE:** Depending on the device, some Master Clock options might not be available. See the chip configuration details for the availability and chip-specific meaning of each option.<br><br>00 Bus Clock selected.<br>01 Master Clock (MCLK) 1 option selected.<br>10 Master Clock (MCLK) 2 option selected.<br>11 Master Clock (MCLK) 3 option selected. |
| 25<br>BCP | Bit Clock Polarity<br><br>Configures the polarity of the bit clock.<br><br>0 Bit Clock is active high with drive outputs on rising edge and sample inputs on falling edge.<br>1 Bit Clock is active low with drive outputs on falling edge and sample inputs on rising edge. |
| 24<br>BCD | Bit Clock Direction<br><br>Configures the direction of the bit clock.<br><br>0 Bit clock is generated externally in Slave mode.<br>1 Bit clock is generated internally in Master mode. |
| 23–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7–0<br>DIV | Bit Clock Divide<br><br>Divides down the audio master clock to generate the bit clock when configured for an internal bit clock. The division value is (DIV + 1) * 2. |

## 53.3.13 SAI Receive Configuration 3 Register (I2Sx_RCR3)

This register must not be altered when RCSR[RE] is set.

Address: 4002_F000h base + 8Ch offset = 4002_F08Ch




**I2Sx_RCR3 field descriptions**

| Field | Description |
|---|---|
| 31–24<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

*Table continues on the next page...*

**I2Sx_RCR3 field descriptions (continued)**

| Field | Description |
|---|---|
| 23–18<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 17–16<br>RCE | Receive Channel Enable<br><br>Enables the corresponding data channel for receive operation. A channel must be enabled before its FIFO is accessed.<br><br>0 Receive data channel N is disabled.<br>1 Receive data channel N is enabled. |
| 15–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4–0<br>WDFL | Word Flag Configuration<br><br>Configures which word the start of word flag is set. The value written should be one less than the word number (for example, write zero to configure for the first word in the frame). When configured to a value greater than the Frame Size field, then the start of word flag is never set. |

## 53.3.14 SAI Receive Configuration 4 Register (I2Sx_RCR4)

This register must not be altered when RCSR[RE] is set.

Address: 4002_F000h base + 90h offset = 4002_F090h

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

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | SYWD | | | | | 0 | | | MF | FSE | 0 | FSP | FSD |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**I2Sx_RCR4 field descriptions**

| Field | Description |
|---|---|
| 31–29<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 28<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 27–26<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 25–24<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 23–21<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

*Table continues on the next page...*

I2Sx_RCR4 field descriptions (continued)

| Field | Description |
|---|---|
| 20–16<br>FRSZ | Frame Size<br><br>Configures the number of words in each frame. The value written must be one less than the number of words in the frame. For example, write 0 for one word per frame. The maximum supported frame size is 32 words. |
| 15–13<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 12–8<br>SYWD | Sync Width<br><br>Configures the length of the frame sync in number of bit clocks. The value written must be one less than the number of bit clocks. For example, write 0 for the frame sync to assert for one bit clock only. The sync width cannot be configured longer than the first word of the frame. |
| 7–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>MF | MSB First<br><br>Configures whether the LSB or the MSB is received first.<br><br>0 LSB is received first.<br>1 MSB is received first. |
| 3<br>FSE | Frame Sync Early<br><br>0 Frame sync asserts with the first bit of the frame.<br>1 Frame sync asserts one bit before the first bit of the frame. |
| 2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 1<br>FSP | Frame Sync Polarity<br><br>Configures the polarity of the frame sync.<br><br>0 Frame sync is active high.<br>1 Frame sync is active low. |
| 0<br>FSD | Frame Sync Direction<br><br>Configures the direction of the frame sync.<br><br>0 Frame Sync is generated externally in Slave mode.<br>1 Frame Sync is generated internally in Master mode. |

## 53.3.15 SAI Receive Configuration 5 Register (I2Sx_RCR5)

This register must not be altered when RCSR[RE] is set.

Address: 4002_F000h base + 94h offset = 4002_F094h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | WNW | | | | | 0 | | | W0W | | | | | 0 | | | FBT | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### I2Sx_RCR5 field descriptions

| Field | Description |
|---|---|
| 31–29<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 28–24<br>WNW | Word N Width<br><br>Configures the number of bits in each word, for each word except the first in the frame. The value written must be one less than the number of bits per word. Word width of less than 8 bits is not supported. |
| 23–21<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 20–16<br>W0W | Word 0 Width<br><br>Configures the number of bits in the first word in each frame. The value written must be one less than the number of bits in the first word. Word width of less than 8 bits is not supported if there is only one word per frame. |
| 15–13<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 12–8<br>FBT | First Bit Shifted<br><br>Configures the bit index for the first bit received for each word in the frame. If configured for MSB First, the index of the next bit received is one less than the current bit received. If configured for LSB First, the index of the next bit received is one more than the current bit received. The value written must be greater than or equal to the word width when configured for MSB First. The value written must be less than or equal to 31-word width when configured for LSB First. |
| 7–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 53.3.16 SAI Receive Data Register (I2Sx_RDR*n*)

Reading this register introduces one additional peripheral clock wait state on each read.

Address: 4002_F000h base + A0h offset + (4d × i), where i=0d to 1d

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RDR[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### I2Sx_RDR*n* field descriptions

| Field | Description |
|---|---|
| 31–0<br>RDR[31:0] | Receive Data Register<br><br>The corresponding RCR3[RCE] bit must be set before accessing the channel's receive data register. Reads from this register when the receive FIFO is not empty will return the data from the top of the receive FIFO. Reads from this register when the receive FIFO is empty are ignored. |

## 53.3.17 SAI Receive FIFO Register (I2Sx_RFR*n*)

The MSB of the read and write pointers is used to distinguish between FIFO full and empty conditions. If the read and write pointers are identical, then the FIFO is empty. If the read and write pointers are identical except for the MSB, then the FIFO is full.

Address: 4002_F000h base + C0h offset + (4d × i), where i=0d to 1d

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

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

**I2Sx_RFR*n* field descriptions**

| Field | Description |
|---|---|
| 31–20<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 19–16<br>WFP | Write FIFO Pointer<br><br>FIFO write pointer for receive data channel. |
| 15<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 14–4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3–0<br>RFP | Read FIFO Pointer<br><br>FIFO read pointer for receive data channel. |

## 53.3.18 SAI Receive Mask Register (I2Sx_RMR)

This register is double-buffered and updates:

1. When RCSR[RE] is first set
2. At the end of each frame

This allows the masked words in each frame to change from frame to frame.

Address: 4002_F000h base + E0h offset = 4002_F0E0h

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RWM | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**I2Sx_RMR field descriptions**

| Field | Description |
|---|---|
| 31–0<br>RWM | Receive Word Mask<br><br>Configures whether the receive word is masked (received data ignored and not written to receive FIFO) for the corresponding word in the frame.<br><br>0 Word N is enabled.<br>1 Word N is masked. |

## 53.3.19 SAI MCLK Control Register (I2Sx_MCR)

The MCLK Control Register (MCR) controls the clock source and direction of the audio master clock.

Address: 4002_F000h base + 100h offset = 4002_F100h

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

| Bit | 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 |

**I2Sx_MCR field descriptions**

| Field | Description |
|---|---|
| 31<br>DUF | Divider Update Flag<br><br>Provides the status of on-the-fly updates to the MCLK divider ratio.<br><br>0 MCLK divider ratio is not being updated currently.<br>1 MCLK divider ratio is updating on-the-fly. Further updates to the MCLK divider ratio are blocked while this flag remains set. |
| 30<br>MOE | MCLK Output Enable<br><br>Enables the MCLK divider and configures the MCLK signal pin as an output. When software clears this field, it remains set until the MCLK divider is fully disabled. |

*Table continues on the next page...*

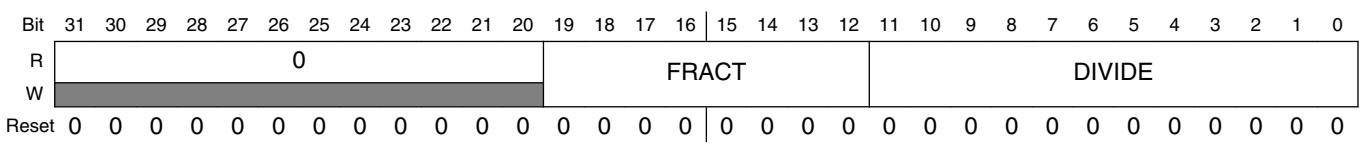
**I2Sx_MCR field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 MCLK signal pin is configured as an input that bypasses the MCLK divider.<br>1 MCLK signal pin is configured as an output from the MCLK divider and the MCLK divider is enabled. |
| 29–26<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 25–24<br>MICS | MCLK Input Clock Select<br><br>Selects the clock input to the MCLK divider. This field cannot be changed while the MCLK divider is enabled. See the chip configuration details for information about the connections to these inputs.<br><br>00 MCLK divider input clock 0 selected.<br>01 MCLK divider input clock 1 selected.<br>10 MCLK divider input clock 2 selected.<br>11 MCLK divider input clock 3 selected. |
| 23–0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 53.3.20 SAI MCLK Divide Register (I2Sx_MDR)

The MCLK Divide Register (MDR) configures the MCLK divide ratio. Although the MDR can be changed when the MCLK divider clock is enabled, additional writes to the MDR are blocked while MCR[DUF] is set. Writes to the MDR when the MCLK divided clock is disabled do not set MCR[DUF].

Address: 4002_F000h base + 104h offset = 4002_F104h




**I2Sx_MDR field descriptions**

| Field | Description |
|---|---|
| 31–20<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 19–12<br>FRACT | MCLK Fraction<br><br>Sets the MCLK divide ratio such that: MCLK output = MCLK input * ( (FRACT + 1) / (DIVIDE + 1) ).<br>FRACT must be set equal or less than the value in the DIVIDE field. |
| 11–0<br>DIVIDE | MCLK Divide<br><br>Sets the MCLK divide ratio such that: MCLK output = MCLK input * ( (FRACT + 1) / (DIVIDE + 1) ).<br>FRACT must be set equal or less than the value in the DIVIDE field. |

# 53.4 Functional description

## 53.4.1 SAI clocking

The SAI clocks include:

- The audio master clock
- The bit clock
- The bus clock

### 53.4.1.1 Audio master clock

The audio master clock is used to generate the bit clock when the receiver or transmitter is configured for an internally generated bit clock. The transmitter and receiver can independently select between the bus clock and up to three audio master clocks to generate the bit clock.

Each SAI peripheral can control the input clock selection, pin direction and divide ratio of one audio master clock. The input clock selection and pin direction cannot be altered if an SAI module using that audio master clock has been enabled. The MCLK divide ratio can be altered while an SAI is using that master clock, although the change in the divide ratio takes several cycles. MCR[DUF] can be polled to determine when the divide ratio change has completed.

The audio master clock generation and selection is chip-specific. Refer to chip-specific clocking information about how the audio master clocks are generated. A typical implementation appears in the following figure.

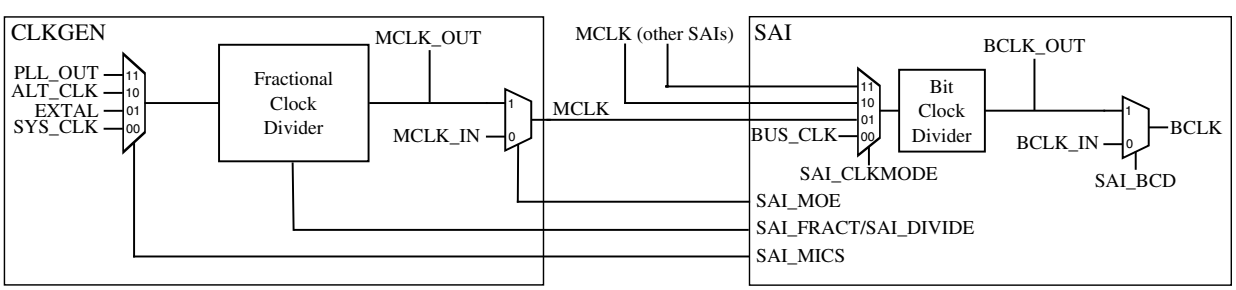



**Figure 53-58. SAI master clock generation**

The MCLK fractional clock divider uses both clock edges from the input clock to generate a divided down clock that will approximate the output frequency, but without creating any new clock edges. Configuring FRACT and DIVIDE to the same value will result in a divide by 1 clock, while configuring FRACT higher than DIVIDE is not supported. The duty cycle can range from 66/33 when FRACT is set to one less than DIVIDE down to 50/50 for integer divide ratios, and will approach 50/50 for large non-integer divide ratios. There is no cycle to cycle jitter or duty cycle variance when the divide ratio is an integer or half integer, otherwise the divider output will oscillate between the two divided frequencies that are the closest integer or half integer divisors of the divider input clock frequency. The maximum jitter is therefore equal to half the divider input clock period, since both edges of the input clock are used in generating the divided clock.

#### 53.4.1.2 Bit clock

The SAI transmitter and receiver support asynchronous free-running bit clocks that can be generated internally from an audio master clock or supplied externally. There is also the option for synchronous bit clock and frame sync operation between the receiver and transmitter or between multiple SAI peripherals.

Externally generated bit clocks must be:

- Enabled before the SAI transmitter or receiver is enabled
- Disabled after the SAI transmitter or receiver is disabled and completes its current frames

If the SAI transmitter or receiver is using an externally generated bit clock in asynchronous mode and that bit clock is generated by an SAI that is disabled in stop mode, then the transmitter or receiver should be disabled by software before entering stop mode. This issue does not apply when the transmitter or receiver is in a synchronous mode because all synchronous SAIs are enabled and disabled simultaneously.

#### 53.4.1.3 Bus clock

The bus clock is used by the control and configuration registers and to generate synchronous interrupts and DMA requests.

## 53.4.2 SAI resets

The SAI is asynchronously reset on system reset. The SAI has a software reset and a FIFO reset.

### 53.4.2.1 Software reset

The SAI transmitter includes a software reset that resets all transmitter internal logic, including the bit clock generation, status flags, and FIFO pointers. It does not reset the configuration registers. The software reset remains asserted until cleared by software.

The SAI receiver includes a software reset that resets all receiver internal logic, including the bit clock generation, status flags and FIFO pointers. It does not reset the configuration registers. The software reset remains asserted until cleared by software.

### 53.4.2.2 FIFO reset

The SAI transmitter includes a FIFO reset that synchronizes the FIFO write pointer to the same value as the FIFO read pointer. This empties the FIFO contents and is to be used after TCSR[FEF] is set, and before the FIFO is re-initialized and TCSR[FEF] is cleared. The FIFO reset is asserted for one cycle only.

The SAI receiver includes a FIFO reset that synchronizes the FIFO read pointer to the same value as the FIFO write pointer. This empties the FIFO contents and is to be used after the RCSR[FEF] is set and any remaining data has been read from the FIFO, and before the RCSR[FEF] is cleared. The FIFO reset is asserted for one cycle only.

## 53.4.3 Synchronous modes

The SAI transmitter and receiver can operate synchronously to each other.

### 53.4.3.1 Synchronous mode

The SAI transmitter and receiver can be configured to operate with synchronous bit clock and frame sync.

If the transmitter bit clock and frame sync are to be used by both the transmitter and receiver:

- The transmitter must be configured for asynchronous operation and the receiver for synchronous operation.
- In synchronous mode, the receiver is enabled only when both the transmitter and receiver are enabled.
- It is recommended that the transmitter is the last enabled and the first disabled.

If the receiver bit clock and frame sync are to be used by both the transmitter and receiver:

- The receiver must be configured for asynchronous operation and the transmitter for synchronous operation.
- In synchronous mode, the transmitter is enabled only when both the receiver and transmitter are both enabled.
- It is recommended that the receiver is the last enabled and the first disabled.

When operating in synchronous mode, only the bit clock, frame sync, and transmitter/receiver enable are shared. The transmitter and receiver otherwise operate independently, although configuration registers must be configured consistently across both the transmitter and receiver.

### 53.4.4 Frame sync configuration

When enabled, the SAI continuously transmits and/or receives frames of data. Each frame consists of a fixed number of words and each word consists of a fixed number of bits. Within each frame, any given word can be masked causing the receiver to ignore that word and the transmitter to tri-state for the duration of that word.

The frame sync signal is used to indicate the start of each frame. A valid frame sync requires a rising edge (if active high) or falling edge (if active low) to be detected and the transmitter or receiver cannot be busy with a previous frame. A valid frame sync is also ignored (slave mode) or not generated (master mode) for the first four bit clock cycles after enabling the transmitter or receiver.

The transmitter and receiver frame sync can be configured independently with any of the following options:

- Externally generated or internally generated
- Active high or active low
- Assert with the first bit in frame or asserts one bit early
- Assert for a duration between 1 bit clock and the first word length
- Frame length from 1 to 32 words per frame
- Word length to support 8 to 32 bits per word

    - First word length and remaining word lengths can be configured separately
- Words can be configured to transmit/receive MSB first or LSB first

These configuration options cannot be changed after the SAI transmitter or receiver is enabled.

## 53.4.5 Data FIFO

Each transmit and receive channel includes a FIFO of size 8 × 32-bit. The FIFO data is accessed using the SAI Transmit/Receive Data Registers.

### 53.4.5.1 Data alignment

Data in the FIFO can be aligned anywhere within the 32-bit wide register through the use of the First Bit Shifted configuration field, which selects the bit index (between 31 and 0) of the first bit shifted.

Examples of supported data alignment and the required First Bit Shifted configuration are illustrated in Figure 53-59 for LSB First configurations and Figure 53-60 for MSB First configurations.

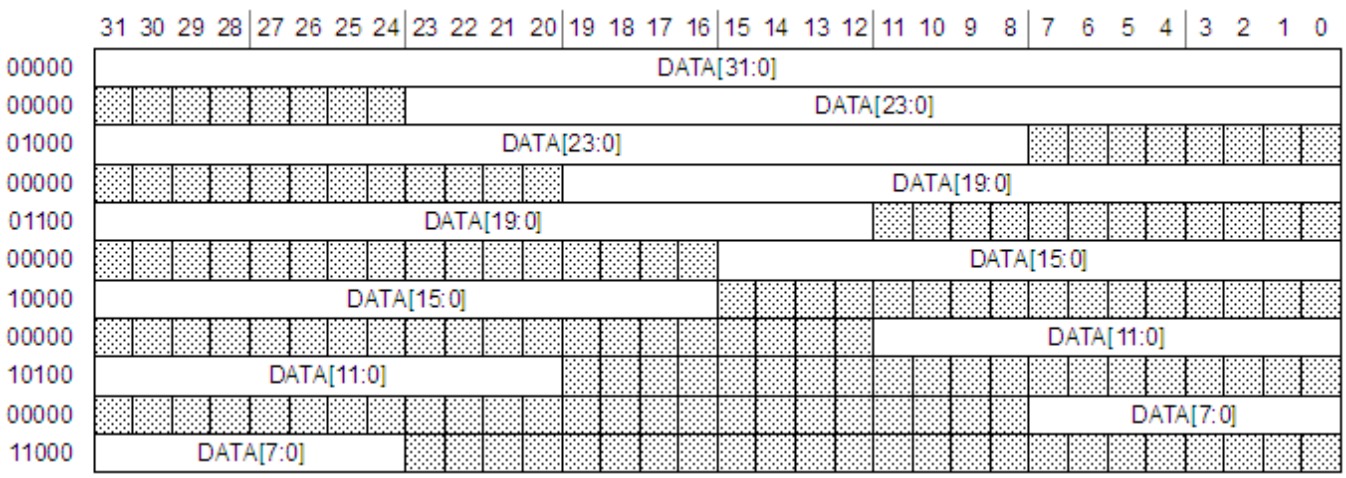



**Figure 53-59. SAI first bit shifted, LSB first**

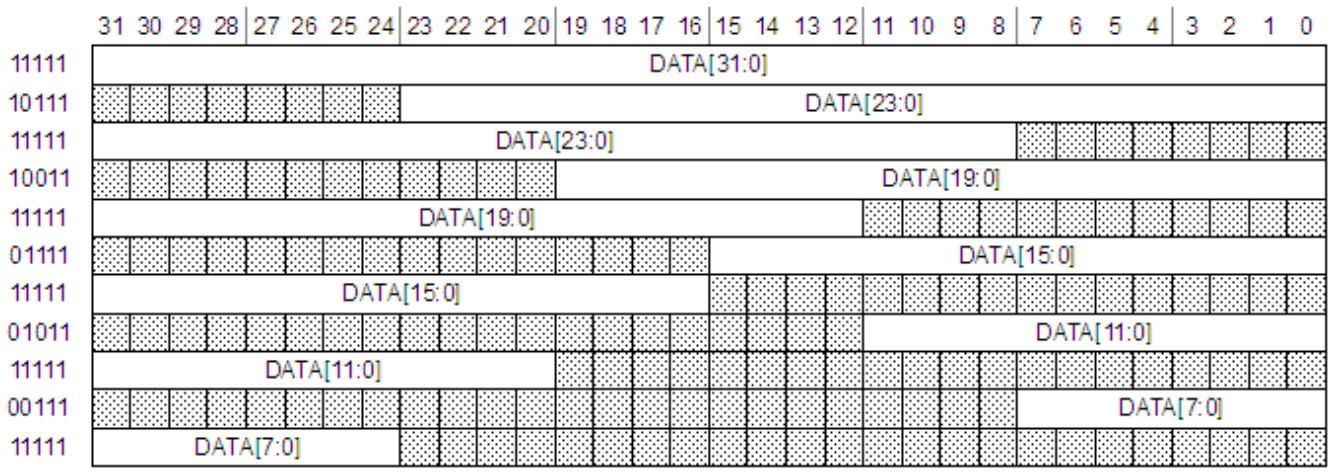



**Figure 53-60. SAI first bit shifted, MSB first**

### 53.4.5.2 FIFO pointers

When writing to a TDR, the WFP of the corresponding TFR increments after each valid write. The SAI supports 8-bit, 16-bit and 32-bit writes to the TDR and the FIFO pointer will increment after each individual write. Note that 8-bit writes should only be used when transmitting up to 8-bit data and 16-bit writes should only be used when transmitting up to 16-bit data.

Writes to a TDR are ignored if the corresponding bit of TCR3[TCE] is clear or if the FIFO is full. If the Transmit FIFO is empty, the TDR must be written at least three bit clocks before the start of the next unmasked word to avoid a FIFO underrun.

When reading an RDR, the RFP of the corresponding RFR increments after each valid read. The SAI supports 8-bit, 16-bit and 32-bit reads from the RDR and the FIFO pointer will increment after each individual read. Note that 8-bit reads should only be used when receiving up to 8-bit data and 16-bit reads should only be used when receiving up to 16-bit data.

Reads from an RDR are ignored if the corresponding bit of RCR3[RCE] is clear or if the FIFO is empty. If the Receive FIFO is full, the RDR must be read at least three bit clocks before the end of an unmasked word to avoid a FIFO overrun.

### 53.4.6 Word mask register

The SAI transmitter and receiver each contain a word mask register, namely TMR and RMR, that can be used to mask any word in the frame. Because the word mask register is double buffered, software can update it before the end of each frame to mask a particular word in the next frame.

The TMR causes the Transmit Data pin to be tri-stated for the length of each selected word and the transmit FIFO is not read for masked words.

The RMR causes the received data for each selected word to be discarded and not written to the receive FIFO.

### 53.4.7 Interrupts and DMA requests

The SAI transmitter and receiver generate separate interrupts and separate DMA requests, but support the same status flags. Asynchronous versions of the transmitter and receiver interrupts are generated to wake up the CPU from stop mode.

#### 53.4.7.1 FIFO request flag

The FIFO request flag is set based on the number of entries in the FIFO and the FIFO watermark configuration.

The transmit FIFO request flag is set when the number of entries in any of the enabled transmit FIFOs is less than or equal to the transmit FIFO watermark configuration and is cleared when the number of entries in each enabled transmit FIFO is greater than the transmit FIFO watermark configuration.

The receive FIFO request flag is set when the number of entries in any of the enabled receive FIFOs is greater than the receive FIFO watermark configuration and is cleared when the number of entries in each enabled receive FIFO is less than or equal to the receive FIFO watermark configuration.

The FIFO request flag can generate an interrupt or a DMA request.

#### 53.4.7.2 FIFO warning flag

The FIFO warning flag is set based on the number of entries in the FIFO.

The transmit warning flag is set when the number of entries in any of the enabled transmit FIFOs is empty and is cleared when the number of entries in each enabled transmit FIFO is not empty.

The receive warning flag is set when the number of entries in any of the enabled receive FIFOs is full and is cleared when the number of entries in each enabled receive FIFO is not full.

The FIFO warning flag can generate an Interrupt or a DMA request.

#### 53.4.7.3 FIFO error flag

The transmit FIFO error flag is set when the any of the enabled transmit FIFOs underflow. After it is set, all enabled transmit channels repeat the last valid word read from the transmit FIFO until TCSR[FEF] is cleared and the next transmit frame starts. All enabled transmit FIFOs must be reset and initialized with new data before TCSR[FEF] is cleared.

RCSR[FEF] is set when the any of the enabled receive FIFOs overflow. After it is set, all enabled receive channels discard received data until RCSR[FEF] is cleared and the next next receive frame starts. All enabled receive FIFOs should be emptied before RCSR[FEF] is cleared.

The FIFO error flag can generate only an interrupt.

#### 53.4.7.4 Sync error flag

The sync error flag, TCSR[SEF] or RCSR[SEF], is set when configured for an externally generated frame sync and the external frame sync asserts when the transmitter or receiver is busy with the previous frame. The external frame sync assertion is ignored and the sync error flag is set. When the sync error flag is set, the transmitter or receiver continues checking for frame sync assertion when idle or at the end of each frame.

The sync error flag can generate an interrupt only.

#### 53.4.7.5 Word start flag

The word start flag is set at the start of the second bit clock for the selected word, as configured by the Word Flag register field.

The word start flag can generate an interrupt only.

# Chapter 54
# General-Purpose Input/Output (GPIO)

## 54.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The general-purpose input and output (GPIO) module communicates to the processor core via a zero wait state interface (IOPORT) for maximum pin performance. The GPIO registers support 8-bit, 16-bit or 32-bit accesses.

The GPIO data direction and output data registers control the direction and output data of each pin when the pin is configured for the GPIO function. The GPIO input data register displays the logic value on each pin when the pin is configured for any digital function, provided the corresponding Port Control and Interrupt module for that pin is enabled.

Efficient bit manipulation of the general-purpose outputs is supported through the addition of set, clear, and toggle write-only registers for each port output data register.

### 54.1.1 Features

- Features of the GPIO module include:
  - Pin input data register visible in all digital pin-multiplexing modes
  - Pin output data register with corresponding set/clear/toggle registers
  - Pin data direction register
  - Zero wait state access to GPIO registers through IOPORT

**NOTE**

GPIO module is clocked by system clock.

## 54.1.2 Modes of operation

The following table depicts different modes of operation and the behavior of the GPIO module in these modes.

**Table 54-1. Modes of operation**

| Modes of operation | Description |
|---|---|
| Run | The GPIO module operates normally. |
| Wait | The GPIO module operates normally. |
| Stop | The GPIO module is disabled. |
| Debug | The GPIO module operates normally. |

## 54.1.3 GPIO signal descriptions

**Table 54-2. GPIO signal descriptions**

| GPIO signal descriptions | Description | I/O |
|---|---|---|
| PORTA31–PORTA0 | General-purpose input/output | I/O |
| PORTB31–PORTB0 | General-purpose input/output | I/O |
| PORTC31–PORTC0 | General-purpose input/output | I/O |
| PORTD31–PORTD0 | General-purpose input/output | I/O |
| PORTE31–PORTE0 | General-purpose input/output | I/O |

**NOTE**

Not all pins within each port are implemented on each device. See the chapter on signal multiplexing for the number of GPIO ports available in the device.

### 54.1.3.1 Detailed signal description

**Table 54-3. GPIO interface-detailed signal descriptions**

| Signal | I/O | Description | |
|---|---|---|---|
| PORTA31–PORTA0<br>PORTB31–PORTB0<br>PORTC31–PORTC0<br>PORTD31–PORTD0<br>PORTE31–PORTE0 | I/O | General-purpose input/output | |
| | | State meaning | Asserted: The pin is logic 1.<br>Deasserted: The pin is logic 0. |
| | | Timing | Assertion: When output, this signal occurs on the rising-edge of the system clock. For input, it may occur at any time and input may be asserted asynchronously to the system clock.<br>Deassertion: When output, this signal occurs on the rising-edge of the system clock. For input, it may occur at any time and input may be asserted asynchronously to the system clock. |

## 54.2 Memory map and register definition

Any read or write access to the GPIO memory space that is outside the valid memory map results in a bus error.

**GPIO memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 400F_F000 | Port Data Output Register (GPIOA_PDOR) | 32 | R/W | 0000_0000h | 54.2.1/1595 |
| 400F_F004 | Port Set Output Register (GPIOA_PSOR) | 32 | W (always reads 0) | 0000_0000h | 54.2.2/1595 |
| 400F_F008 | Port Clear Output Register (GPIOA_PCOR) | 32 | W (always reads 0) | 0000_0000h | 54.2.3/1596 |
| 400F_F00C | Port Toggle Output Register (GPIOA_PTOR) | 32 | W (always reads 0) | 0000_0000h | 54.2.4/1596 |
| 400F_F010 | Port Data Input Register (GPIOA_PDIR) | 32 | R | 0000_0000h | 54.2.5/1597 |
| 400F_F014 | Port Data Direction Register (GPIOA_PDDR) | 32 | R/W | 0000_0000h | 54.2.6/1597 |
| 400F_F040 | Port Data Output Register (GPIOB_PDOR) | 32 | R/W | 0000_0000h | 54.2.1/1595 |

*Table continues on the next page...*

## GPIO memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 400F_F044 | Port Set Output Register (GPIOB_PSOR) | 32 | W (always reads 0) | 0000_0000h | 54.2.2/1595 |
| 400F_F048 | Port Clear Output Register (GPIOB_PCOR) | 32 | W (always reads 0) | 0000_0000h | 54.2.3/1596 |
| 400F_F04C | Port Toggle Output Register (GPIOB_PTOR) | 32 | W (always reads 0) | 0000_0000h | 54.2.4/1596 |
| 400F_F050 | Port Data Input Register (GPIOB_PDIR) | 32 | R | 0000_0000h | 54.2.5/1597 |
| 400F_F054 | Port Data Direction Register (GPIOB_PDDR) | 32 | R/W | 0000_0000h | 54.2.6/1597 |
| 400F_F080 | Port Data Output Register (GPIOC_PDOR) | 32 | R/W | 0000_0000h | 54.2.1/1595 |
| 400F_F084 | Port Set Output Register (GPIOC_PSOR) | 32 | W (always reads 0) | 0000_0000h | 54.2.2/1595 |
| 400F_F088 | Port Clear Output Register (GPIOC_PCOR) | 32 | W (always reads 0) | 0000_0000h | 54.2.3/1596 |
| 400F_F08C | Port Toggle Output Register (GPIOC_PTOR) | 32 | W (always reads 0) | 0000_0000h | 54.2.4/1596 |
| 400F_F090 | Port Data Input Register (GPIOC_PDIR) | 32 | R | 0000_0000h | 54.2.5/1597 |
| 400F_F094 | Port Data Direction Register (GPIOC_PDDR) | 32 | R/W | 0000_0000h | 54.2.6/1597 |
| 400F_F0C0 | Port Data Output Register (GPIOD_PDOR) | 32 | R/W | 0000_0000h | 54.2.1/1595 |
| 400F_F0C4 | Port Set Output Register (GPIOD_PSOR) | 32 | W (always reads 0) | 0000_0000h | 54.2.2/1595 |
| 400F_F0C8 | Port Clear Output Register (GPIOD_PCOR) | 32 | W (always reads 0) | 0000_0000h | 54.2.3/1596 |
| 400F_F0CC | Port Toggle Output Register (GPIOD_PTOR) | 32 | W (always reads 0) | 0000_0000h | 54.2.4/1596 |
| 400F_F0D0 | Port Data Input Register (GPIOD_PDIR) | 32 | R | 0000_0000h | 54.2.5/1597 |
| 400F_F0D4 | Port Data Direction Register (GPIOD_PDDR) | 32 | R/W | 0000_0000h | 54.2.6/1597 |
| 400F_F100 | Port Data Output Register (GPIOE_PDOR) | 32 | R/W | 0000_0000h | 54.2.1/1595 |
| 400F_F104 | Port Set Output Register (GPIOE_PSOR) | 32 | W (always reads 0) | 0000_0000h | 54.2.2/1595 |
| 400F_F108 | Port Clear Output Register (GPIOE_PCOR) | 32 | W (always reads 0) | 0000_0000h | 54.2.3/1596 |
| 400F_F10C | Port Toggle Output Register (GPIOE_PTOR) | 32 | W (always reads 0) | 0000_0000h | 54.2.4/1596 |

*Table continues on the next page...*

**GPIO memory map (continued)**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 400F_F110 | Port Data Input Register (GPIOE_PDIR) | 32 | R | 0000_0000h | 54.2.5/1597 |
| 400F_F114 | Port Data Direction Register (GPIOE_PDDR) | 32 | R/W | 0000_0000h | 54.2.6/1597 |

## 54.2.1 Port Data Output Register (GPIOx_PDOR)

This register configures the logic levels that are driven on each general-purpose output pins.

**NOTE**

Do not modify pin configuration registers associated with pins not available in your selected package. All un-bonded pins not available in your package will default to DISABLE state for lowest power consumption.

Address: Base address + 0h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PDO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**GPIOx_PDOR field descriptions**

| Field | Description |
|---|---|
| 31–0<br>PDO | Port Data Output<br><br>Register bits for un-bonded pins return a undefined value when read.<br><br>0 Logic level 0 is driven on pin, provided pin is configured for general-purpose output.<br>1 Logic level 1 is driven on pin, provided pin is configured for general-purpose output. |

## 54.2.2 Port Set Output Register (GPIOx_PSOR)

This register configures whether to set the fields of the PDOR.

Address: Base address + 4h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | PTSO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**GPIOx_PSOR field descriptions**

| Field | Description |
|---|---|
| 31–0<br>PTSO | Port Set Output<br><br>Writing to this register will update the contents of the corresponding bit in the PDOR as follows:<br><br>0 Corresponding bit in PDORn does not change.<br>1 Corresponding bit in PDORn is set to logic 1. |

## 54.2.3 Port Clear Output Register (GPIOx_PCOR)

This register configures whether to clear the fields of PDOR.

Address: Base address + 8h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | PTCO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**GPIOx_PCOR field descriptions**

| Field | Description |
|---|---|
| 31–0<br>PTCO | Port Clear Output<br><br>Writing to this register will update the contents of the corresponding bit in the Port Data Output Register (PDOR) as follows:<br><br>0 Corresponding bit in PDORn does not change.<br>1 Corresponding bit in PDORn is cleared to logic 0. |

## 54.2.4 Port Toggle Output Register (GPIOx_PTOR)

Address: Base address + Ch offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | PTTO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**GPIOx_PTOR field descriptions**

| Field | Description |
|---|---|
| 31–0<br>PTTO | Port Toggle Output<br><br>Writing to this register will update the contents of the corresponding bit in the PDOR as follows: |

**GPIOx_PTOR field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Corresponding bit in PDORn does not change.<br>1 Corresponding bit in PDORn is set to the inverse of its existing logic state. |

## 54.2.5 Port Data Input Register (GPIOx_PDIR)

**NOTE**

Do not modify pin configuration registers associated with pins not available in your selected package. All un-bonded pins not available in your package will default to DISABLE state for lowest power consumption.

Address: Base address + 10h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PDI | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**GPIOx_PDIR field descriptions**

| Field | Description |
|---|---|
| 31–0<br>PDI | Port Data Input<br><br>Reads 0 at the unimplemented pins for a particular device. Pins that are not configured for a digital function read 0. If the Port Control and Interrupt module is disabled, then the corresponding bit in PDIR does not update.<br><br>0 Pin logic level is logic 0, or is not configured for use by digital function.<br>1 Pin logic level is logic 1. |

## 54.2.6 Port Data Direction Register (GPIOx_PDDR)

The PDDR configures the individual port pins for input or output.

Address: Base address + 14h offset

| Bit | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PDD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**GPIOx_PDDR field descriptions**

| Field | Description |
|---|---|
| 31–0<br>PDD | Port Data Direction<br><br>Configures individual port pins for input or output.<br><br>0 Pin is configured as general-purpose input, for the GPIO function.<br>1 Pin is configured as general-purpose output, for the GPIO function. |

## 54.3 Functional description

### 54.3.1 General-purpose input

The logic state of each pin is available via the Port Data Input registers, provided the pin is configured for a digital function and the corresponding Port Control and Interrupt module is enabled.

The Port Data Input registers return the synchronized pin state after any enabled digital filter in the Port Control and Interrupt module. The input pin synchronizers are shared with the Port Control and Interrupt module, so that if the corresponding Port Control and Interrupt module is disabled, then synchronizers are also disabled. This reduces power consumption when a port is not required for general-purpose input functionality.

### 54.3.2 General-purpose output

The logic state of each pin can be controlled via the port data output registers and port data direction registers, provided the pin is configured for the GPIO function. The following table depicts the conditions for a pin to be configured as input/output.

| If | Then |
|---|---|
| A pin is configured for the GPIO function and the corresponding port data direction register bit is clear. | The pin is configured as an input. |
| A pin is configured for the GPIO function and the corresponding port data direction register bit is set. | The pin is configured as an output and and the logic state of the pin is equal to the corresponding port data output register. |

To facilitate efficient bit manipulation on the general-purpose outputs, pin data set, pin data clear, and pin data toggle registers exist to allow one or more outputs within one port to be set, cleared, or toggled from a single register write.

The corresponding Port Control and Interrupt module does not need to be enabled to update the state of the port data direction registers and port data output registers including the set/clear/toggle registers.

# Chapter 55
# JTAG Controller (JTAGC)

## 55.1 Introduction

**NOTE**

For the chip-specific implementation details of this module's instances see the chip configuration information.

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format.

### 55.1.1 Block diagram

The following is a simplified block diagram of the JTAG Controller (JTAGC) block. Refer to the chip-specific configuration information as well as Register description for more information about the JTAGC registers.

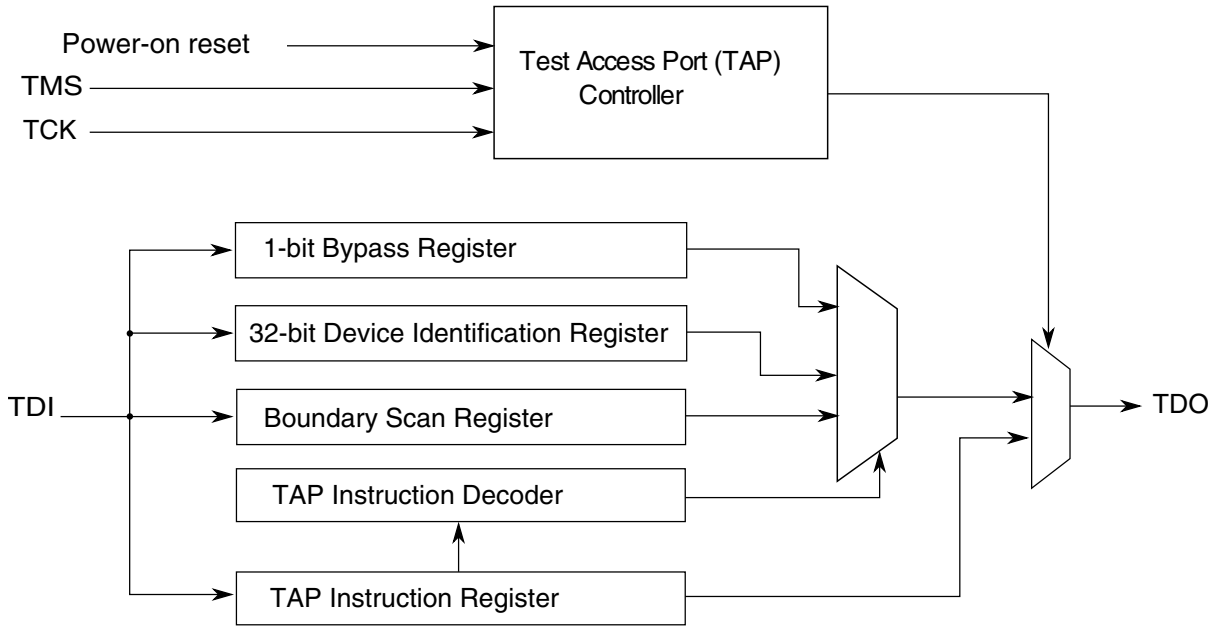



**Figure 55-1. JTAG (IEEE 1149.1) block diagram**

## 55.1.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
  - 4 pins (TDI, TMS, TCK, and TDO)
- Instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to Table 55-3 for a list of supported instructions.
- Bypass register, boundary scan register, and device identification register.
- TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

## 55.1.3 Modes of operation

The JTAGC block uses a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

#### 55.1.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, or the TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered
- The instruction register is loaded with the IDCODE instruction

#### 55.1.3.2 IEEE 1149.1-2001 defined test modes

The JTAGC block supports several IEEE 1149.1-2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, HIGHZ, CLAMP, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the BYPASS, HIGHZ, CLAMP or reserved instructions are active. The functionality of each test mode is explained in more detail in JTAGC block instructions.

#### 55.1.3.3 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC block into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

## 55.2 External signal description

The JTAGC consists of a set of signals that connect to off chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

**Table 55-1. JTAG signal properties**

| Name | I/O | Function | Reset State | Pull |
|---|---|---|---|---|
| TCK | Input | Test Clock | — | Down |
| TDI | Input | Test Data In | — | Up |
| TDO | Output | Test Data Out | High Z[1] | — |
| TMS | Input | Test Mode Select | — | Up |

1. TDO output buffer enable is negated when the JTAGC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented at the TDO pad for use when JTAGC is inactive.

### 55.2.1 TCK—Test clock input

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

### 55.2.2 TDI—Test data input

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

### 55.2.3 TDO—Test data output

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is three-stateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in TAP controller state machine.

### 55.2.4 TMS—Test mode select

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

## 55.3 Register description

This section provides a detailed description of the JTAGC block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

### 55.3.1 Instruction register

The JTAGC block uses a 4-bit instruction register as shown in the following figure. The instruction register allows instructions to be loaded into the block to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 0001b , making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

| | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| R | 0 | 0 | 0 | 1 |
| W | Instruction Code | | | |
| Reset: | 0 | 0 | 0 | 1 |

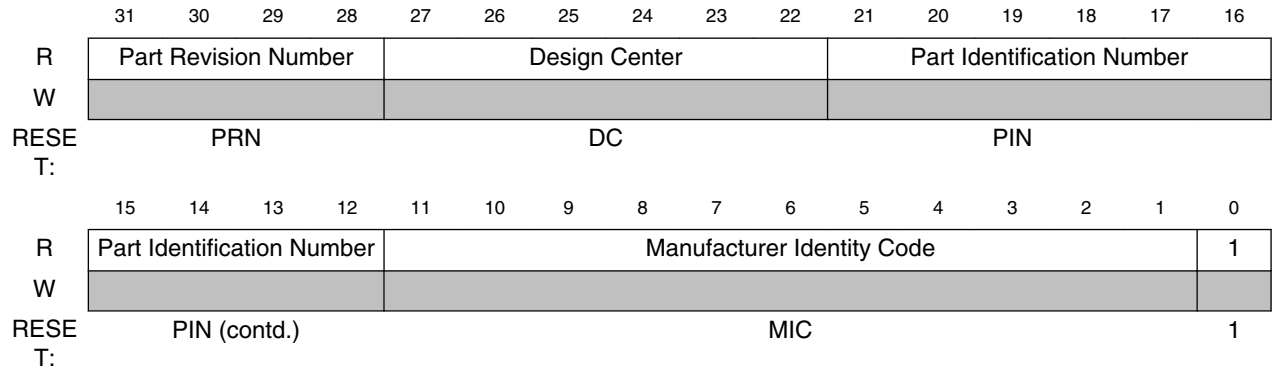
**Figure 55-2. Instruction register**

### 55.3.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS, CLAMP, HIGHZ or reserve instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

## 55.3.3 Device identification register

The device identification (JTAG ID) register, shown in the following figure, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the Capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.




The following table describes the device identification register functions.

**Table 55-2. Device identification register field descriptions**

| Field | Description |
|---|---|
| PRN | Part Revision Number. Contains the revision number of the part. Value is 0x0. |
| DC | Design Center. Indicates the design center. Value is 0x2C. |
| PIN | Part Identification Number. Contains the part number of the device. Value is TBD. |
| MIC | Manufacturer Identity Code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID. Value is 0x00E . |
| IDCODE ID | IDCODE Register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1. |

## 55.3.4 Boundary scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in Boundary scan. The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

## 55.4 Functional description

This section explains the JTAGC functional description.

### 55.4.1 JTAGC reset configuration

While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

### 55.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the currently loaded instruction.

Data is shifted between TDI and TDO though the selected register starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register, test data registers, and the bypass register.

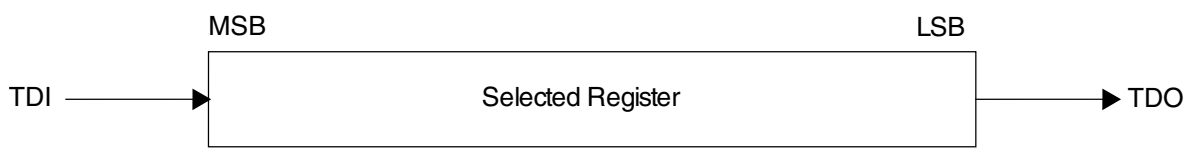
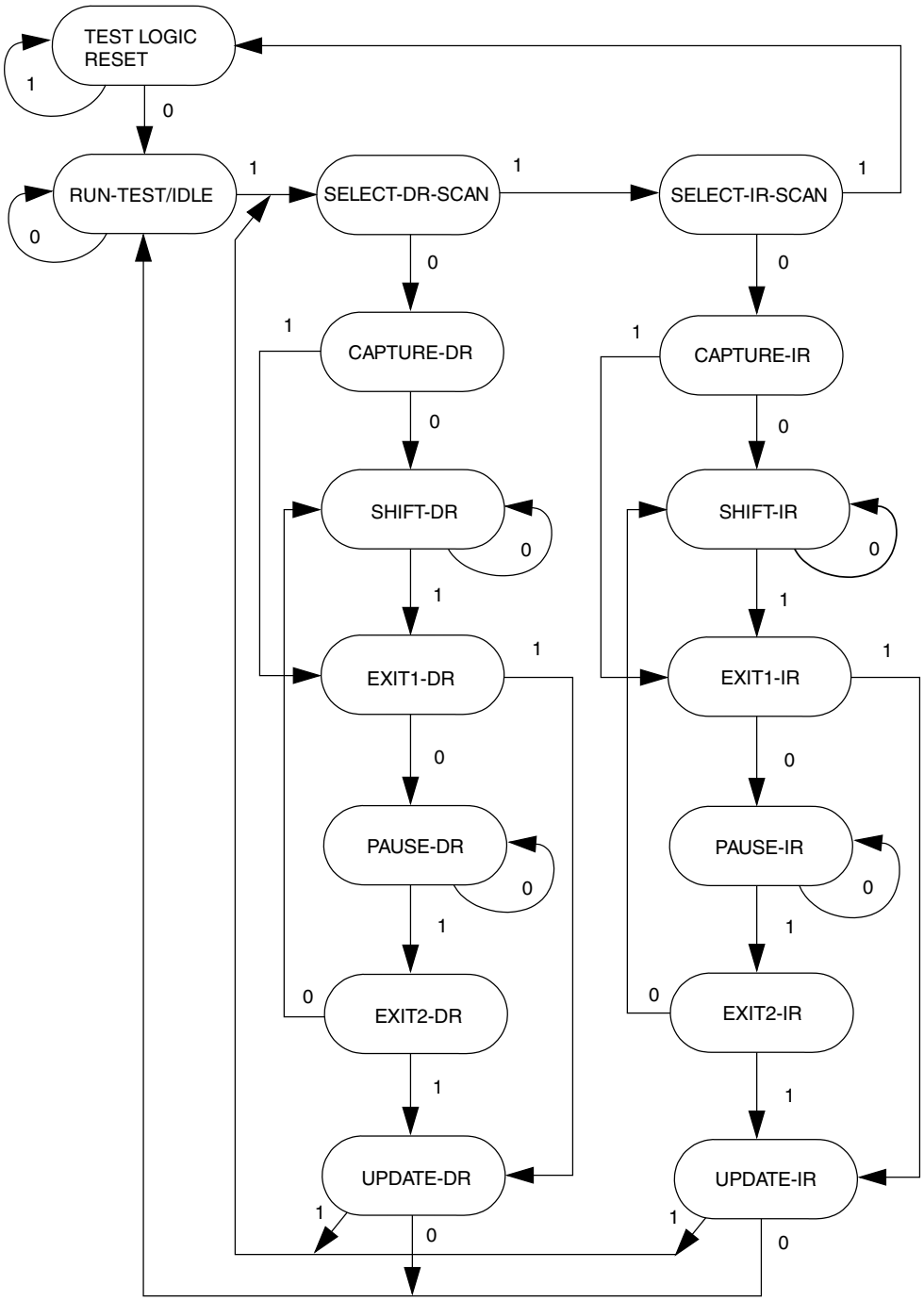



**Figure 55-3. Shifting data through a register**

### 55.4.3 TAP controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. The following figure shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As the following figure shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.




The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

**Figure 55-4. IEEE 1149.1-2001 TAP controller finite state machine**

### 55.4.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting the JTAGC enable to a logic 1 value.

#### 55.4.3.2 Selecting an IEEE 1149.1-2001 register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC block instructions while the JTAGC is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

### 55.4.4 JTAGC block instructions

The JTAGC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

**Table 55-3. 4-bit JTAG instructions**

| Instruction | Code[3:0] | Instruction summary |
|---|---|---|
| IDCODE | 0000 | Selects device identification register for shift |
| SAMPLE/PRELOAD | 0010 | Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation |
| SAMPLE | 0011 | Selects boundary scan register for shifting and sampling without disturbing functional operation |
| EXTEST | 0100 | Selects boundary scan register and applies preloaded values to output pins.<br>**NOTE:** Execution of this instruction asserts functional reset. |
| Factory debug reserved | 0101 | Intended for factory debug only |
| Factory debug reserved | 0110 | Intended for factory debug only |
| Factory debug reserved | 0111 | Intended for factory debug only |
| ARM JTAG-DP Reserved | 1000 | This instruction goes the ARM JTAG-DP controller. See the ARM JTAG-DP documentation for more information. |
| HIGHZ | 1001 | Selects bypass register and three-states all output pins.<br>**NOTE:** Execution of this instruction asserts functional reset. |
| ARM JTAG-DP Reserved | 1010 | This instruction goes the ARM JTAG-DP controller. See the ARM JTAG-DP documentation for more information. |

*Table continues on the next page...*

**Table 55-3. 4-bit JTAG instructions (continued)**

| Instruction | Code[3:0] | Instruction summary |
|---|---|---|
| ARM JTAG-DP Reserved | 1011 | This instruction goes the ARM JTAG-DP controller. See the ARM JTAG-DP documentation for more information. |
| CLAMP | 1100 | Selects bypass register and applies preloaded values to output pins.<br>**NOTE:** Execution of this instruction asserts functional reset. |
| EZPORT | 1101 | Enables the EZPORT function for the SoC |
| ARM JTAG-DP Reserved | 1110 | This instruction goes the ARM JTAG-DP controller. See the ARM JTAG-DP documentation for more information. |
| BYPASS | 1111 | Selects bypass register for data operations |

### 55.4.4.1 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC block is reset.

### 55.4.4.2 SAMPLE/PRELOAD instruction

The SAMPLE/PRELOAD instruction has two functions:

- The SAMPLE portion of the instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- The PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the Shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

#### 55.4.4.3 SAMPLE instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

#### 55.4.4.4 EXTEST External test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

#### 55.4.4.5 HIGHZ instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. While HIGHZ is active all output drivers are placed in an inactive drive state (e.g., high impedance). HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

#### 55.4.4.6 CLAMP instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

#### 55.4.4.7 BYPASS instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

### 55.4.5 Boundary scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

## 55.5 Initialization/Application information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC block and enable access to registers, the following sequence is required:

1. Place the JTAGC in reset through TAP controller state machine transitions controlled by TMS
2. Load the appropriate instruction for the test or action to be performed

# Appendix A
# Release Notes for Revision 4

## A.1 General changes throughout document

- Removed LON registers from UART chapter
- Removed EARS register from eDMA chapter
- Removed LPOPO bit from SMC_VLLCTRL register in the SMC chapter.
- Removed C9 and C10 registers from the MCG chapter

## A.2 About This Document chapter changes

- No substantial content changes

## A.3 Introduction chapter changes

- No substantial content changes

## A.4 Chip Configuration chapter changes

- Added the subsection "Writing SPI Transmit FIFO" to the SPI configuration section

## A.5 Memory Map chapter changes

- No substantial content changes

## A.6 Clock Distribution chapter changes

- No substantial content changes

## A.7 Reset and Boot chapter changes

- No substantial content changes

## A.8 Power Management chapter changes

- No substantial content changes

## A.9 Security chapter changes

- No substantial content changes

## A.10 Debug chapter changes

- No substantial content changes

## A.11 Signal Multiplexing and Signal Descriptions chapter changes

- No substantial content changes

## A.12 PORT changes

- No substantial content changes

## A.13 SIM changes

- No substantial content changes

## A.14 RCM changes

- No substantial content changes

## A.15 SMC changes

- No substantial content changes

## A.16 PMC changes

- No substantial content changes

## A.17 LLWU changes

- No substantial content changes

## A.18 MCM changes

- No substantial content changes

## A.19 Crossbar switch module changes

- No substantial content changes

## A.20 MPU changes

- No substantial content changes

## A.21 AIPS-Lite changes

- No substantial content changes

## A.22 DMAMUX module changes

- No substantial content changes

## A.23 eDMA module changes

- No substantial content changes

## A.24 EWM changes

- No substantial content changes

## A.25 WDOG changes

- No substantial content changes

## A.26 MCG changes

- No substantial content changes

## A.27 OSC changes

- No substantial content changes

## A.28 RTC Oscillator changes

- No substantial content changes

## A.29 FMC changes

- No substantial content changes

## A.30 FTFE changes

- No substantial content changes

## A.31 EzPort changes

- No substantial content changes

## A.32 FlexBus changes

- No substantial content changes

## A.33 CRC changes

- No substantial content changes

## A.34 MMCAU changes

- No substantial content changes

## A.35 RNGA chapter changes

- No substantial content changes

## A.36 ADC changes

- No substantial content changes

## A.37 CMP changes

- No substantial content changes

## A.38 DAC changes

- No substantial content changes

## A.39 VREF changes

- No substantial content changes

## A.40 PDB changes

- No substantial content changes

## A.41 FTM changes

- No substantial content changes

## A.42 PIT module changes

- No substantial content changes

## A.43 LPTMR changes

- No substantial content changes

## A.44 CMT changes

- No substantial content changes

## A.45 RTC changes

- No substantial content changes

## A.46 USB changes

- No substantial content changes

## A.47 USBDCD changes

- No substantial content changes

## A.48 USB VREG changes

- No substantial content changes

## A.49 FlexCAN module changes

- No substantial content changes

## A.50 DSPI chapter changes

- No substantial content changes

## A.51 I2C changes

- No substantial content changes

## A.52 UART changes

- No substantial content changes

## A.53 SDHC changes

- No substantial content changes

## A.54 I2S/SAI changes

- No substantial content changes

## A.55 GPIO changes

- No substantial content changes

## A.56 JTAGC module changes

- No substantial content changes

***How to Reach Us:***

**Home Page:**
freescale.com

**Web Support:**
freescale.com/support



Document Number K21P144M120SF5RM
Revision 4, November 2014