KL82 Sub-Family Reference Manual

Supports: MKL82Z128VMC7(R), MKL82Z128VLK7(R), MKL82Z128VLL7(R), MKL82Z128VLH7(R), MKL82Z128VMP7(R)

Document Number: KL82P121M72SF0RM
Rev. 3, 08/2016

Contents

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

Chapter 1
About This Manual

Chapter 2
Introduction

Chapter 3

Core Overview

Chapter 4
Memories and Memory Interfaces

Chapter 5
Clock Distribution using MCG

Chapter 6
Reset and Boot

Chapter 7
Kinetis ROM Bootloader

Chapter 8
Power Management

Chapter 12
Port Control and Interrupts (PORT)

Chapter 13

System Integration Module (SIM)

Chapter 14

EMV SIM

Chapter 15
Reset Control Module (RCM)

Chapter 16
System Mode Controller (SMC)

Chapter 17
Power Management Controller (PMC)

Chapter 18
Interrupt Multiplexer (INTMUX)

Chapter 19
Low-Leakage Wakeup Unit (LLWU)

Chapter 20
Miscellaneous Control Module (MCM)

Chapter 21
Crossbar-Lite Switch (AXBS-Lite)

Chapter 22
Peripheral Bridge (AIPS-Lite)

Chapter 23
Memory Protection Unit (MPU)

Chapter 24
Bit Manipulation Engine (BME)

Chapter 25
Direct Memory Access Multiplexer (DMAMUX)

Chapter 26
Enhanced Direct Memory Access (eDMA)

Chapter 27
External Watchdog Monitor (EWM)

Chapter 28
Watchdog Timer (WDOG)

Chapter 29
Multipurpose Clock Generator (MCG)

Chapter 30
Oscillator (OSC)

Chapter 31
RTC Oscillator (OSC32K)

Chapter 32
Micro Trace Buffer (MTB)

Chapter 33
Flash Memory Controller (FMC)

Chapter 34
Flash Memory Module (FTFA)

Chapter 35
Quad Serial Peripheral Interface (QuadSPI)

Chapter 36
Cyclic Redundancy Check (CRC)

Chapter 37
True Random Number Generator

Chapter 38
LP Trusted Cryptography (LTC)

Chapter 39
Analog-to-Digital Converter (ADC)

Chapter 40
Comparator (CMP)

Chapter 41
Voltage Reference (VREFV1)

Chapter 42
12-bit Digital-to-Analog Converter (DAC)

Chapter 43
Timer/PWM Module (TPM)

Chapter 44
Low-Power Timer (LPTMR)

Chapter 45
Periodic Interrupt Timer (PIT)

Chapter 46
Real Time Clock (RTC)

Chapter 47
Universal Serial Bus Full Speed OTG Controller (USBFSOTG)

Chapter 48
Serial Peripheral Interface (SPI)

Chapter 49
Inter-Integrated Circuit (I2C)

Chapter 50
Low Power Universal asynchronous receiver/transmitter (LPUART)

Chapter 51
FlexIO

Chapter 52
Touch Sensing Input (TSI)

Chapter 53
General-Purpose Input/Output (GPIO)

Chapter 1
About This Manual

1.1 Audience

This reference manual is intended for system software and hardware developers and applications programmers who want to develop products with this device. It assumes that the reader understands operating systems, microprocessor system design, and basic principles of software and hardware.

1.2 Organization

This manual has two main sets of chapters.

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

1.3 Module descriptions

Each module chapter has two main parts:

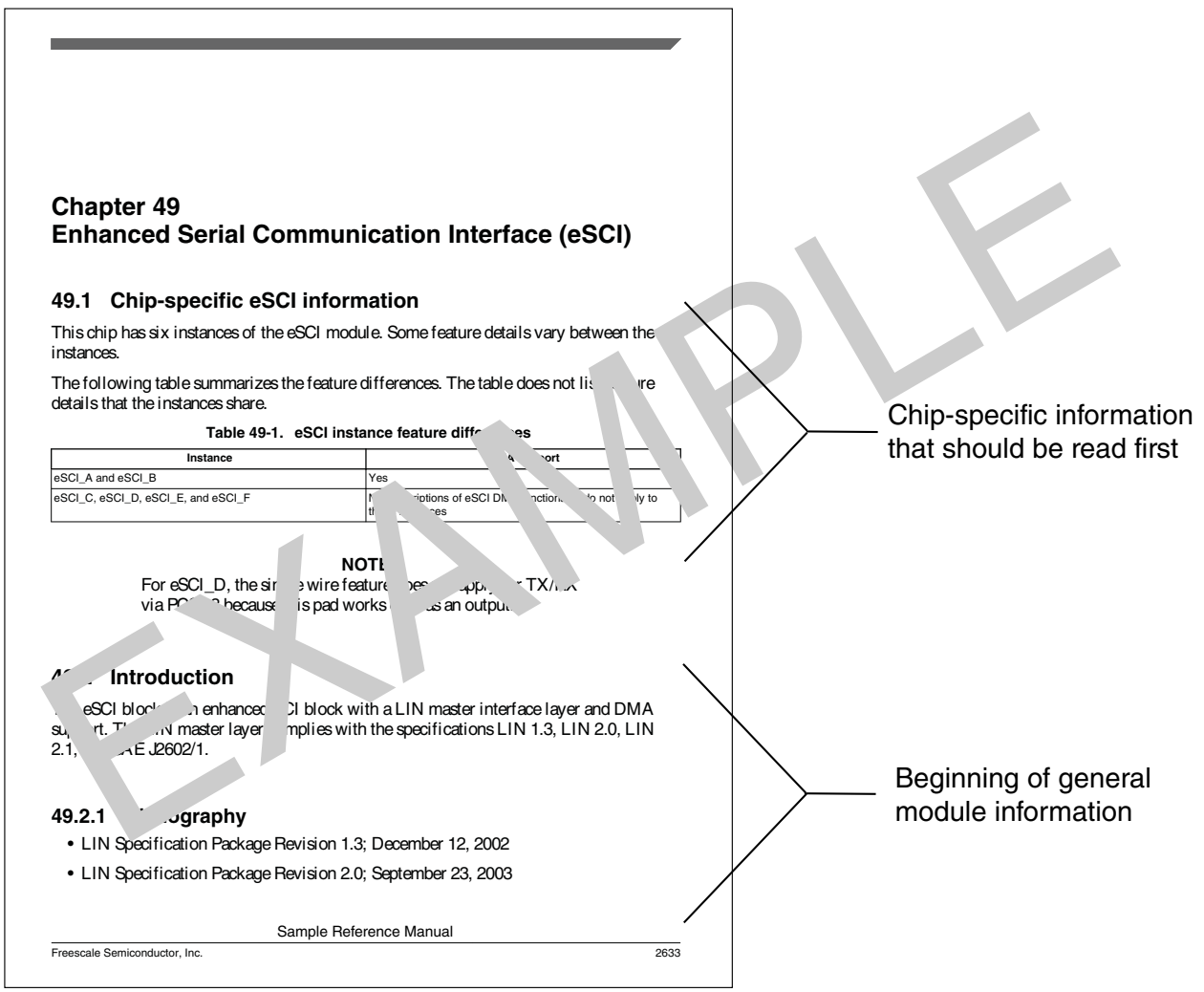
- **Chip-specific:** The first section, *Chip-specific [module name] information*, includes the number of module instances on the chip and possible implementation differences between the module instances, such as differences in FIFO depths or the number of

channels supported. It may also include functional connections between the module instances and other modules. Read this section *first* because its content is crucial to understanding the information in other sections of the chapter.

- **General:** The subsequent sections provide general information about the module, including its signals, registers, and functional description.

NOTE

If there is a conflict between the chip-specific module information (first section) and the general module information (subsequent sections), the chip-specific information supersedes the general information.



Chapter 49
Enhanced Serial Communication Interface (eSCI)

49.1 Chip-specific eSCI information

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

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

Table 49-1. eSCI instance feature differences

Instance	Support
eSCI_A and eSCI_B	Yes
eSCI_C, eSCI_D, eSCI_E, and eSCI_F	No descriptions of eSCI DMA functions apply to these instances

NOTE
For eSCI_D, the single wire feature does support TX/RX via PC[3] because this pad works as an output

Introduction

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

49.2.1 Bibliography

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

Sample Reference Manual

Freescale Semiconductor, Inc. 2633

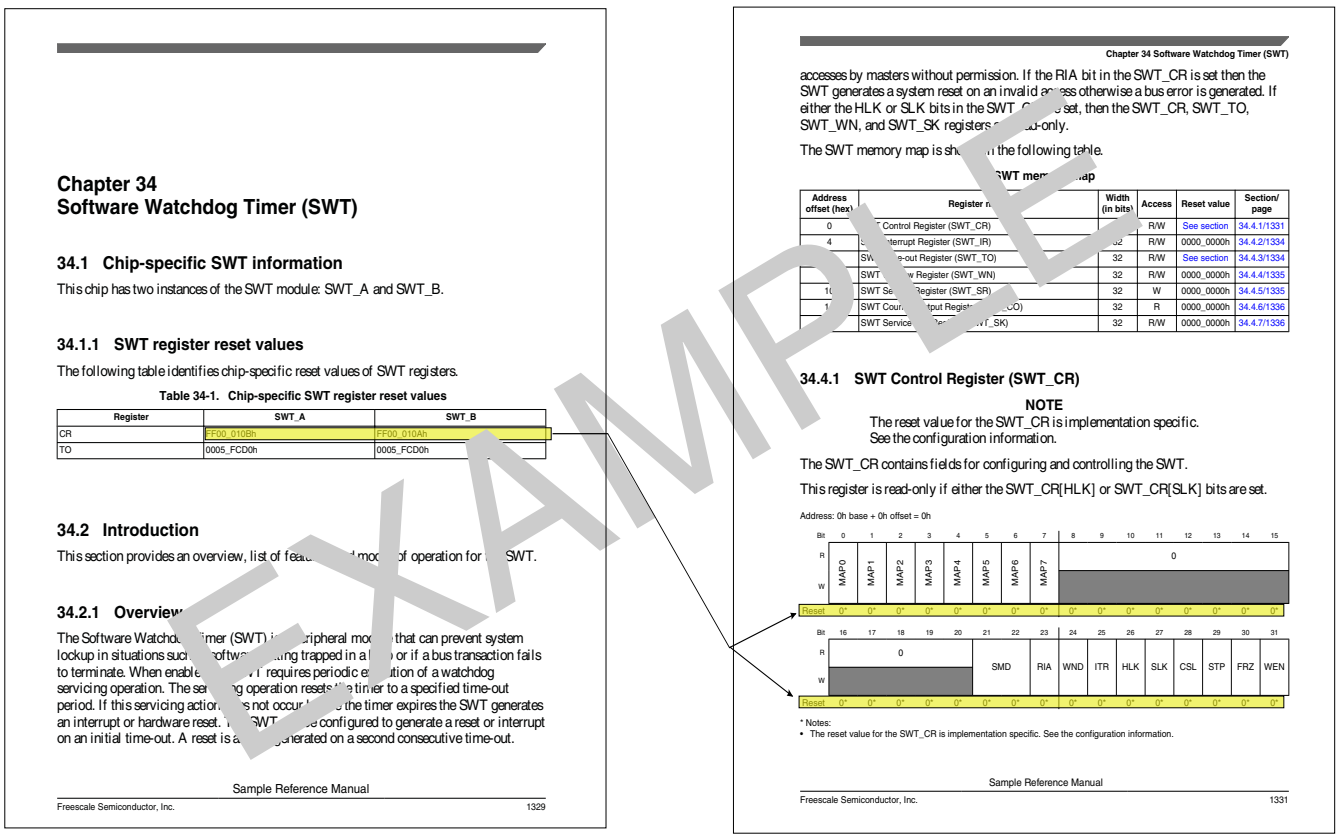
Chip-specific information that should be read first

Beginning of general module information

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

1.3.1 Example: chip-specific information that supersedes content in the same chapter

The example below shows chip-specific information that supersedes general module information presented later in the chapter. In this case, the chip-specific register reset values supersede the reset values that appear in the register diagram.



Chapter 34
Software Watchdog Timer (SWT)

34.1 Chip-specific SWT information

This chip has two instances of the SWT module: SWT_A and SWT_B.

34.1.1 SWT register reset values

The following table identifies chip-specific reset values of SWT registers.

Table 34-1. Chip-specific SWT register reset values

Register	SWT_A	SWT_B
CR	FF00_010Bh	FF00_010Ah
TO	0005_FCD0h	0005_FCD0h

34.2 Introduction

This section provides an overview, list of feat... mod... of operation for ... SWT.

34.2.1 Overview

The Software Watchd... imer (SWT) i... ripheral mo... that can prevent system lockup in situations suc... softw... ting trapped in a l... or if a bus transaction fails to terminate. When enable... T requires periodic e... tion of a watchdog servicing operation. The se... g operation resets ... timer to a specified time-out period. If this servicing actio... s not occur ... the timer expires the SWT generates an interrupt or hardware reset. ... SWT ... e configured to generate a reset or interrupt on an initial time-out. A reset is a... enerated on a second consecutive time-out.

Sample Reference Manual

Freescale Semiconductor, Inc. 1329

Chapter 34 Software Watchdog Timer (SWT)

accesses by masters without permission. If the RIA bit in the SWT_CR is set then the SWT generates a system reset on an invalid a... ss otherwise a bus error is generated. If either the HLK or SLK bits in the SWT_C... set, then the SWT_CR, SWT_TO, SWT_WN, and SWT_SK registers ... -only.

The SWT memory map is sh... n the following table.

SWT mem... ap

Address offset (hex)	Register n...	Width (in bits)	Access	Reset value	Section/ page
0	... Control Register (SWT_CR)		R/W	See section	34.4.1/1331
4	... terrupt Register (SWT_IR)	...2	R/W	0000_0000h	34.4.2/1334
	SW... e-out Register (SWT_TO)	32	R/W	See section	34.4.3/1334
	SWT ... w Register (SWT_WN)	32	R/W	0000_0000h	34.4.4/1335
10	SWT Se... egister (SWT_SR)	32	W	0000_0000h	34.4.5/1335
1...	SWT Coun... tput Regis... CO)	32	R	0000_0000h	34.4.6/1336
	SWT Service ... T_SK)	32	R/W	0000_0000h	34.4.7/1336

34.4.1 SWT Control Register (SWT_CR)

NOTE
The reset value for the SWT_CR is implementation specific.
See the configuration information.

The SWT_CR contains fields for configuring and controlling the SWT.

This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MAP0	MAP1	MAP2	MAP3	MAP4	MAP5	MAP6	MAP7	0							
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SMD	RIA	WND	ITR	HLK	SLK	CSL	STP	FRZ	WEN	
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value for the SWT_CR is implementation specific. See the configuration information.

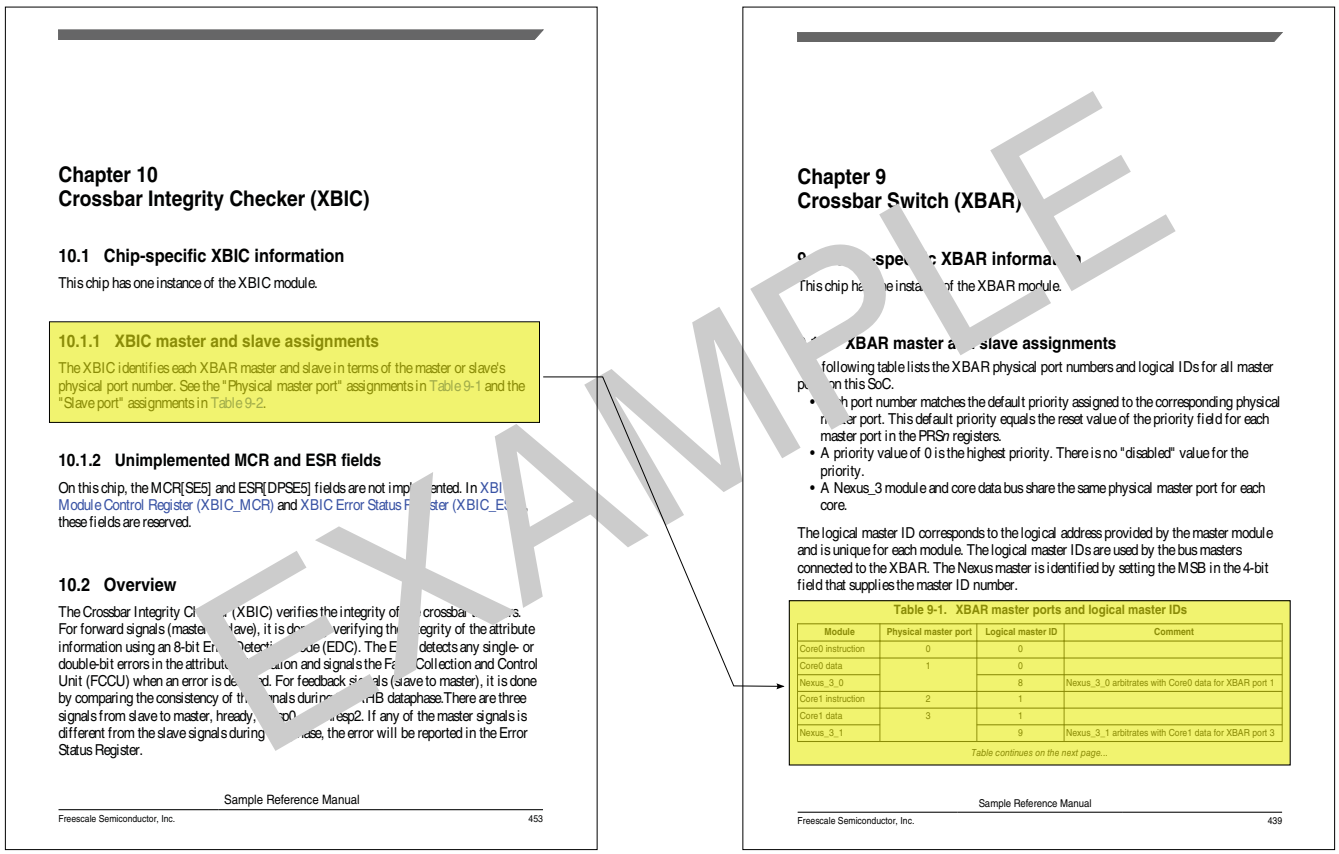
Sample Reference Manual

Freescale Semiconductor, Inc. 1331

Figure 1-2. Example: chip-specific information that supersedes content in the same chapter

1.3.2 Example: chip-specific information that refers to a different chapter

The chip-specific information below refers to another chapter's chip-specific information. In this case, read both sets of chip-specific information before reading further in the chapter.



Chapter 10
Crossbar Integrity Checker (XBIC)

10.1 Chip-specific XBIC information

This chip has one instance of the XBIC module.

10.1.1 XBIC master and slave assignments

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

10.1.2 Unimplemented MCR and ESR fields

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

10.2 Overview

The Crossbar Integrity Ch... (XBIC) verifies the integrity of ... crossbar ... For forward signals (maste... lave), it is do... verifying th... egrity of the attribute information using an 8-bit Er... Detecti... ue (EDC). The E... detects any single- or double-bit errors in the attribut... ation and signals the Fa... Collection and Control Unit (FCCU) when an error is de... d. For feedback si... ls (slave to master), it is done by comparing the consistency of th... nals during ... AHB dataphase.There are three signals from slave to master, hready, hr... resp2. If any of the master signals is different from the slave signals during ... ase, the error will be reported in the Error Status Register.

Sample Reference Manual
Freescale Semiconductor, Inc. 453

Chapter 9
Crossbar Switch (XBAR)

9 ...-spe... : XBAR informa...

This chip ha... e insta... of the XBAR module.

... XBAR master a... slave assignments

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

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

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

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

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

Table continues on the next page...

Sample Reference Manual
Freescale Semiconductor, Inc. 439

Figure 1-3. Example: chip-specific information that refers to a different chapter

1.4 Register descriptions

Module chapters present register information in:

- Memory maps including:
  - Addresses
  - The name and acronym/abbreviation of each register
  - The width of each register (in bits)
  - Each register's reset value
  - The page number on which each register is described
- Register figures
- Field-description tables
- Associated text

The register figures show the field structure using the conventions in the following figure.

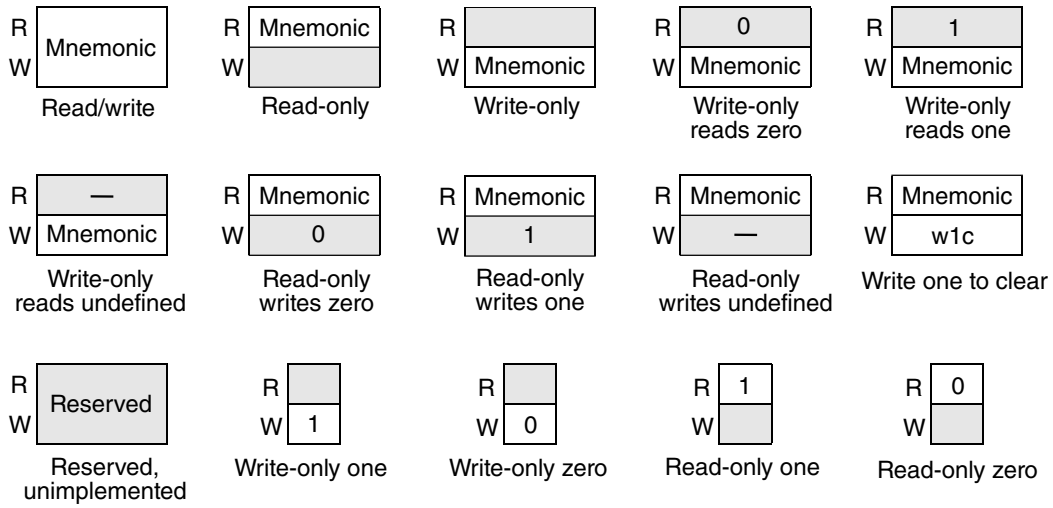


Figure 1-4. Register figure conventions

1.5 Conventions

1.5.1 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*.

1.5.2 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

Table continues on the next page...

Example	Description
	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: • A subset of a register's named field 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. • A continuous range of individual signals of a bus For example, XAD[7:0] refers to signals 7–0 of the XAD bus.

1.5.3 Special terms

The following terms have special meanings:

Term	Meaning
asserted	Refers to the state of a signal as follows: • An active-high signal is asserted when high (1). • An active-low signal is asserted when low (0).
deasserted	Refers to the state of a signal as follows: • An active-high signal is deasserted when low (0). • An active-low signal is deasserted when high (1). In some cases, deasserted signals are described as *negated*.
reserved	Refers to a memory space, register, field, or programming setting. Writes to a reserved location can result in unpredictable functionality or behavior. • Do not modify the default value of a reserved programming setting, such as the reset value of a reserved register field. • Consider undefined locations in memory to be reserved.
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 Block diagram

The below figure shows a top-level block diagram of the device.

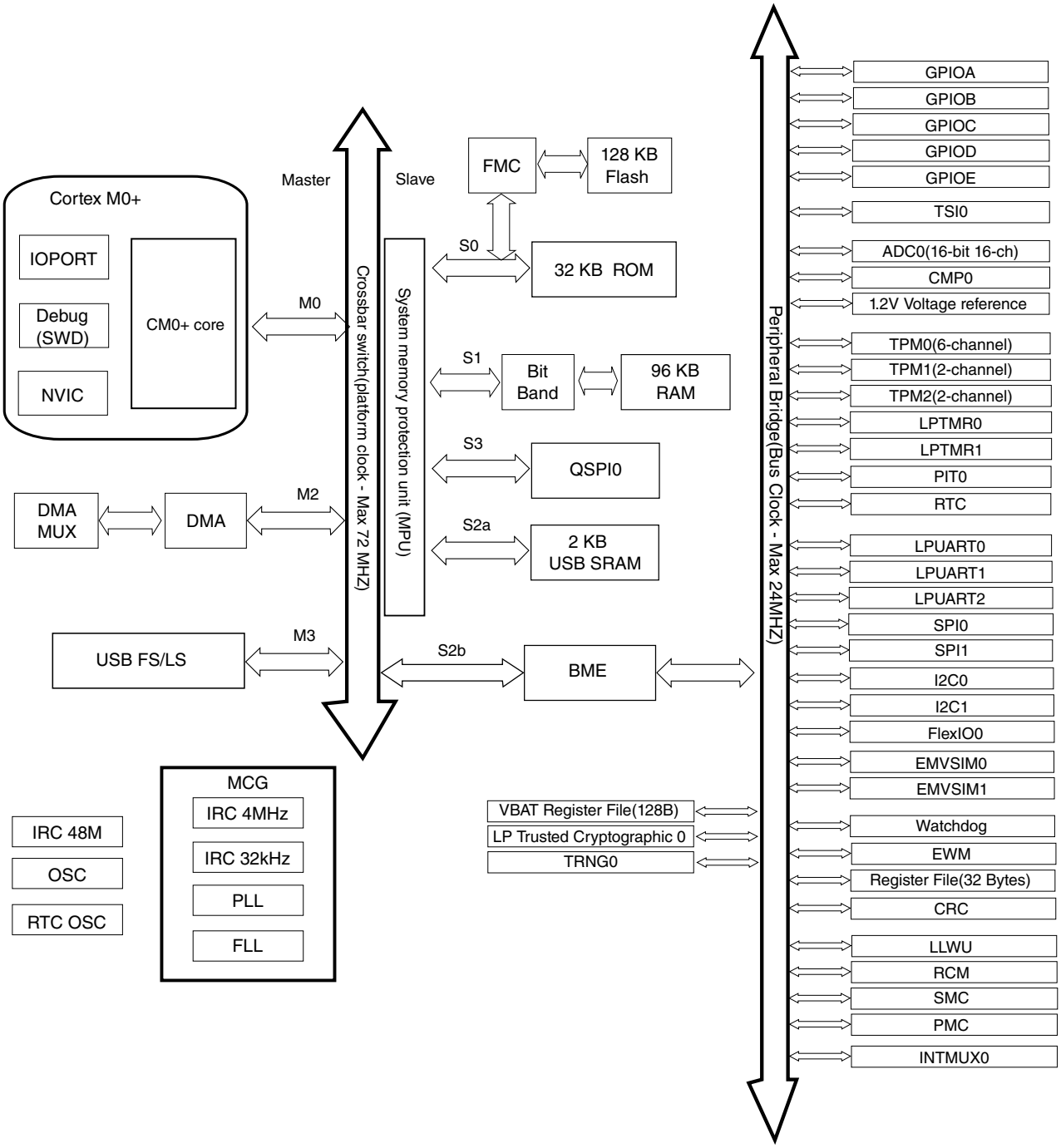


Figure 2-1. KL82 block diagram

2.3 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®-M0+ based core	• 32-bit MCU core from ARM's Cortex-M class, 1.77 CoreMark®/MHz from single-cycle access memories
System	• System integration module • Power management and mode controllers • Multiple power modes available based on high speed run, run, wait, stop, and power-down modes • Low-leakage wakeup unit • Miscellaneous control module • Crossbar-lite switch • Memory protection unit • Peripheral bridge-lite • Bit manipulation engine (BME) • Direct memory access (DMA) controller with multiplexer to increase available DMA requests. DMA can handle transfers in VLPS mode • External watchdog monitor • Watchdog

Memories	• Internal memories include:
	  • Program flash memory
	  • SRAM
	• 32-byte system register file
	• 128-byte VBAT register file
	• Quad serial peripheral interface
	• 32 KB boot ROM

Clocks	• Multiple clock generation options available from internally- and externally-generated clocks • System oscillator to provide clock source for the MCU • RTC oscillator to provide clock source for the RTC • 48 MHz Internal Reference Clock
Security	• Cyclic Redundancy Check module for error detection • Hardware encryption, along with a random number generator
Analog	• High speed analog-to-digital converter with integrated programmable gain amplifier • Comparator • Digital-to-analog converter • Internal voltage reference • Bandgap voltage reference
Timers	• Low power timer/PWM • Periodic interrupt timer • Low power timer • Independent real time clock
Communications	• USB FS OTG controller • Built-in FS/LS transceiver • Serial peripheral interface (SPI) • Inter-integrated circuit (I2C) • Low Power Universal Asynchronous Receiver Transmitter (LPUART) • FlexIO • EMVSIM (ISO-7816) Module
Human-Machine Interfaces (HMI)	• General purpose input/output controller • Capacitive touch sense input interface enabled in hardware

2.3.1 ARM® Cortex®-M0+ Core Modules

The following core modules are available on this device.

Table 2-2. Core modules

Module	Description
ARM Cortex-M0+	The ARM Cortex-M0+ is the newest member of the Cortex M Series of processors targeting microcontroller applications focused on very cost sensitive, deterministic, interrupt driven environments. The Cortex M0+ processor is based on the ARMv6 Architecture and Thumb®-2 ISA and is 100% instruction set compatible with its predecessor, the Cortex-M0 core, and upward compatible to Cortex-M3 and M4 cores.
NVIC	The ARMv6-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. 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 hardware is based on the ARM CoreSight™ architecture. • Serial Wire Debug (SWD)

2.3.2 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 (SMC)	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. More than 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.

Table continues on the next page...

Table 2-3. System modules (continued)

Module	Description
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.
Miscellaneous control module (MCM)	The MCM includes integration logic and embedded trace buffer details.
Crossbar switch (AXBS)	The AXBS 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 (AIPS-Lite)	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.
Enhanced direct memory access (eDMA) controller	The eDMA 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 operations 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.
Bit manipulation engine (BME)	The BME provides hardware support for atomic read-modify-write operations to the peripheral address space.

2.3.3 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 • 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 LLS2 and VLLS2 low leakage modes.
System register file	32-byte register file that is accessible during all power modes and is powered by VDD.
VBAT register file	128-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
Quad Serial Peripheral Interface	The Quad Serial Peripheral Interface (QuadSPI) block acts as an interface to one single or two external serial flash devices, each with up to eight bidirectional data lines.
32 KB Boot ROM	Provisions the internal flash or QSPI with an embedded firmware image during manufacturing, or at any time during the life of the device. It does this by acting as a slave device, and listening to various peripheral ports where a master can start communication

2.3.4 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: • Phase-locked loop (PLL) — Voltage-controlled oscillator (VCO) • Frequency-locked loop (FLL) — Digitally-controlled oscillator (DCO) • Internal reference clocks — Can be used as a clock source for other on-chip peripherals
48 MHz internal reference clock (IRC48M)	The IRC48M provides an internally generated clock source. Clock recovery circuitry uses the incoming USB data stream to adjust the internal oscillator and enables the internal oscillator to meet the requirements for USB clock tolerance.
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.

2.3.5 Analog modules

The following analog modules are available on this device:

Table 2-6. Analog modules

Module	Description
16-bit analog-to-digital converters (ADC)	16-bit successive-approximation ADC.
Comparator (CMP)	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.

Table continues on the next page...

Table 2-6. Analog modules (continued)

Module	Description
12-bit digital-to-analog converters (DAC)	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.

2.3.6 Timer modules

The following timer modules are available on this device:

Table 2-7. Timer modules

Module	Description
Timer/PWM module (TPM)	• Selectable TPM clock mode • Prescaler divide-by 1, 2, 4, 8, 16, 32, 64, or 128 • 16-bit free-running counter or modulo counter with counting be up or updown • Configurable channels for input capture, output compare, or edge-aligned PWM mode • Support the generation of an interrupt per channel • Support the generation of an interrupt when the counter overflows • Support selectable trigger input to optionally reset or cause the counter to start incrementing • Support the generation of hardware triggers when the counter overflows and per channel
Periodic interrupt timers (PIT)	• Four general purpose interrupt timers • Interrupt timers for triggering ADC conversions • 32-bit counter resolution • DMA support
Low-power timer (LPTMR)	• Selectable clock for prescaler/glitch filter of 1 kHz (internal LPO), 32.768 kHz (external crystal), or internal reference clock • Configurable glitch filter or prescaler with 16-bit counter • 16-bit time or pulse counter with compare • Interrupt generated on timer compare • Hardware trigger generated on timer compare
Real-time clock (RTC)	• Independent power supply, POR, and 32 kHz crystal oscillator • 32-bit seconds counter with 32-bit alarm • 16-bit Prescaler with compensation that can correct errors between 0.12 ppm and 3906 ppm • Time and monotonic counters are invalidated on tamper detection

2.3.7 Security and Integrity modules

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

Table 2-8. Security and integrity modules

Module	Description
LP trusted cryptography (LTC)	LTC is a hardware block that supports the following algorithms: DES, 3DES, AES algorithms
True random number generator (TRNG)	A hardware entropy source which can be used to seed a software random number generator.
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.

2.3.8 Communication interfaces

The following communication interfaces are available on this device:

Table 2-9. Communication modules

Module	Description
FlexIO	Supports a wide range of protocols including, but not limited to: • UART • I2C • SPI • I2S • Camera IF • LCD RGB • PWM / Waveform generation Functions in STOP/VLPS modes DMA support
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.
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.
EMVSIM (ISO-7816) Module	Allows communication on ISO-7816
Low Power Universal asynchronous receiver/transmitter (LPUART)	Asynchronous serial bus communication interface with programmable 8- or 9-bit data format

2.3.9 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. Configurable slew rate on all output pins.
Capacitive touch sense input (TSI)	Contains up to 16 channel inputs for capacitive touch sensing applications. Operation is available in low-power modes via interrupts.

2.4 Ordering information

The following chips are available for ordering.

Table 2-11. Ordering information

Product		Memory		Package		IO and ADC channel		
Part number	Marking (Line1/Line2)	Flash (KB)	SRAM (KB)	Pin count	Package	GPIOs	GPIOs (INT/ HD)[1]	ADC channels (SE/DP)
MKL82Z128VMC7(R)	MKL82 Z128VMC7	128	96	121	MAPBGA	85	85/0	16/2
MKL82Z128VLL7(R)	MKL82Z128VL L7	128	96	100	LQFP	66	66/0	14/1
MKL82Z128VLK7(R)	MKL82Z128 VLK7	128	96	80	LQFP	56	56/0	12/1
MKL82Z128VMP7(R)	M82N7V	128	96	64	MAPBGA	41	41/0	11/1
MKL82Z128VLH7(R)	MKL82Z128V LH7	128	96	64	LQFP	41	41/0	11/1

1. INT: interrupt pin numbers; HD: high drive pin numbers

NOTE

The 100-, 64-pin LQFP and 64-pin MAPBGA packages supporting MKL82Z128VLL7, MKL82Z128VLH7 and MKL82Z128VMP7 part numbers for this product are not yet available. However, these packages are included in Package Your Way program for Kinetis MCUs. Visit nxp.com/KPYW for more details.

Chapter 3
Core Overview

3.1 ARM Cortex-M0+ Core

This device has one ARM Cortex M0+ CPUs.

The enhanced ARM Cortex M0+ is the member of the Cortex-M Series of processors targeting microcontroller cores focused on very cost sensitive, low power applications. It has a single 32-bit AMBA AHB-Lite interface and includes an NVIC component. It also has hardware debug functionality including support for simple program trace capability. The processor supports the ARMv6-M instruction set (Thumb) architecture including all but three 16-bit Thumb opcodes (52 total) plus seven 32-bit instructions. It is upward compatible with other Cortex-M profile processors.

3.1.1 Buses, interconnects, and interfaces

The ARM Cortex-M0+ core has two bus interfaces:

- Single 32-bit AMBA-3 AHB-Lite system interface that provides connections to peripherals and all system memory, which includes flash memory and RAM
- Single 32-bit I/O port bus interfacing to the GPIO with 1-cycle loads and stores

3.1.2 System tick timer

The CLKSOURCE field in SysTick Control and Status register selects either the core clock (when CLKSOURCE = 1) or a divide-by-16 of the core clock (when CLKSOURCE = 0). Because the timing reference is a variable frequency, the TENMS field in the SysTick Calibration Value Register is always 0.

3.1.3 Debug facilities

This device supports standard ARM 2-pin SWD debug port.

3.1.4 Core privilege levels

The ARM documentation uses different terms from 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 Nested vectored interrupt controller (NVIC) configuration

3.2.1 Interrupt priority levels

This device supports four priority levels for interrupts. Therefore, in the NVIC, each source in the IPR registers contains two 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	0	0	IRQ2		0	0	0	0	0	0	IRQ1		0	0	0	0	0	0	IRQ0		0	0	0	0	0	0
W																																

3.2.2 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.

3.2.3 Interrupt connections

The device has one NVIC module.

There is one INTMUX module, the INTMUX0. This INTMUX0 module has four outputs that each is given a fixed NVICn vector space. INTMUX0-[CH3:CH0] has four dedicated vector slots on NVIC.

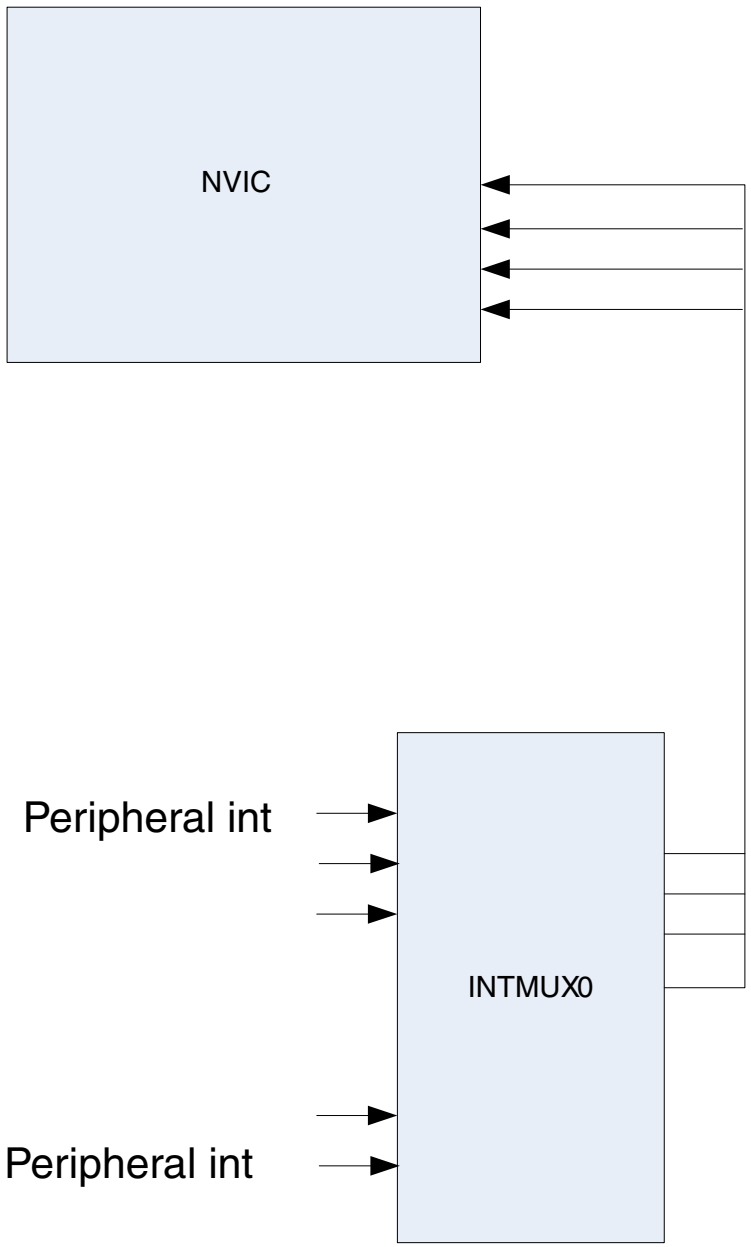


Figure 3-1. Interrupt routing block diagram

CPU can mask off any interrupts connected to the 32-slot NVICn module.

The INTMUX0 module allows the ORing of selectable interrupt sources to one specific NVICx vector interrupt slot of the CPU. The INTMUX0 module also allows ANDing of selectable interrupt sources to one specific NVICx vector interrupt slot of the CPU.

3.2.4 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 3-2. NVIC

Address	Vector	IRQ	Source module	Source description
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 (MPU)
0x0000_0010	4	—	—	—
0x0000_0014	5	—	—	—
0x0000_0018	6	—	—	—
0x0000_001C	7	—	—	—
0x0000_0020	8	—	—	—
0x0000_0024	9	—	—	—
0x0000_0028	10	—	—	—
0x0000_002C	11	—	ARM core	Supervisor call (SVCall)
0x0000_0030	12	—	—	—
0x0000_0134	13	—	—	—
0x0000_0038	14	—	ARM core	Pendable request for system service (PendableSrvReq)
0x0000_003C	15	—	ARM core	System tick timer (SysTick)
On Platform vectors	—	—	—	—
0x0000_0040	16	0	DMA0	DMA0 channel 0 or 4 transfer complete[1]
0x0000_0044	17	1	DMA0	DMA0 channel 1 or 5 transfer complete[1]
0x0000_0048	18	2	DMA0	DMA0 channel 2 or 6 transfer complete[1]
0x0000_004C	19	3	DMA0	DMA0 channel 3 or 7 transfer complete[1]
0x0000_0050	20	4	DMA0	DMA0 error interrupt
Off Platform vectors	—	—	—	—
0x0000_0054	21	5	FlexIO0	Flexible IO
0x0000_0058	22	6	TPM0	Timer/PWM module 0
0x0000_005C	23	7	TPM1	Timer/PWM module 1
0x0000_0060	24	8	TPM2	Timer/PWM module 2

Table continues on the next page...

Table 3-2. NVIC (continued)

Address	Vector	IRQ	Source module	Source description
0x0000_0064	25	9	PIT0	Periodic interrupt timer 0
0x0000_0068	26	10	SPI0	Serial peripheral interface 0
0x0000_006C	27	11	EMVSIM0	EMVSIM0
0x0000_0070	28	12	LPUART0	Low power UART 0
0x0000_0074	29	13	LPUART1	Low power UART 1
0x0000_0078	30	14	I2C0	I2C module 0
0x0000_007C	31	15	QSPI0	QSPI0
0x0000_0080	32	16	—	—
0x0000_0084	33	17	PortA	Port A
0x0000_0088	34	18	PortB	Port B
0x0000_008C	35	19	PortC	Port C
0x0000_0090	36	20	PortD	Port D
0x0000_0094	37	21	PortE	Port E
0x0000_0098	38	22	LLWU	Low leakage wake up
0x0000_009C	39	23	LTC0	Low power trusted cryptographic 0
0x0000_00A0	40	24	USB0	Universal serial bus 0
0x0000_00A4	41	25	ADC0	Analog to digital convertor 0
0x0000_00A8	42	26	LPTMR0	Low power timer 0
0x0000_00AC	43	27	RTC Secs	Real time clock seconds
0x0000_00B0	44	28	INTMUX0-0	Selectable peripheral interrupt INTMUX0-0
0x0000_00B4	45	29	INTMUX0-1	Selectable peripheral interrupt INTMUX0-1
0x0000_00B8	46	30	INTMUX0-2	Selectable peripheral interrupt INTMUX0-2
0x0000_00BC	47	31	INTMUX0-3	Selectable peripheral interrupt INTMUX0-3

1. DMA0 channel 4/5/6/7 interrupt can be optional masked through SIM_MISCCTRL[DMAINTSELn] (n=0-3) fields

3.2.5 INTMUX0 input mux assignment

Table 3-3. INTMUX0 peripheral interrupt assigment

INTMUX0 input	Periphera interrupt source
0	LPTMR1
1	Reserved
2	Reserved
3	Reserved
4	SPI1
5	LPUART2
6	EMVSIM1
7	I2C1

Table continues on the next page...

Table 3-3. INTMUX0 peripheral interrupt assigment (continued)

INTMUX0 input	Periphera interrupt source
8	TSI0
9	PMC
10	FTFA
11	MCG
12	WDOG0/EWM
13	DAC0
14	TRNG0
15	Reserved
16	CMP0
17	Reserved
18	RTC Alarm
19	Reserved
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	DMA0-4
25	DMA0-5
26	DMA0-6
27	DMA0-7
28	Reserved
29	Reserved
30	Reserved
31	Reserved

3.3 Asynchronous wake-up interrupt controller (AWIC) configuration

3.3.1 AWIC overview

The primary function of the AWIC block 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.

3.3.2 Wake-up sources

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

Table 3-4. AWIC Partial Stop, Stop and VLPS wake-up sources

Wake-up source	Description
Available system resets	RESET_b pin and WDOG when LPO is its clock source, and Debug
Low-voltage detect	Power mode controller
Low-voltage warning	Power mode controller
Pin interrupts	Port control module - any enabled pin interrupt is capable of waking the system
ADC0	The ADC is functional when using internal clock source
CMPx	Since no system clocks are available, functionality is limited, trigger mode provides wakeup functionality with periodic sampling
I2Cx	Address match wakeup
LPUARTx	Functional when using clock source which is active in Stop and VLPS modes
USB FS/LS Controller	Wakeup
FlexIO0	Functional when using clock source which is active in Stop and VLPS modes
LPTMR	Functional when using clock source which is active in Stop, VLPS and LLS/VLLS modes
RTC	Functional in Stop/VLPS modes
TPM	Functional when using clock source which is active in Stop and VLPS modes
TSI0	Wakeup
NMI	Non-maskable interrupt

Chapter 4
Memories and Memory Interfaces

4.1 Flash memory

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

Table 4-1. Flash memory size

Program flash (KB)	Block 0 (P-Flash) address range
128 KB	0x0000_0000—0x0001_FFFF

4.1.1 Flash memory types

This device contains the following type of flash memory:

- Program flash memory — non-volatile flash memory that can execute program code

4.1.2 Flash memory sizes

The devices covered in this document contain:

- 1 block 128 KB of program flash consisting of 4 KB sectors

4.1.3 Flash security

How flash security is implemented on this device is described in Flash security.

4.1.4 Flash modes

The flash memory is always configured in NVM normal mode. There are no operating conditions in which the flash is configured for NVM special mode.

4.1.5 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 from the SWJ-DP debug port by setting DAP_CONTROL[0]. If DAP_STATUS[0]=1, it indicates the mass erase command has been accepted. DAP_STATUS[0]=0 when the mass erase completes.

4.1.6 FTFA_FOPT register

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

4.1.7 Flash access control introduction

The flash access control (FAC) is a NXP or third-party configurable memory protection scheme optimized to allow end users to utilize software libraries while offering programmable restrictions to these libraries. The flash memory is divided into equal size segments that provide protection to proprietary software libraries. The protection of these segments is controlled as the FAC evaluates of the access rights for each transaction routed to the on-chip flash memory. Configurability allows an increasing number of protected segments while supporting two levels of vendors adding their proprietary software to a device.

Flash access control aligns to the three privilege levels supported by ARM Cortex-M family products where the most secure state - supervisor/privileged-secure - aligns to the execute-only and supervisor-only access control. The unsecure state of user-non-secure aligns to no access control states set, and the mid-level state where user-secure aligns to using the access control of execute-only.

Control for this protection scheme is implemented in Program Once NVM locations and is configurable through a Program Once flash command operations. The NVM locations controlling FAC are unaffected by Erase All Blocks flash command and debug interface initiated mass erase operations.

NOTE

The FAC protection scheme has eight XACC and eight SACC registers to control up to 64 segments. For program flash sizes 128 KB or less, the memory is divided into 32 segments, controlled by the four lower-order XACC and SACC registers.

4.2 SRAM

4.2.1 SRAM sizes

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

This device contains SRAM tightly coupled to the ARM Cortex-M0+ core. The on-chip SRAM is split into SRAM_L and SRAM_U regions where the SRAM_L and SRAM_U ranges form a contiguous block in the memory map anchored at address 0x2000_0000. As such:

- SRAM_L is anchored at address 0x2000_0000 and occupies the space below this address, that is, addresses less than 0x2000_0000.
- SRAM_U is anchored at address 0x2000_0000 and occupies the space at and above this beginning address, that is, addresses greater than or equal to 0x2000_0000.

The amount of SRAM for the devices covered in this document is shown in the following table:

Table 4-2. SRAM memory size

SRAM (KB)	SRAM_L size (KB)	SRAM_U size (KB)	Address range
96	24	72	0x1FFF_A000–0x2001_1FFF

4.2.2 SRAM retention in low power modes

The SRAM is retained down to LLS3 and VLLS3 mode.

In LLS2 and VLLS2 the 32KB region of SRAM_U based at 0x2000_0000 are powered. These regions (or partitions) of SRAM are labeled as follows:

- RAM1: the 32KB region of SRAM_U are powered always powered in VLLS2 and LLS2
- RAM3: the rest of system RAM

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

4.3 QuadSPI memory interface

This device contains a QSPI controller which can interface with external flash devices to expand the storage size. With the QSPI controller enabled, the external flash can be mapped to a 128 MB system address range 0x6800_0000 to 0x6FFF_FFFF and can be read directly just like on-chip memory. For the flash space which exceeds above range, it can be read in through a 16 words QSPI FIFO located at system address 0x6700_0000. QSPI is not available when the system enters Stop modes, Compute Operation mode, LLSx and VLLSx modes.

4.4 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.

4.5 VBAT register file

This device includes a 128-byte register file that are powered in all power modes and are powered by VBAT. The VBAT register file clock gate is controlled by SIM_SCGC6[RTC_RF].

It is reset only during VBAT power-on reset.

4.6 Memory map

4.6.1 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.

4.6.2 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.

The system memory map includes multiple aliased address spaces that are intended for specific purposes.

- The bitbanding functionality uses aliased regions that map to the SRAM_U. This functionality maps each 32-bit word of the aliased address space to a unique bit in the underlying RAM to support single-bit insert and extract operations from the processor.

Table 4-3. System memory map

System 32-bit Address Range	Destination Slave	Access
0x0000_0000-0x03FF_FFFF [1]	Program flash and read-only data (Includes exception vectors in first 1024 bytes)	All masters
0x0400_0000-0x07FF_FFFF	Reserved	—
0x0800_0000-0x0FFF_FFFF	Reserved	—
0x1000_0000-0x17FF_FFFF	Reserved	—
0x1800_0000-0x1BFF_FFFF	Reserved	—
0x1C00_0000–0x1C00_7FFF	ROM	All masters
0x1C00_8000-0x1FFF_9FFF	Reserved	—
0x1FFF_A000–0x1FFF_FFFF[2]	SRAM_L: lower SRAM	All masters
0x2000_0000–0x2001_1FFF[2]	SRAM_U: upper SRAM bitband region	All masters
0x2001_2000-0x21FF_FFFF	Reserved	—
0x2200_0000-0x23FF_FFFF	Aliased to TCMU SRAM bitband	Cortex-M0+ core only
0x2400_0000-0x2FFF_FFFF	Reserved	—
0x3000_0000-0x33FF_FFFF	Reserved	—
0x3400_0000-0x3FFF_FFFF	Reserved	—
0x4000_0000-0x4007_FFFF	AIPS0	Cortex-M0+ core & DMA
0x4008_0000-0x400F_EFFF	Reserved	—
0x400F_F000-0x400F_FFFF	GPIO	Cortex-M0+ core & DMA

Table continues on the next page...

Table 4-3. System memory map (continued)

System 32-bit Address Range	Destination Slave	Access
0x4010_0000-0x4010_07FF	USB SRAM	All masters
0x4010_0800-0x41FF_FFFF	Reserved	—
0x4200_0000-0x43FF_FFFF	Reserved	—
0x4400_0000-0x5FFF_FFFF	Bit manipulation engine (BME) access to AIPS0 peripheral slots 0-127	Cortex-M0+ core only
0x6000_0000-0x66FF_FFFF	Reserved	—
0x6700_0000-0x677F_FFFF	QSPI0 Rx buffer	All masters
0x6780_0000-0x67FF_FFFF	Reserved	—
0x6800_0000-0x6FFF_FFFF	QSPI0 (External Memory)	All masters
0x7000_0000-0x7FFF_FFFF	Reserved	—
0x8000_0000-0x87FF_FFFF	Reserved	—
0x8800_0000-0x8FFF_FFFF	Reserved	—
0x9000_0000-0x97FF_FFFF	Reserved	—
0x9800_0000-0x9FFF_FFFF	Reserved	—
0xA000_0000-0xDFFF_FFFF	Reserved	—
0xE000_0000-0xE00F_FFFF	Private peripherals	Cortex-M0+ core only
0xE010_0000-0xEFFF_FFFF	Reserved	—
0xF000_0000-0xFFFF_FFFF	Private peripheral bus (PPB)	Cortex-M0+ core only

1. This map provides the complete architectural address space definition for the flash. Based on the physical sizes of the memories implemented for a particular device, the actual address regions used may be smaller.
2. This range varies depending on amount of SRAM implemented for a particular device.

NOTE

1. Access to the AIPS-Lite peripheral bridge and general purpose input/output (GPIO) module address space is limited to the core and DMA.
2. ARM Cortex-M0+ core access privileges also include accesses via the debug interface.

4.6.2.1 Aliased bit-band regions

The SRAM_U resources reside in the Cortex-M0+ 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

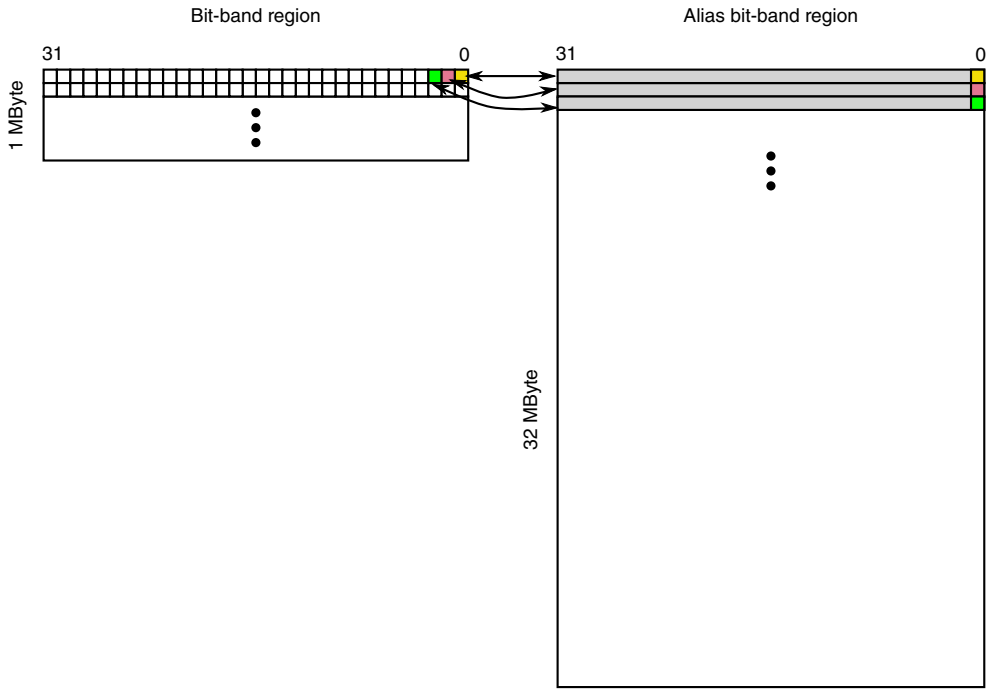


Figure 4-1. Alias bit-band mapping

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.

4.6.3 Flash memory map

The flash memory 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.

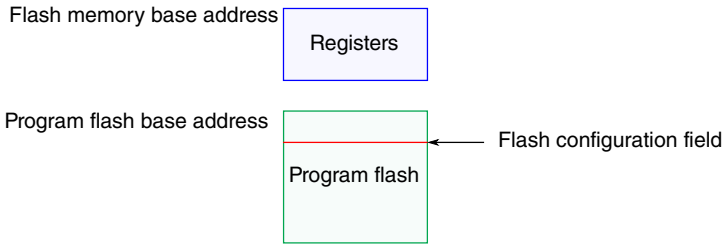


Figure 4-2. Flash memory map

The on-chip Flash is implemented in a portion of the allocated Flash range to form a contiguous block in the memory map beginning at address 0x0000_0000. See Flash memory for details of supported ranges.

Accesses to the flash memory ranges outside the amount of Flash on the device causes the bus cycle to be terminated with an error followed by the appropriate response in the requesting bus master. Read collision events in which flash memory is accessed while a flash memory resource is being manipulated by a flash command also generates a bus error response.

4.6.4 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 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.

4.6.5 Peripheral bridge (AIPS-Lite) memory map

The peripheral memory map is accessible via one slave port on the crossbar in the 0x4000_0000–0x4007_FFFF region. The device implements one peripheral bridge that defines a 512 KB address space.

- AIPS-Lite covers 512 KB

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

The two regions associated with this space are:

- A 128 KB region, partitioned as 32 spaces, each 4 KB in size and reserved for on-platform peripheral devices. The AIPS controller generates unique module enables for all 32 spaces.
- A 384 KB region, partitioned as 96 spaces, each 4 KB in size and reserved for off-platform modules. The AIPS controller generates unique module enables for all 96 spaces.

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.

4.6.5.1 Peripheral bridge 0 (AIPS-Lite 0) memory map

NOTE

- Peripherals generate transfer error for reserved addresses in their assigned slots

Table 4-4. Peripheral bridge slot assignments

System 32-bit base address	Slot number	Module
0x4000_0000	0	AIPS-Lite
0x4000_1000	1	—
0x4000_2000	2	—
0x4000_3000	3	—
0x4000_4000	4	—
0x4000_5000	5	—
0x4000_6000	6	—
0x4000_7000	7	—
0x4000_8000	8	DMA controller (DMA)
0x4000_9000	9	DMA TCD
0x4000_A000	10	—
0x4000_B000	11	—
0x4000_C000	12	—
0x4000_D000	13	System MPU
0x4000_E000	14	—
0x4000_F000	15	GPIO controller[1]
0x4001_0000	16	—

Table continues on the next page...

Table 4-4. Peripheral bridge slot assignments (continued)

System 32-bit base address	Slot number	Module
0x4001_1000	17	—
0x4001_2000	18	—
0x4001_3000	19	—
0x4001_4000	20	—
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	—
Off-platform modules		
0x4002_0000	32	Flash memory unit (FTFA)
0x4002_1000	33	DMAMUX
0x4002_2000	34	—
0x4002_3000	35	—
0x4002_4000	36	INTMUX0
0x4002_5000	37	TRNG0
0x4002_6000	38	—
0x4002_7000	39	—
0x4002_8000	40	—
0x4002_9000	41	—
0x4002_A000	42	—
0x4002_B000	43	—
0x4002_C000	44	SPI0
0x4002_D000	45	SPI1
0x4002_E000	46	—
0x4002_F000	47	—
0x4003_0000	48	—
0x4003_1000	49	—
0x4003_2000	50	CRC
0x4003_3000	51	—
0x4003_4000	52	—
0x4003_5000	53	—
0x4003_6000	54	—

Table continues on the next page...

Table 4-4. Peripheral bridge slot assignments (continued)

System 32-bit base address	Slot number	Module
0x4003_7000	55	PIT0
0x4003_8000	56	TPM0
0x4003_9000	57	TPM1
0x4003_A000	58	TPM2
0x4003_B000	59	ADC0
0x4003_C000	60	—
0x4003_D000	61	RTC
0x4003_E000	62	VBAT register file
0x4003_F000	63	DAC0
0x4004_0000	64	LPTMR0
0x4004_1000	65	System register file
0x4004_2000	66	—
0x4004_3000	67	—
0x4004_4000	68	LPTMR1
0x4004_5000	69	TSI0
0x4004_6000	70	—
0x4004_7000	71	SIM low power logic
0x4004_8000	72	SIM
0x4004_9000	73	Port A mutiplex control
0x4004_A000	74	Port B mutiplex control
0x4004_B000	75	Port C mutiplex control
0x4004_C000	76	Port D mutiplex control
0x4004_D000	77	Port E mutiplex control
0x4004_E000	78	EMVSIM0
0x4004_F000	79	EMVSIM1
0x4005_0000	80	—
0x4005_1000	81	LTC0
0x4005_2000	82	Watchdog
0x4005_3000	83	—
0x4005_4000	84	LPUART0
0x4005_5000	85	LPUART1
0x4005_6000	86	LPUART2
0x4005_7000	87	—
0x4005_8000	88	—
0x4005_9000	89	—
0x4005_A000	90	QSPI0
0x4005_B000	91	—
0x4005_C000	92	—
0x4005_D000	93	—

Table continues on the next page...

Table 4-4. Peripheral bridge slot assignments (continued)

System 32-bit base address	Slot number	Module
0x4005_E000	94	—
0x4005_F000	95	FlexIO0
0x4006_0000	96	—
0x4006_1000	97	EWM
0x4006_2000	98	—
0x4006_3000	99	—
0x4006_4000	100	MCG
0x4006_5000	101	OSC
0x4006_6000	102	I2C0
0x4006_7000	103	I2C1
0x4006_8000	104	—
0x4006_9000	105	—
0x4006_A000	106	—
0x4006_B000	107	—
0x4006_C000	108	—
0x4006_D000	109	—
0x4006_E000	110	—
0x4006_F000	111	—
0x4007_0000	112	—
0x4007_1000	113	—
0x4007_2000	114	USB FS
0x4007_3000	115	CMP0
0x4007_4000	116	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	LLWU
0x4007_D000	125	PMC
0x4007_E000	126	SMC
0x4007_F000	127	RCM
0x400F_F000	—	GPIO (All ports: PTA to PTE)[2]

1. Alias of the GPIO module
2. This address range is for accessing GPIO through AIPS. For single cycle access, IOPORT is used

4.6.6 Private peripherals

Table 4-5. Private peripherals

System 32-bit Address Range	Destination Slave
0xE000_0000–0xE000_DFFF	Reserved
0xE000_E000–0xE000_EFFF	System control space(SCS)
0xE000_E000–0xE000_E00F	Reserved
0xE000_E010–0xE000_E0FF	SysTick
0xE000_E100–0xE000_ECFF	NVIC
0xE000_ED00– 0xE000_ED8F	System control block
0xE000_ED90–0xE000_EDEF	Reserved
0xE000_EDF0–0xE000_EEFF	Debug
0xE000_EF00–0xE000_EFFF	Reserved
0xE000_F000–0xE00F_EFFF	Reserved
0xE00F_F000-–0xE00F_FFFF	M0+ core ROM table

4.6.7 Private peripheral bus (PPB) memory map

The PPB is a part of the defined ARM bus architecture and provides access to select processor-local modules. These resources are accessible only from the core; other system masters do not have access to them. The M0+ core does not support a true "External PPB". It has an IOPORT which is used to access the GPIO.

All of the resources shown in the table below are standard ARM supplied blocks supporting the ARM Cortex-M0+ except for the miscellaneous control modules.

Table 4-6. PPB memory map

System 32-bit address range	Resource
0xF000_0000–0xF000_0FFF	Micro trace buffer (MTB) for program trace
0xF000_1000–0xF000_1FFF	MTB data watchpoint and trace
0xF000_2000–0xF000_2FFF	Debug ROM table
0xF000_3000–0xF000_3FFF	Miscellaneous control module (MCM)
0xF000_4000–0xF000_4FFF	—
0xF000_5000–0xF000_5FFF	Reserved
0xF000_6000–0xF7FF_FFFF	Reserved
0xF800_0000–0xFFFF_FFFF	IOPORT (connected to GPIO)[1]

1. This port accesses one port of the GPIO while the other ports of the GPIO are connected to AIPS as normal IPS connections.

NOTE

The platform provides access control registers in the GPIO module thereby securing the accesses to GPIO. Please refer to the General-Purpose Input/Output (GPIO) for more details.

Chapter 5
Clock Distribution using MCG

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 IRC48MCLK, 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.

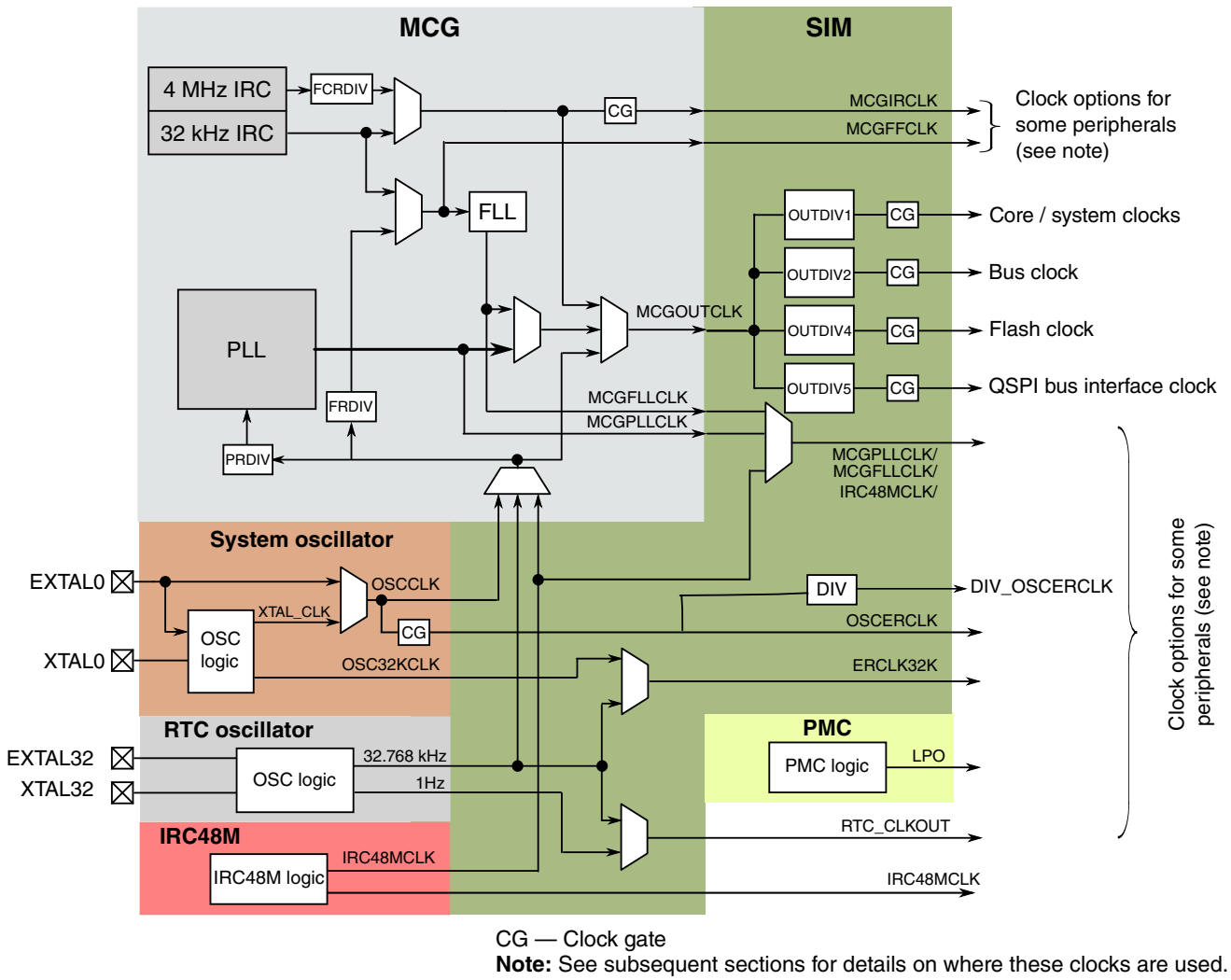
5.2 Programming model

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. See 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*



CG — Clock gate
Note: See subsequent sections for details on where these clocks are used.

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-M0+ core
System clock	MCGOUTCLK divided by OUTDIV1 clocks the crossbar switch and bus masters directly connected to the crossbar.
Bus clock	MCGOUTCLK divided by OUTDIV2 clocks the bus slaves and peripherals (excluding memories)
Flash clock	MCGOUTCLK divided by OUTDIV4 clocks the flash memory
QSPI bus interface clock	MCGOUTCLK divided by OUTDIV5 clocks the QSPI module bus interface
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, 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. MCGPLLCLK may clock some modules.
MCG external reference clock	Input clock to the MCG sourced by the system oscillator (OSCCLK) or RTC oscillator
IRC48MCLK	Internal 48 MHz oscillator that can be used as a reference to the MCG and also may clock some on-chip modules.
OSCCLK	System oscillator output of the internal oscillator or sourced directly from EXTAL
OSCERCLK	System oscillator output sourced from OSCCLK that may clock some on-chip modules
OSC32KCLK	System oscillator 32 kHz output
ERCLK32K	Clock source for some modules that is chosen as OSC32KCLK or the RTC clock. It is VLPOSCCLK for TSI.
RTC clock	RTC oscillator output for the RTC module
LPO	PMC 1 kHz 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	HSRun mode clock frequency	Run mode clock frequency	VLPR mode clock frequency	Clock source	Clock is disabled when…
MCGOUTCLK	Up to 144 MHz	Up to 144MHz	Up to 4 MHz	MCG	In all stop modes except for partial stop modes and during PLL locking when MCGOUTCLK derived from PLL.
MCGFLLCLK	Up to 96 MHz	Up to 96 MHz	N/A	MCG	MCG clock controls do not enable. Overriding forced disable in all low powers modes (including STOP and VLPx modes).
MCGPLLCLK	Up to 144 MHz	Up to 144 MHz	N/A	MCG	MCG clock controls do not enable, in Stop mode but PLLSTEN=0, or in VLPS, LLS and VLLSx modes
Core clock	Up to 96 MHz	Up to 72 MHz	Up to 4 MHz	MCGOUTCLK clock divider	In all Wait and Stop modes
System clock	Up to 96 MHz	Up to 72 MHz	Up to 4 MHz	MCGOUTCLK clock divider	In all Stop modes and compute operation
Bus clock	Up to 24 MHz	Up to 24 MHz	Up to 1 MHz	MCGOUTCLK clock divider	In all Stop modes except for partial Stop2 mode, and compute operation
Flash clock	Up to 24 MHz	Up to 24 MHz	Up to 1MHz in BLPE, Up to 800KHz in BLPI	MCGOUTCLK clock divider	In all Stop modes except for partial Stop2 mode
QSPI bus interface clock	Up to 96 MHz, must be system clock/2 or upper	Up to 72 MHz, must be system clock/2 or upper	Up to 4 MHz	MCGOUTCLK clock divider	In all Stop modes except for partial Stop2 mode
Internal reference (MCGIRCLK)	Up to 4 MHz IRC oscillator	Up to 4 MHz IRC oscillator	Up to 4 MHz IRC oscillator	MCG	MCG_C1[IRCLKEN] cleared, Stop or VLPS mode and MCG_C1[IREFSTEN] cleared, or LLS/VLLS mode
External reference (OSCERCLK)	Up to 48 MHz (bypass), 30-40 kHz, or 3-32 MHz (crystal)	Up to 48 MHz (bypass), 30-40 kHz, or 3-32 MHz (crystal)	Up to (bypass), 30-40 kHz (low-range crystal) or Up to (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	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

Table continues on the next page...

Table 5-1. Clock summary (continued)

Clock name	HSRun mode clock frequency	Run mode clock frequency	VLPR mode clock frequency	Clock source	Clock is disabled when…
Internal 48 MHz clock (IRC48MCLK)	48 MHz	48 MHz	N/A	IRC48M	USB, MCG or SIM control does not enable and QSPI does not select it. Overriding forced disable in VLPS, LLSx, VLLSx.
RTC_CLKOUT	1 Hz or 32 kHz	1 Hz or 32 kHz	1 Hz or 32 kHz	RTC clock	RTC_CLKOUT is disabled in LLS and VLLSx modes. Overriding clocking is possible via SIM_SOPT1[OSC32KSEL] to drive CLKOUT32K out in all low power modes.
CLKOUT32K	32 kHz	32 kHz	32 kHz	ERCLK32K - which is system OSC or RTC OSC depending on SIM_SOPT1[OSC32 KSEL]	SIM_SOPT1[OSC32KSEL] not configured to drive ERCLK32K out.
LPO	1 kHz	1 kHz	1 kHz	PMC	in VLLS0
USB FS clock	48 MHz	48 MHz	N/A	MCGPLLCLK, IRC48MCLK or MCGFLLCLK with fractional clock divider, or USB_CLKIN	USB FS OTG is disabled in all Stop modes and Compute Operation mode
LPUART clock	Up to 48 MHz	Up to 48 MHz	Up to 16 MHz	MCGIRCLK, IRC48MCLK, OSCERCLK, or MCGPLLCLK/ MCGFLLCLK	SIM_SOPT2[LPUARTSRC] is cleared
FlexIO clock	Up to 96 MHz	Up to 72 MHz	Up to 4 MHz	System clock, MCGIRCLK, IRC48MCLK, OSCERCLK, or MCGPLLCLK/ MCGFLLCLK	FlexIO is disabled
EMVSIM clock	Up to 48 MHz	Up to 48 MHz	Up to 4 MHz	MCGIRCLK, IRC48MCLK, OSCERCLK, or MCGPLLCLK/ MCGFLLCLK	SIM_SOPT2[EMVSIMSRC] is cleared
TPM clock	Up to 48 MHz	Up to 48 MHz	Up to 4 MHz	MCGIRCLK, IRC48MCLK, OSCERCLK, or MCGPLLCLK/ MCGFLLCLK	SIM_SOPT2[TPMSRC] is cleared

Table continues on the next page...

Table 5-1. Clock summary (continued)

Clock name	HSRun mode clock frequency	Run mode clock frequency	VLPR mode clock frequency	Clock source	Clock is disabled when…
QSPI clocking	Up to 96 MHz [1]	Up to 96 MHz	Up to 4 MHz	MCGPLL2XCLK, MCGPLLCLK/ MCGFLLCLK, IRC48MHZ, or OSCERCLK	QSPI is disabled in all Stop modes and Compute Operation

1. QuadSPI clocks are limited to 72 MHz in dual data rate (DDR) configurations for both Run and HSRun modes.

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 96 MHz or slower in HSRun mode, 72 MHz or slower in Run mode.
2. The bus clock frequency must be programmed to 24 MHz or slower in HSRun mode, 24 MHz or slower in Run mode, and an integer divide of the core clock. The core clock to bus clock ratio is limited to a max value of 8.

NOTE

Flash erase and programming operations are not allowed in HSRun mode.

3. The flash clock frequency must be programmed to 24 MHz or less, less than or equal to the bus clock, and an integer divide of the core clock. The core clock to flash clock ratio is limited to a max value of 8.
4. The QuadSPI clock must be 72 MHz or less in DDR (dual) mode and 96 MHz or less in SDR (single) mode.

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

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 FTFA_FOPT[LPBOOT] bit controls the reset value of the core clock, system clock, bus clock, and flash clock dividers as shown below:

FTFA_FOPT [LPBOOT]	Core/system clock	Bus clock	Flash clock	Description
0	0x7 (divide by 8)	0x7 (divide by 8)	0xF (divide by 16)	Low power boot
1	0x0 (divide by 1)	0x0 (divide by 1)	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 (FTFA_FOPT[LPBOOT]) bit resides in flash is logic 1 in the flash erased state.

To enable the low power boot option program FTFA_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 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			
ARM Cortex-M0+ core	System clock	Core clock	—
NVIC	System clock	—	—
DAP	System clock	—	SWD_CLK
System modules			
DMA	System clock	—	—
DMAMUX	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	—	—
INTMUX	Bus clock	—	—
MCM	System clock	—	—
EWM	Bus clock	LPO	—
Watchdog timer	Bus clock	LPO	—
Clocks			
MCG	Flash clock	MCGOUTCLK, MCGPLLCLK, MCGFLLCLK, MCGIRCLK, OSCERCLK	—
OSC	Bus clock	OSCERCLK	—
IRC48M	—	IRC48MCLK	—
Memory and memory interfaces			
Flash controller	System clock	Flash clock	—
Flash memory	Flash clock	—	—
QSPI controller	QSPI bus interface clock	QSPI clock	QSPIx_SCK
Security			
CRC	Bus clock	—	—
TRNG	Bus clock	—	—
LTC Encryption Engine	System clock	—	—
Analog			
ADC	Bus clock	OSCERCLK, IRC48MCLK	—
CMP	Bus clock	—	—

Table continues on the next page...

Table 5-2. Module clocks (continued)

Module	Bus interface clock	Internal clocks	I/O interface clocks
DAC	Bus clock	—	—
VREF	Flash clock	—	—
Timers			
TPM	Bus clock	TPM clock	TPM_CLKIN0, TPM_CLKIN1
PDB	Bus clock	—	—
PIT	Bus clock	—	—
LPTMR	Flash clock	LPO, OSCERCLK, MCGIRCLK, ERCLK32K	—
RTC	Flash clock	EXTAL32	—
Communication interfaces			
USB FS OTG	System clock	USB FS clock	—
SPI	System clock	—	DSPI_SCK
I2C	Bus clock	—	I2C_SCL
LPUART	Bus clock	LPUART clock	—
EMVSIM	Bus clock	EMVSIM clock	—
FlexIO	Bus clock	FlexIO clock	—
Human-machine interfaces			
GPIO	Platform clock	—	—
TSI	Flash clock	LPO, ERCLK32K, MCGIRCLK	—

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 IRC 48MHz clock

The integrated 48 MHz internal reference clock source (IRC48MCLK) is available in HSRun, Run, Wait and Stop modes of operation. IRC48MCLK is also available in Compute Only, PSTOP2 and PSTOP1 modes of operation when entered from Run mode. IRC48MCLK is forced disabled when the MCU transitions into VLPS , LLSx, and VLLSx low power modes.

NOTE

IRC48MCLK is not forced disabled in Stop modes and must be disabled by software prior to Stop entry unless it is required. IRC48MCLK is not forced disabled in VLPR and must be disabled by software prior to VLPR entry.

IRC48MCLK is enabled via the following control settings while operating in these modes:

- USB Control register enables — enabled when USB_CLK_RECOVER_IRC_EN[IRC_EN]=1
- or MCG Control register selects IRC48 MHz clock — enabled when MCG_C7[OSCSEL]=10
- or SIM Control register selects IRC48 MHz clock — enabled when SIM_SOPT2[PLLFLLSEL]=11

In USB device applications, the IRC48M block can be enabled in USB clock recovery mode in which the internal IRC48M oscillator is tuned to match the clock extracted from the incoming USB data stream. This functionality provides the capability of generating a high precision 48 MHz clock source without requiring an on-chip PLL or an associated off-chip crystal circuit.

The IRC48MCLK is also available for use as:

- An oscillator reference to the MCG - from which core, system, bus, and flash clock sources can be derived
- an ADC alternate clock source
- clock source for LPUART communications

5.7.3 WDOG clocking

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

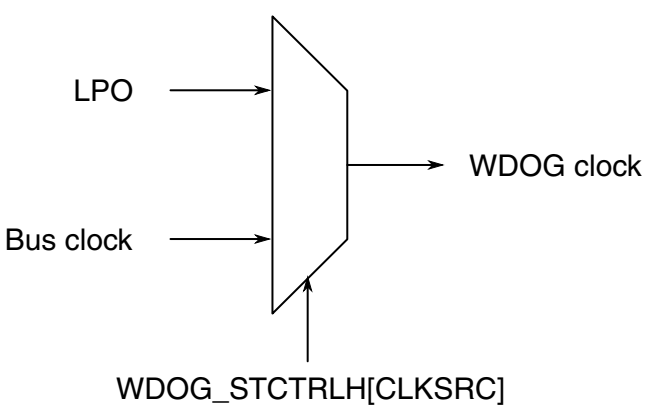


Figure 5-2. WDOG clock generation

5.7.4 PORT digital filter clocking

The digital filters in each of the PORT*x* modules 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.

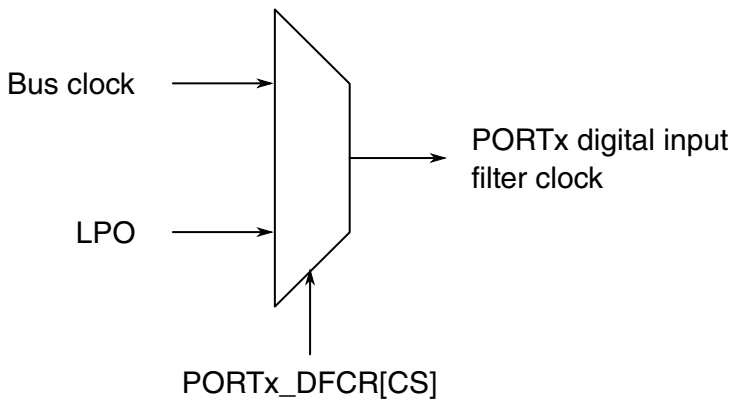


Figure 5-3. PORTx digital input filter clock generation

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.

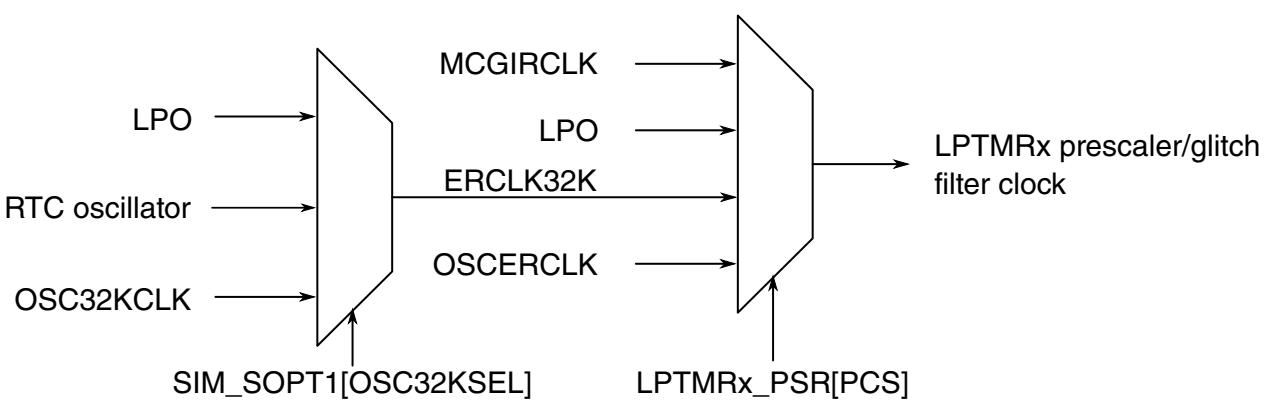


Figure 5-4. LPTMRx prescaler/glitch filter clock generation

5.7.6 TPM clocking

The counter for the TPM modules have a selectable clock as shown in the following figure.

NOTE

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

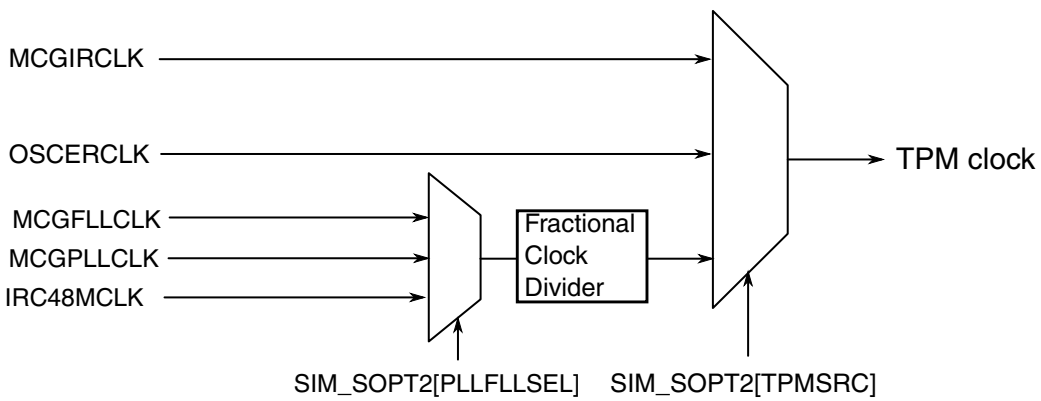


Figure 5-5. TPM clock generation

5.7.7 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.

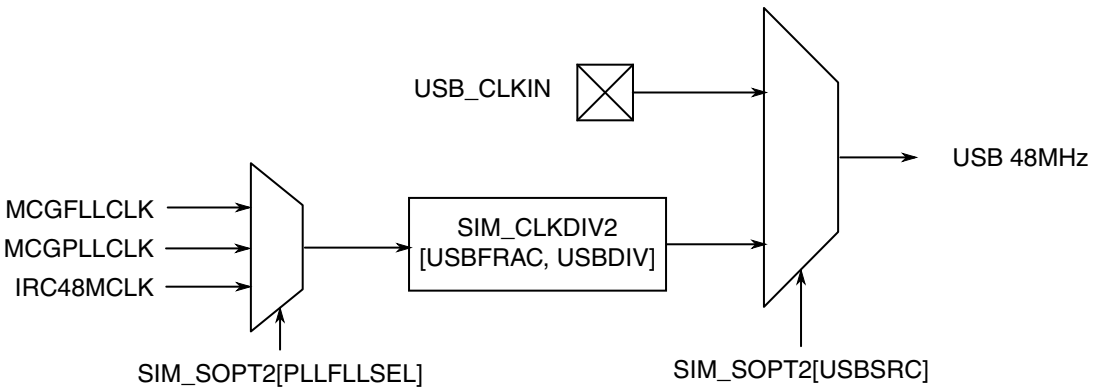


Figure 5-6. USB 48 MHz clock source

NOTE

The MCGFLLCLK does not meet the USB jitter specifications for certification. The IRC48MCLK is only usable as a USB clock source in USB Device operation with the USB Clock Recover function enabled.

5.7.8 LPUART clocking

Each LPUART module has a selectable clock as shown in the following figure.

NOTE

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

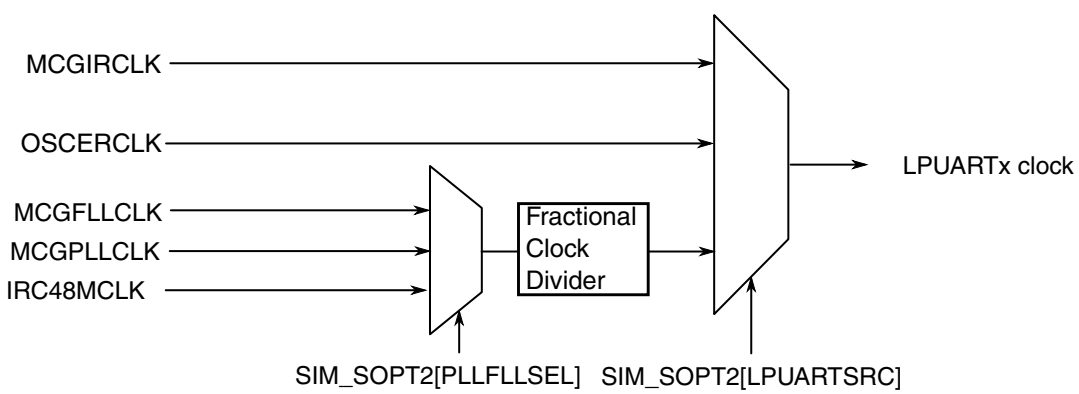


Figure 5-7. LPUART clock generation

5.7.9 QSPI clocking

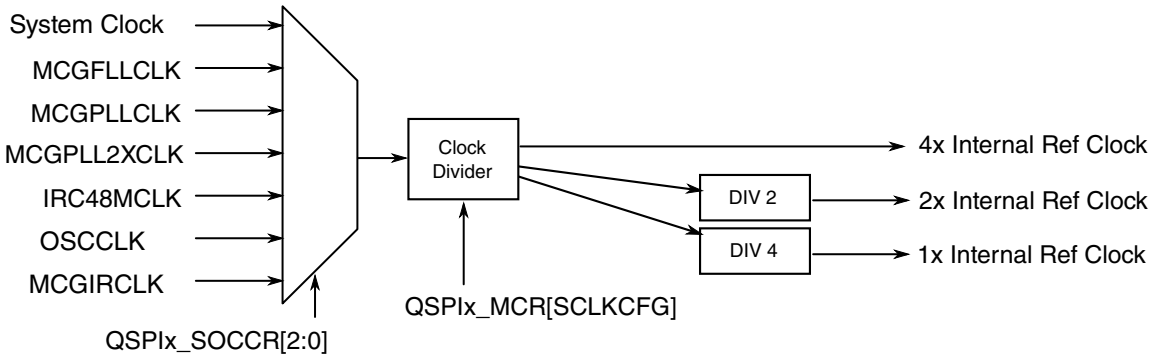


Figure 5-8. QSPI clock generation

DDR mode of the QSPI requires a 4x, 2x, and 1x internal reference clock. In DDR mode, the clock divider output is used as the 4x internal reference clock. The 2x and 1x clocks are divided down from that clock. The maximum flash frequency supported is 100 MHz SDR and 75 MHz DDR, which requires MCGPLL2XCLK to achieve the 4x speed in DDR mode.

5.7.10 LP Trusted Cryptography (LTC) clocking

LTC is clocked from the system clock.

5.7.11 TRNG clocking

TRNG is clocked from the bus clock.

5.7.12 FlexIO clocking

The FlexIO timers and shifters have a selectable clock as shown in the following figure.

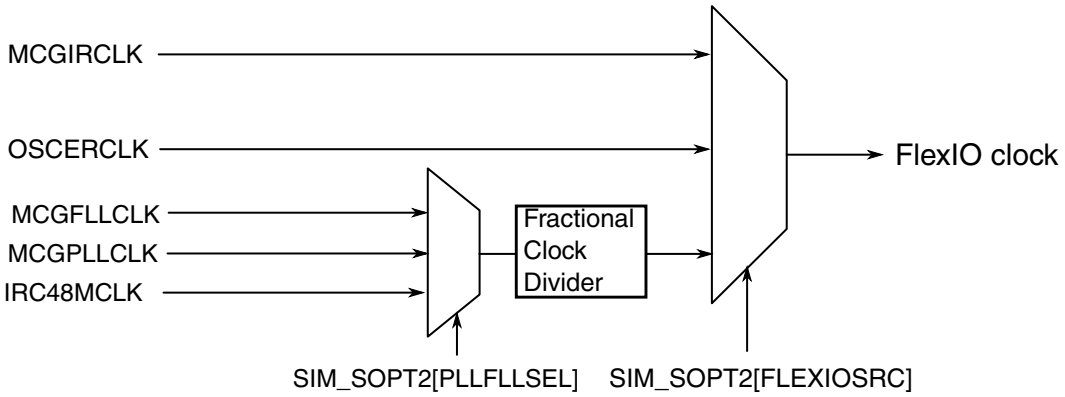


Figure 5-9. FlexIO clock generation

NOTE

The chosen clock must remain enabled if FlexIO is to continue operation in required low-power modes.

5.7.13 EMVSIM clocking

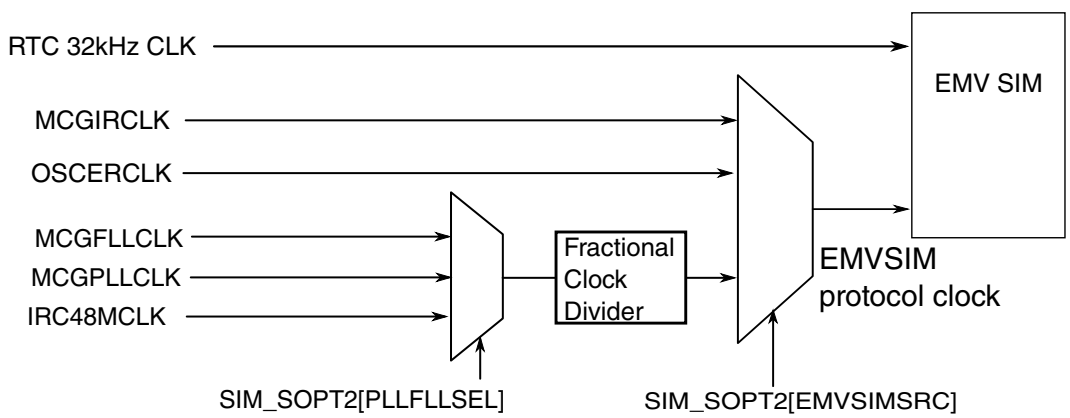


Figure 5-10. EMVSIM clock generation

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) • Low-voltage detect (LVD) • Computer operating properly (COP) watchdog reset • Low leakage wakeup (LLWU) reset • Multipurpose clock generator loss of clock (LOC) reset • Multipurpose clock generator loss of lock (LOL) reset • Stop mode acknowledge error (SACKERR) • Software reset (SW) • Lockup reset (LOCKUP) • MDM DAP system reset
Debug reset	• Debug 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 where the CPU is executing code. 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 V_{DD} supply has risen above the LVD low threshold (V_{LVDL}). The POR and LVD bits in SRS0 register are set following a POR.

V_{DDIO} must be powered up after V_{DD}, and powered down prior to V_{DD} power down. In other words, V_{DDIO} cannot be powered when V_{DD} is off.

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.

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 $\overline{\text{RESET}}$ pin filter

The $\overline{\text{RESET}}$ pin filter supports filtering from both the 1 kHz LPO clock and the bus clock. RCM_RPFC[RSTFLTSS], RCM_RPFC[RSTFLTSRW], and RCM_RPFW[RSTFLTSEL] control this functionality; see the RCM chapter. The filters are asynchronously reset by Chip POR. The reset value for each filter assumes the $\overline{\text{RESET}}$ pin is negated.

For all stop modes where LPO clock is still active (Stop, VLPS, LLS, VLLS3, VLLS2, and VLLS1), the only filtering option is the LPO-based digital filter. The filtering logic either switches to bypass operation or has continued filtering operation depending on the filtering mode selected. When entering VLLS0, the $\overline{\text{RESET}}$ 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.

6.2.2.2 Low-voltage detect (LVD)

The chip includes a system for managing low voltage conditions on the V_{DD} domain 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 hsrun, 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.

NOTE

V_{DDIO} domain does not contain LVD circuit.

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_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 MCG_C6[CME0] field is set, the clock monitor is enabled. If the external reference falls below f_{loc_low} or f_{loc_high}, as controlled by the MCG_C2[RANGE] field, the MCU resets. The RCM_SRS0[LOC] field is set to indicate this reset source.

NOTE

To prevent unexpected reset events, all clock monitors must 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_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_SRS1[LOCKUP] bit to set.

6.2.2.10 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 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 RESET_b 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_b 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 $\overline{\text{RESET}}$ pin, the $\overline{\text{RESET}}$ 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 $\overline{\text{RESET}}$ pin is released, and the internal Chip Reset negates after the $\overline{\text{RESET}}$ pin is pulled high. Keeping the $\overline{\text{RESET}}$ 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 Resetting the Debug subsystem

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

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

- SW-DP
- AHB-AP
- MDM-AP (MDM control and status registers)

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

- CM0+ core (core debug registers: DHCSR, DCRSR, DCRDR, DEMCR)
- BPU
- DWT
- NVIC
- Crossbar bus switch[1]
- Private peripheral bus[1]

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.

Based on the source selections in FOPT, the device boots initially to internal Flash or internal BOOT ROM. If BOOT ROM, the device executes in boot loader mode or proceeds with a secondary boot to a QSPI device connected to QSPI0.

6.3.2 Boot options

Because this device has BOOT ROM and no dedicated BOOT mode pin, the device's functional mode is controlled by the BOOT ROM reading the flash option boot source selection located within FOPT register.

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.

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

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

NOTE

Reserved bits in the option byte should be left in their default erased state of logic 1. FOPT[7:0] = 0x00 is not a valid configuration. FOPT register is written to 0xFF if the contents of NVM's option byte in the flash configuration field is 0x00.

Table 6-2. Flash Option Register Bit Definitions

Bit Num	Field	Value	Definition
7-6	BOOTSRC_SEL	Boot Source Selection: these bits select the boot sources if boot pin option bit BOOTPIN_OPT = 1	
		00	Boot from Internal Flash
		01	Reserved
		10	Boot from ROM, configure QSPI0, and enter boot loader mode. See Boot sequence for more information
		11	Boot from ROM and enter boot loader mode. See Boot sequence for more information.
5	FAST_INIT	Select initialization speed on POR, VLLSx, and any system reset.	
		0	Slower initialization. The Flash initialization will be slower with the benefit of reduced average current during this time. The duration of the recovery will be controlled by the clock divider selection determined by the LPBOOT setting.
		1	Fast Initialization. The Flash has faster recoveries at the expense of higher current during these times.
4-3	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	BOOTPIN_OPT	External pin selects boot options	
		0	Force Boot from ROM with update if BOOTCFG0 asserted, where BOOTCFG0 is the boot config function which is muxed with NMI pin. The RESET pin should be enabled when this option is selected.
		1	Boot source configured by FOPT[7:6] (BOOTSRC_SEL) bits
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	Core and system clock divider (OUTDIV1) is 0x7 (divide by 8). Device is configured for VLPR mode on exit from reset.
		1	Core and system clock divider (OUTDIV1) is 0x0 (divide by 1). Device is configured for RUN mode on exit from reset.

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.

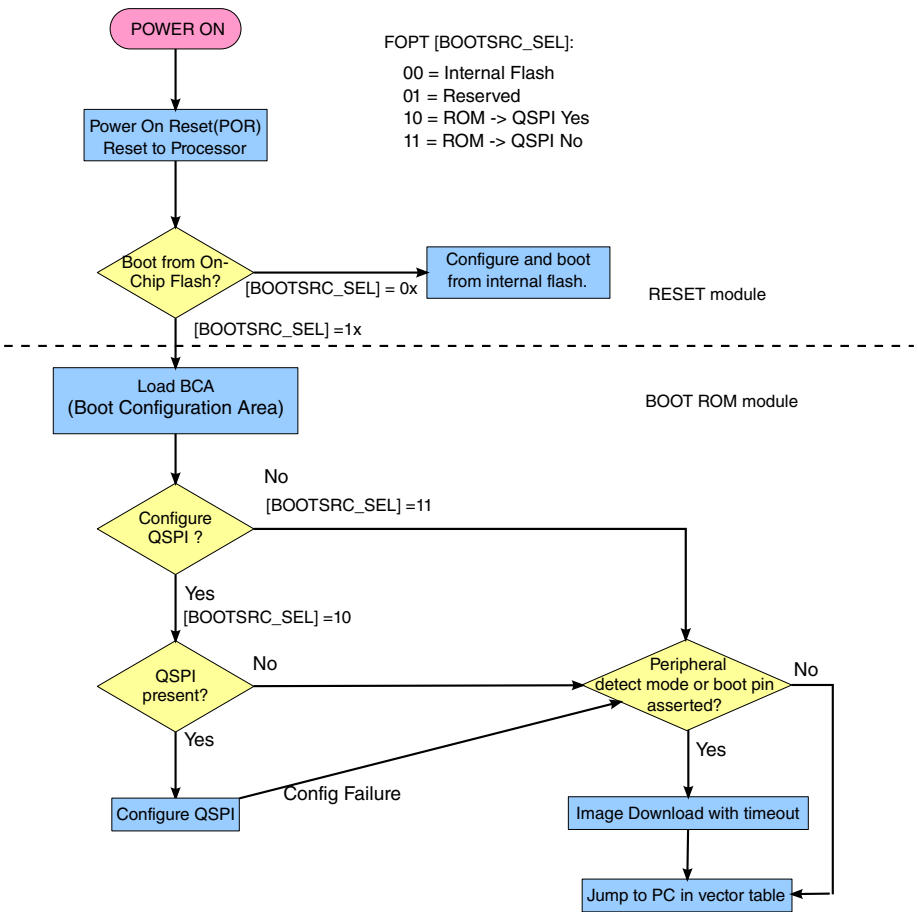


Figure 6-1. Boot Flow For Devices with QSPI

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 reset to disabled).
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 Reset Control logic continues to drive the $\overline{\text{RESET}}$ pin out low.
4. Early in reset sequencing the NVM option byte is read and stored to the Flash Memory module's FOPT register. Based on the following values, the corresponding actions are taken:

a. If the LPBOOT bit is programmed for an alternate clock divider reset value, the system/core clock is switched to a slower clock speed.
b. If the FAST_INIT bit is programmed clear, the Flash initialization switches to slower clock resulting longer recovery times.
c. If the BOOTPIN_OPT bit is set, the boot source of the device is determined by BOOTSRC_SEL field.
d. The BOOTSRC_SEL field determines if the device boots from internal flash, directly from the internal ROM to allow bootloading of firmware, or configuring the QSPI to allow booting from QSPI device.

5. When Flash Initialization completes, the $\overline{\text{RESET}}$ pin is released. If $\overline{\text{RESET}}$ continues to be asserted (an indication of a slow rise time on the $\overline{\text{RESET}}$ pin or external drive in low), the system continues to be held in reset. Once the $\overline{\text{RESET}}$ pin is detected high, the Core clock is enabled and the system is released from reset.
6. If the BOOTSRC_SEL bits indicate a boot from internal flash, 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.

Subsequent system resets follow this same reset flow.

Chapter 7
Kinetis ROM Bootloader

7.1 Chip-Specific Information

This device has various peripherals supported by the Kinetis ROM Bootloader.

Table 7-1. Kinetis Bootloader Peripheral Pinmux

Peripheral	Instance	Port	GPIO	Alt
LPUART	0	LPUART0_RX	PTB16	3
		LPUART0_TX	PTB17	3
	1	LPUART1_RX	PTC3	3
		LPUART1_TX	PTC4	3
	2	LPUART2_RX	PTD2	3
		LPUART2_TX	PTD3	3
I2C	1	I2C1_SCL	PTC10	2
		I2C1_SDA	PTC11	2
Secondary I2C slave address pin		—	PTA1	1
SPI	1	SPI1_PCS0	PTD4	7
		SPI1_SCK	PTD5	7
		SPI1_SOUT	PTD6	7
		SPI1_Sin	PTD7	7
QuadSPI	0	QSPI0A_DATA3	PTE0	5
		QSPI0A_SCLK	PTE1	5
		QSPI0A_DATA0	PTE2	5
		QSPI0A_DATA2	PTE3	5
		QSPI0A_DATA1	PTE4	5
		QSPI0A_SS0_B	PTE5	5
		QSPI0A_SS1_B	PTE7	7
		QSPI0A_DQS	PTE11	7
		QSPI0B_DATA3	PTE6	5
		QSPI0B_SCLK	PTE7	5

Table continues on the next page...

Table 7-1. Kinetis Bootloader Peripheral Pinmux (continued)

Peripheral	Instance	Port	GPIO	Alt
		QSPI0B_DATA0	PTE8	5
		QSPI0B_DATA2	PTE9	5
		QSPI0B_DATA1	PTE10	5
		QSPI0B_SS0_B	PTE11	5
CRC status pin		—	PTC5	1

In addition to the pins listed in the above table, the bootloader uses GPIO PTA11 to indicate a CRC check status failure.

NOTE

Not all QuadSPI pins are supported in all packages, see KL82 signal multiplexing and pin assignments for detailed information.

7.2 Introduction

The Kinetis bootloader is the program residing in the on-chip read-only memory (ROM) of a Kinetis microcontroller device. There is hardware logic in place at boot time that either starts execution of an embedded image available on the internal flash memory, or starts the execution of the Kinetis Bootloader from on-chip ROM.

The Kinetis Bootloader's main task is to provision the internal flash memory with an embedded firmware image during manufacturing, or at any time during the life of the Kinetis device. The Kinetis Bootloader does the provisioning by acting as a slave device, and listening to various peripheral ports where a master can start communication.

For the Kinetis device, the Kinetis Bootloader can interface with USB, I2C, SPI, and LPUART peripherals in slave mode and respond to the commands sent by a master (or host) communicating on one of those ports. The host/master can be a firmware-download application running on a PC or an embedded host communicating with the Kinetis Bootloader. Regardless of the host/master (PC or embedded host), the Kinetis Bootloader always uses a command protocol to communicate with that host/master. Commands are provided to write to memory (internal flash or RAM), erase flash, and get/set bootloader options and property values. The host application can query the set of available commands.

On start-up, the bootloader reads optional configuration parameters from a fixed area on flash called the bootloader configuration area (BCA). These parameters can be modified by the write memory command or by downloaded flash image. BCA parameters include configuration data such as enabled peripherals, peripheral-specific settings, etc.

This chapter describes Kinetis Bootloader features, functionality, command structure and which peripherals are supported.

Features supported by the Kinetis Bootloader in Kinetis ROM:

- Supports USB, I2C, SPI, and LPUART peripheral interfaces
- Automatic detection of the active peripheral
- Ability to disable any peripheral
- LPUART peripheral implements autobaud
- Common packet-based protocol for all peripherals
- Packet error detection and retransmission
- Flash-resident configuration options
- Fully supports internal flash security, including ability to mass erase or unlock security via the backdoor key
- Protection of RAM used by the bootloader while it is running
- Provides command to read properties of the device, such as flash and RAM size
- Multiple options for executing the bootloader either at system start-up or under application control at runtime

Table 7-2. Commands supported by the Kinetis Bootloader in ROM

Command	Description	When flash security is enabled, then this command is
Call	Runs user application code and returns control to bootloader	Not supported
Execute	Run user application code that never returns control to the bootloader	Not supported
FillMemory	Fill a range of bytes in flash with a word pattern	Not supported
FlashEraseAll	Erase the entire flash array	Not supported
FlashEraseRegion	Erase a range of sectors in flash	Not supported
WriteMemory	Write data to memory	Not supported
ReadMemory	Read data from memory	Not supported
FlashProgramOnce	Writes data provided in a command packet to a specified range of bytes in the program once field	Not supported
FlashReadOnce	Returns the contents of the program once field by given index and byte count	Not supported
FlashReadResource	Returns the contents of the IFR field or Flash firmware ID, by given offset, byte count and option	Not supported
ConfigureQuadSpi	Configure the QuadSPI with a programmed configuration block	Supported

Table continues on the next page...

Table 7-2. Commands supported by the Kinetis Bootloader in ROM (continued)

Command	Description	When flash security is enabled, then this command is
FlashSecurityDisable	Attempt to unlock flash security using the backdoor key	Supported
GetProperty	Get the current value of a property	Supported
ReceiveSbFile	Receive an SB file (which is generated by the elftosb tool)	Supported if the SB file is encrypted. See the SB File Decryption Support section for more details.
Reset	Reset the chip	Supported
SetProperty	Attempt to modify a writable property	Supported
FlashEraseAllUnsecure	Erase the entire flash array, including protected sectors	Supported

7.3 Functional Description

The following sub-sections describe the Kinetis Bootloader in KL82 ROM functionality.

7.3.1 Memory Maps

While executing, the Kinetis Bootloader uses ROM and RAM memory.

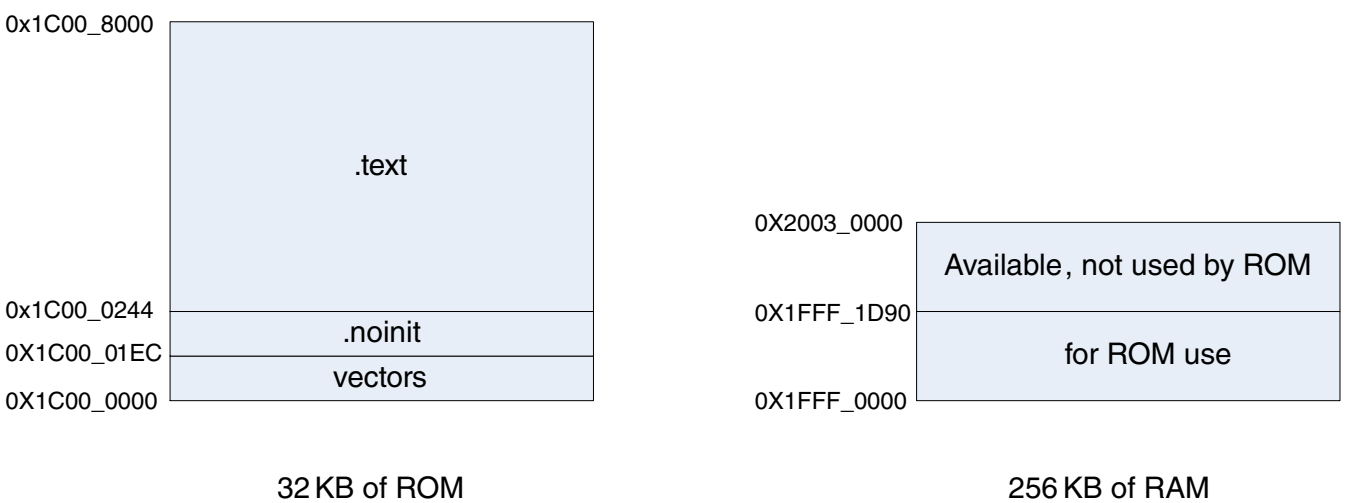


Figure 7-1. Kinetis Bootloader ROM/RAM Memory Maps

7.3.2 The Kinetis Bootloader Configuration Area (BCA)

The Kinetis Bootloader reads data from the Bootloader Configuration Area (BCA) to configure various features of the bootloader. The BCA resides in flash memory at offset 0x3C0, and provides all of the parameters needed to configure the Kinetis Bootloader operation. For uninitialized flash, the Kinetis Bootloader uses a predefined default configuration. A host application can use the Kinetis Bootloader to program the BCA for use during subsequent initializations of the bootloader.

Table 7-3. Configuration Fields for the Kinetis Bootloader

Offset	Size (bytes)	Configuration Field	Description
0x00 - 0x03	4	tag	Magic number to verify bootloader configuration is valid. Must be set to 'kcfg'.
0x04 - 0x07	4	crcStartAddress	Start address for application image CRC check. To generate the CRC, refer to the CRC chapter. If the bits are all set then Kinetis bootloader by default will not perform any CRC check.
0x08 - 0x0B	4	crcByteCount	Byte count for application image CRC check. If the bits are all set then Kinetis bootloader by default will not perform any CRC check.
0x0C - 0x0F	4	crcExpectedValue	Expected CRC value for application CRC check. If the bits are all set then Kinetis bootloader by default will not perform any CRC check.
0x10	1	enabledPeripherals	Bitfield of peripherals to enable. bit 0 LPUART bit 1 I2C bit 2 SPI bit 4 USB Kinetis bootloader will enable the peripheral if corresponding bit is set to 1.
0x11	1	i2cSlaveAddress	If not 0xFF, used as the 7-bit I2C slave address. If 0xFF, defaults to 0x10 for I2C slave address
0x12 - 0x13	2	peripheralDetectionTimeout	Timeout in milliseconds for active peripheral detection. If 0xFFFF, defaults to 5 seconds.
0x14 - 0x15	2	usbVid	Sets the USB Vendor ID reported by the device during enumeration. If 0xFFFF, it defaults to 0x15A2.
0x16- 0x17	2	usbPid	Sets the USB Product ID reported by the device during enumeration.

Table continues on the next page...

Table 7-3. Configuration Fields for the Kinetis Bootloader (continued)

Offset	Size (bytes)	Configuration Field	Description
0x18 - 0x1B	4	usbStringsPointer	Sets the USB Strings reported by the device during enumeration. Default string values are described in the USB peripheral section.
0x1C	1	clockFlags	See Table 7-5, clockFlags Configuration Field
0x1D	1	clockDivider	Inverted value of the divider to use for core and bus clocks when in high speed mode
0x1E	1	bootFlags	If bit 0 is cleared, then Kinetis bootloader will jump to either Quad SPI Flash or internal flash image depending on FOPT BOOTSRC_SEL bits. If the bit is set, then Kinetis bootloader will prepare for host communication over serial peripherals.
0x1F	1	pad byte	N/A
0x20	4	Reserved	-
0x24	4	keyblob data pointer	A pointer to the keyblob data in memory. See the encryption section for details.
0x28	8	Reserved	-
0x30	4	qspiConfigBlockPointer	A pointer to the QSPI config block in internal flash array.
0x34	12	Reserved	-

NOTE

The flash sector containing the BCA should not be located in the execute-only region, because the Kinetis bootloader cannot read an execute-only region.

The whole BCA region must be filled with 0xFFs if BCA is not enabled

The first configuration field 'tag' is a tag value or magic number. The tag value must be set to 'kcfg' for the bootloader configuration data to be recognized as valid. If tag-field verification fails, then the Kinetis Bootloader assumes that the flash is not initialized and uses a predefined default configuration. The tag value is treated as a character string, so bytes 0-3 must be set as shown in the table.

Table 7-4. tag Configuration Field

Offset	tag Byte Value
0	'k' (0x6B)
1	'c' (0x63)
2	'f' (0x66)
3	'g' (0x67)

The flags in the clockFlags configuration field are enabled if the corresponding bit is cleared (0).

Table 7-5. clockFlags Configuration Field

Bit	Flag	Description
0	HighSpeed	Enable high speed mode (i.e., 48 MHz).
1 - 7	Reserved	

7.3.3 Start-up Process

Any of the following conditions will force the hardware to start the Kinetis Bootloader:

- BOOTSRC_SEL field of FOPT register is set to either 0b11 or 0b10. This forces the ROM to run out of reset.
- The BOOTCFG0 pin is asserted. The pin must be configured as BOOTCFG0 by setting the BOOTPIN_OPT bit of FOPT to 0.
- A user applications running on flash or RAM calls into the Kinetis Bootloader entry point address in ROM, to start Kinetis Bootloader execution.

The BOOTSRC_SEL bits (FOPT register, FOPT [7:6]) determine the boot source. The FOPT register is located in the flash configuration field at address 0x40D in the flash memory array. For a complete list of options, see the Boot options section in the Reset and Boot chapter. If BOOTSRC_SEL is set to 0b11 or 0b10, then the device will boot to ROM out of reset. Flash memory defaults to all 1s when erased, so a blank chip will automatically boot to ROM.

The BOOTCFG0 pin is shared with the NMI pin, with NMI being the default usage. Regardless of whether the NMI pin is enabled or not, the NMI functionality is disabled if the ROM is executed out of reset, for as long as the ROM is running.

When the ROM is executed out of reset, vector fetches from the CPU are redirected to the ROM's vector table in ROM memory at offset 0x1C00_0000. This ensures that any exceptions will be handled by the ROM.

After the Kinetis Bootloader has started, the following procedure starts bootloader operations:

1. The RCM_MR [FORCEROM] bits are set, so that the device will reboot back into the ROM if/when the device is reset.
2. Initializes the bootloader's .data and .bss sections.
3. Reads bootloader configuration data from flash at address 0x3C0. The configuration data is only used if the tag field is set to the expected 'kcfg' value. If the tag is incorrect, then the configuration values are set to default, as if the data was all 0xFF bytes.
4. Clocks are configured. See the Clock Configuration section.
5. If BOOTSRC_SEL is set to 0b10, then the QSPI0 interface is configured by reading the configuration block from the base address of the external QuadSPI, or from the address pointed by the qspiConfigBlockPointer in the BCA.
6. Enabled peripherals are initialized.
7. The bootloader waits for communication to begin on a peripheral.
   - If detection times out, then the bootloader jumps to the user application in flash. See Bootloader Exit state section.
   - If communication is detected, then all inactive peripherals are shut down, and the command phase is entered.

NOTE

The flash sector containing the vector table should not be located in the execute-only region, because the Kinetis bootloader cannot read the PC and SP addresses in an execute-only region.

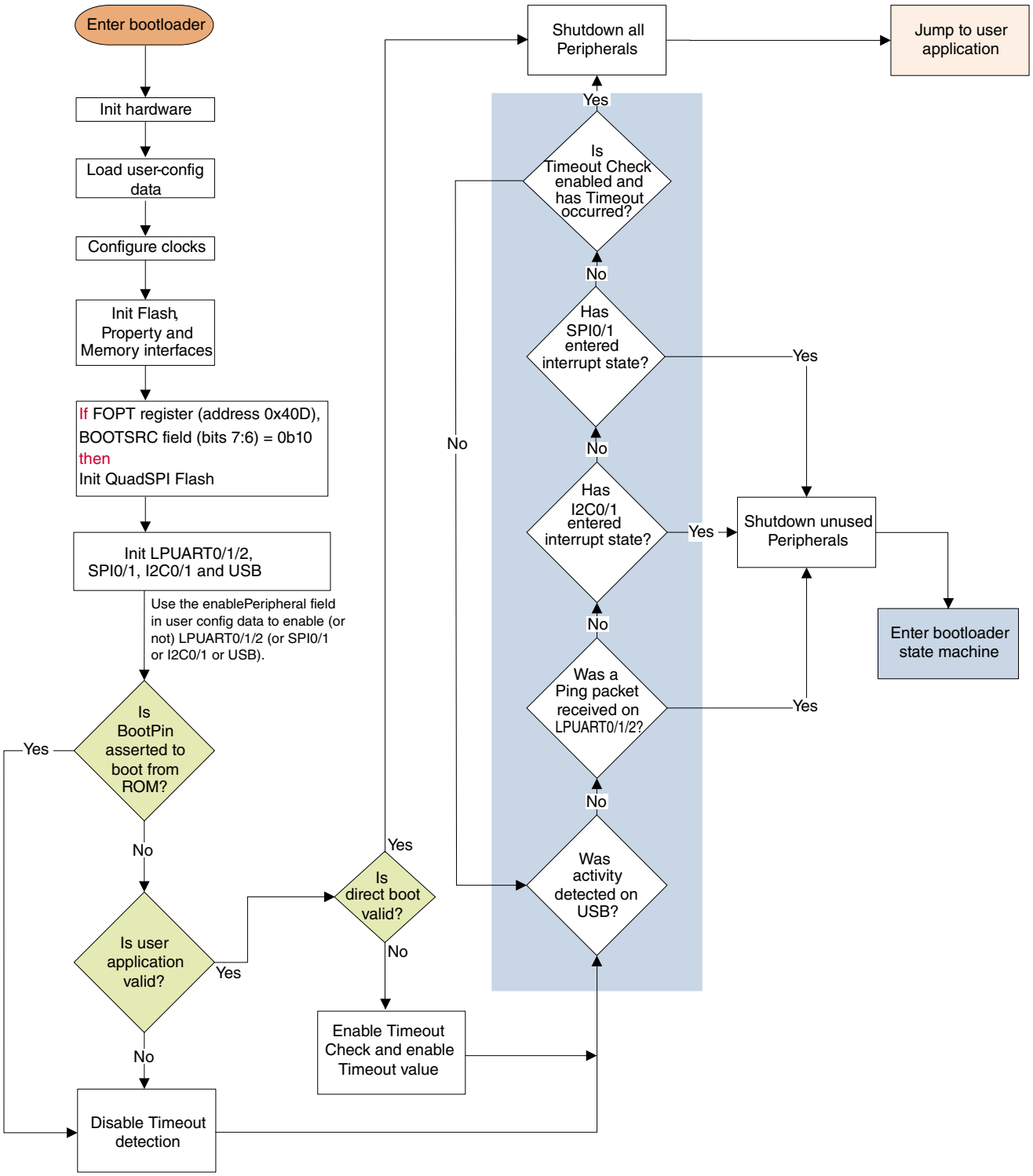


Figure 7-2. Kinetis Bootloader Start-up Flowchart

7.3.4 Clock Configuration

By default, the bootloader does not modify clocks. The Kinetis Bootloader in ROM will use the clock configuration of the chip out of reset unless the clock configuration bits in the FOPT register are cleared, or if a USB peripheral is enabled.

- Alternate clock configurations are supported, by setting fields in the Bootloader Configuration Area (BCA) shown in Table 7-3.
- If the HighSpeed flag of the clockFlags configuration value is cleared, or if a USB peripheral is enabled, the bootloader will enable the internal 48 MHz reference clock.
- In high speed mode, the core and bus clock frequencies are determined by the clockDivider configuration value.
- The core clock divider is set directly from the inverted value of the clockDivider unless a USB peripheral is enabled. If a USB peripheral is enabled and the inverted value of the clockDivider is greater than 2, then the value is reduced to 2, in order to keep the CPU clock above 20 MHz; if the QuadSPI is configured at start-up, then the core clock will be set to 72 MHz and the clock divider will be ignored.
- The bus clock divider is set to 1, unless the resulting bus clock frequency would be greater than the maximum supported value. In this case, the bus clock divider is increased until the bus clock frequency is at or below the maximum.
- Note that the maximum baud rate of serial peripherals is related to the core and bus clock frequencies. To achieve the desired baud rates, high speed mode should be enabled in BCA.

7.3.5 Bootloader Entry Point

The Kinetis Bootloader provides a function (runBootloader) that a user application can call, to run the bootloader.

To get the address of the entry point, the user application reads the word containing the pointer to the bootloader API tree at offset 0x1C of the bootloader's vector table. The vector table is placed at the base of the bootloader's address range, which for the ROM is 0x1C00_0000. Thus, the API tree pointer is at address 0x1C00_001C.

The bootloader API tree is a structure that contains pointers to other structures, which have the function and data addresses for the bootloader. The bootloader entry point is always the first word of the API tree.

The prototype of the entry point is:

```
void run_bootloader(void * arg);
```

The arg parameter is currently unused, and is intended for future expansion (for example, passing options to the bootloader). To ensure future compatibility, a value of NULL should be passed for arg.

Example code to get the entry pointer address from the ROM and start the bootloader.

NOTE

This entry must be called in supervisor (privileged) mode.

```
// Variables

uint32_t runBootloaderAddress;

void (*runBootloader)(void * arg);


// Read the function address from the ROM API tree.

runBootloaderAddress = **(uint32_t **)(0x1c00001c);

runBootloader = (void (*)(void * arg))runBootloaderAddress;


// Start the bootloader.

runBootloader(NULL);
```

7.3.6 Bootloader Protocol

This section explains the general protocol for the packet transfers between the host and the Kinetis Bootloader. The description includes the transfer of packets for different transactions, such as commands with no data phase and commands with incoming or outgoing data phase. The next section describes various packet types used in a transaction.

Each command sent from the host is replied to with a response command.

Commands may include an optional data phase:

- If the data phase is **incoming** (from host to bootloader), then the data phase is part of the **original command**.
- If the data phase is **outgoing** (from bootloader to host), then the data phase is part of the **response command**.

NOTE

In all protocols (described in the next subsections), the Ack sent in response to a Command or Data packet can arrive at any time *before, during, or after* the Command/Data packet has processed.

7.3.6.1 Command with no data phase

The protocol for a command with no data phase contains:

- Command packet (from host)
- Generic response command packet (to host)

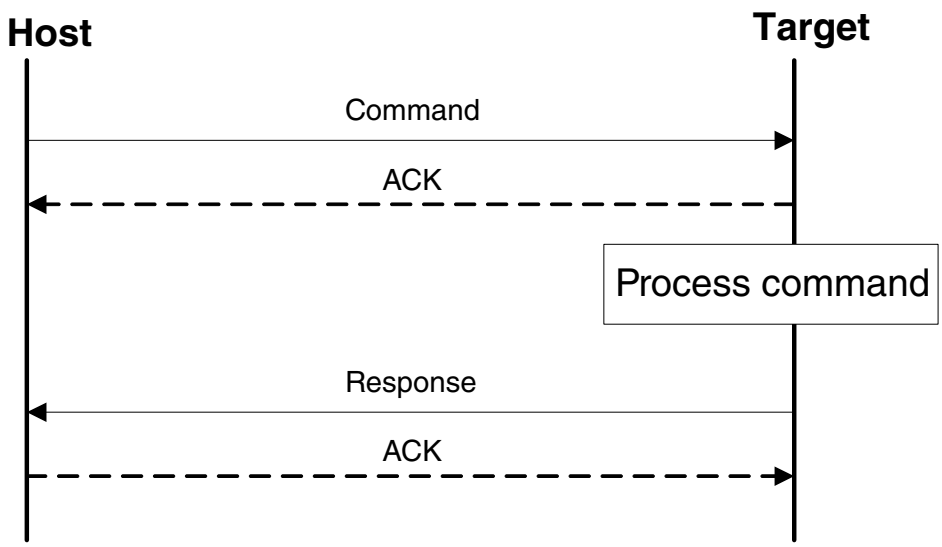


Figure 7-3. Command with No Data Phase

7.3.6.2 Command with incoming data phase

The protocol for a command with an incoming data phase contains:

- Command packet (from host)
- Generic response command packet (to host)
- Incoming data packets (from host)
- Generic response command packet (to host)

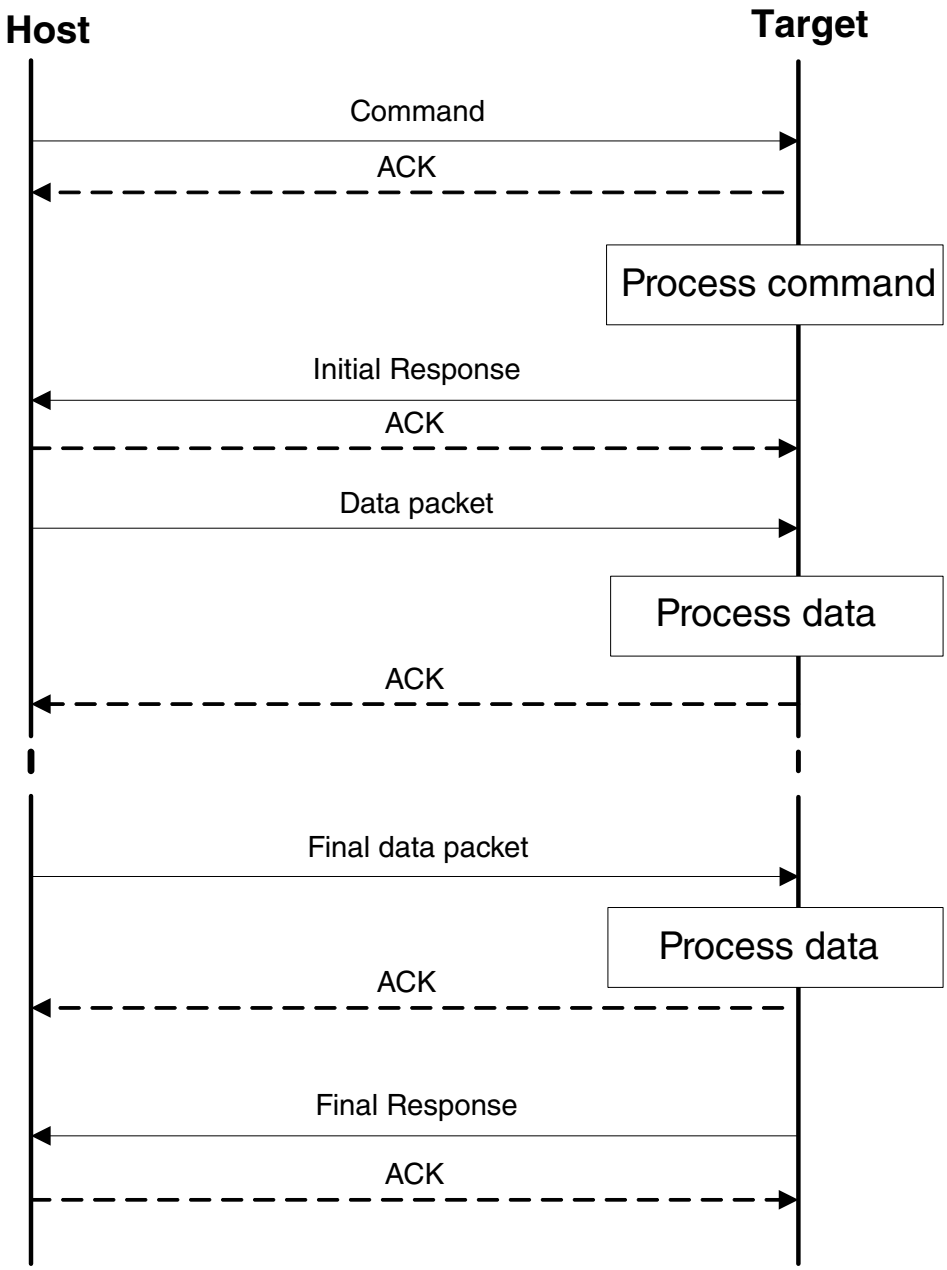


Figure 7-4. Command with incoming data phase

NOTE

- The host may not send any further packets while it (the host) is waiting for the response to a command.
- If the Generic Response packet prior to the start of the data phase does not have a status of kStatus_Success, then the data phase is aborted.
- Data phases may be aborted by the receiving side by sending the final Generic Response early with a status of

kStatus_AbortDataPhase. The host may abort the data phase early by sending a zero-length data packet.
- The final Generic Response packet *sent after the data phase* includes the status for the entire operation.

7.3.6.3 Command with outgoing data phase

The protocol for a command with an outgoing data phase contains:
- Command packet (from host)
- ReadMemory Response command packet (to host) (kCommandFlag_HasDataPhase set)
- Outgoing data packets (to host)
- Generic response command packet (to host)

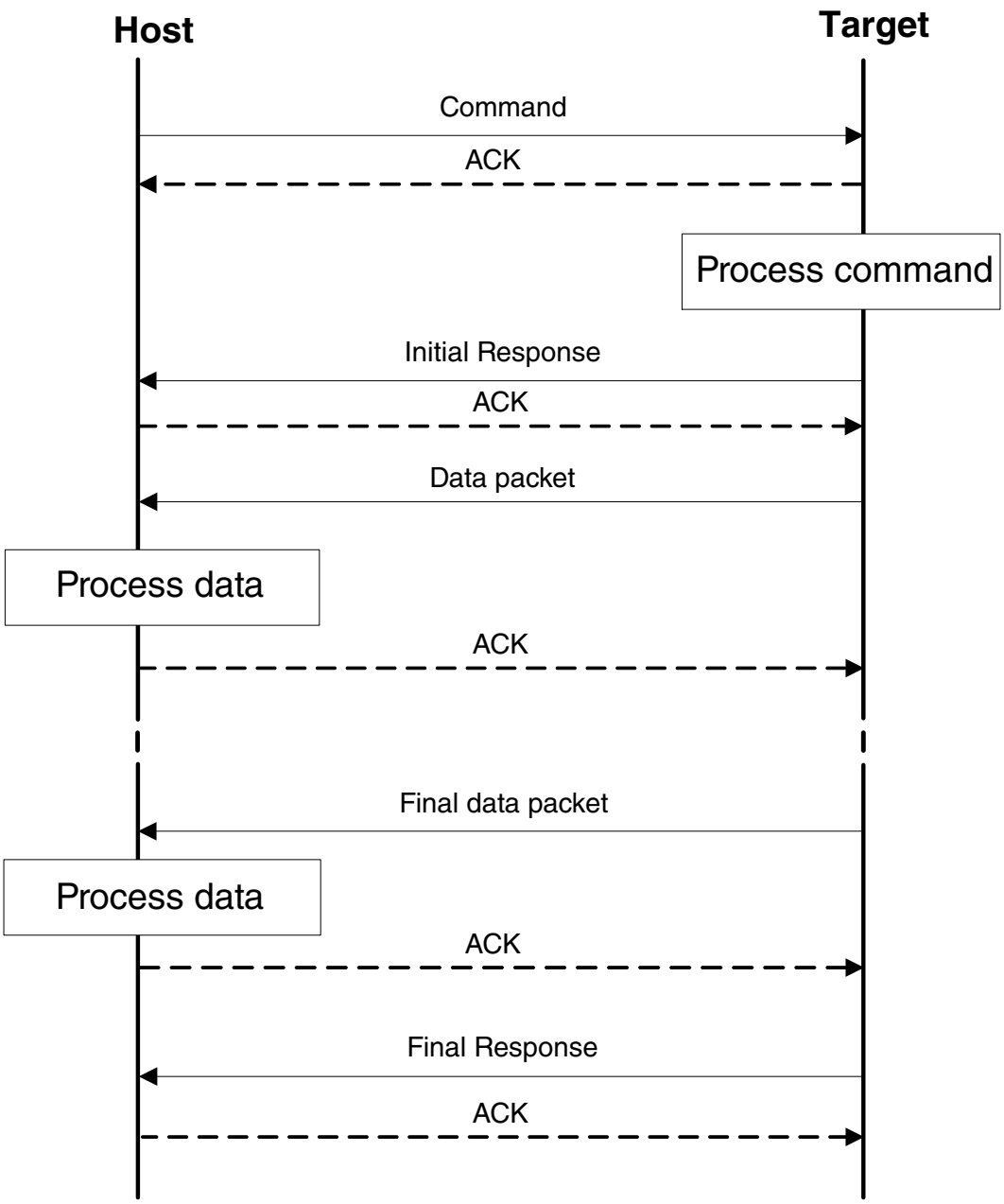


Figure 7-5. Command with outgoing data phase

NOTE

- For the outgoing data phase sequence above, the data phase is really considered part of the response command.
- The host may not send any further packets while it (the host) is waiting for the response to a command.
- If the ReadMemory Response command packet prior to the start of the data phase does not contain the kCommandFlag_HasDataPhase flag, then the data phase is aborted.

- Data phases may be aborted by the host sending the final Generic Response early with a status of kStatus_AbortDataPhase. The sending side may abort the data phase early by sending a zero-length data packet.
- The final Generic Response packet *sent after the data phase* includes the status for the entire operation.

7.3.7 Bootloader Packet Types

The Kinetis Bootloader device works in slave mode. All data communication is initiated by a host, which is either a PC or an embedded host (note that QuadSPI is treated like a storage device, like flash). The Kinetis Bootloader device is the target, which receives a command or data packet. All data communication between host and target is packetized.

NOTE

The term "target" refers to the "Kinetis Bootloader device."

There are 6 types of packets used in the device:

- Ping packet
- Ping Response packet
- Framing packet
- Command packet
- Data packet
- Response packet

All fields in the packets are in little-endian byte order.

7.3.7.1 Ping packet

The Ping packet is the first packet sent from a host to the target (Kinetis Bootloader), to establish a connection on a selected peripheral. For a LPUART peripheral, the Ping packet is used to determine the baudrate. A Ping packet must be sent before any other communications. In response to a Ping packet, the target sends a Ping Response packet.

Table 7-6. Ping Packet Format

Byte #	Value	Name
0	0x5A	start byte
1	0xA6	ping

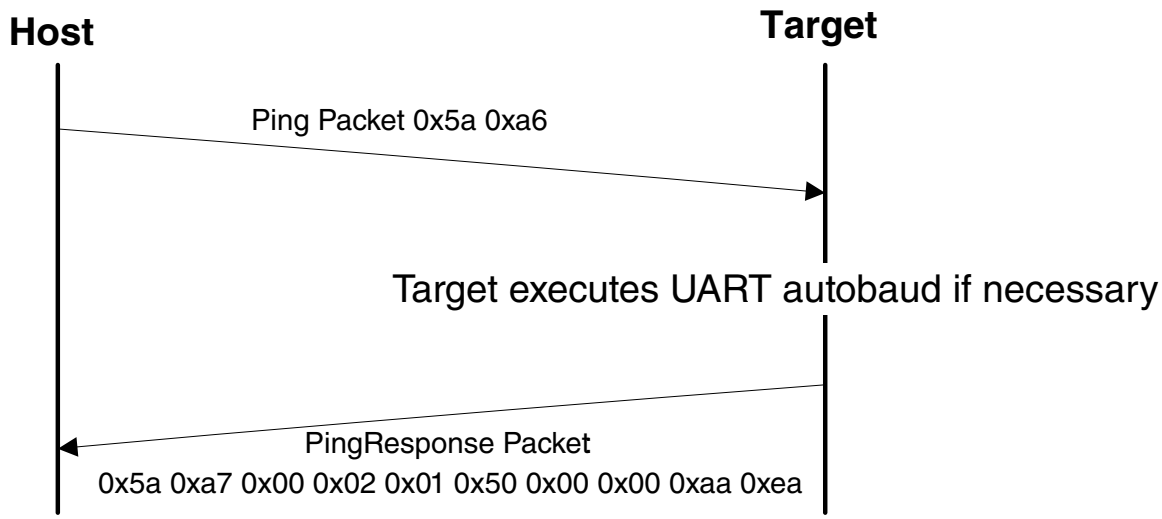


Figure 7-6. Ping Packet Protocol Sequence

7.3.7.2 Ping Response Packet

The target (Kinetis Bootloader) sends a Ping Response packet back to the host after receiving a Ping packet. If communication is over a LPUART peripheral, the target uses the incoming Ping packet to determine the baud rate before replying with the Ping Response packet. Once the Ping Response packet is received by the host, the connection is established, and the host starts sending commands to the target (Kinetis Bootloader).

Table 7-7. Ping Response Packet Format

Byte #	Value	Parameter
0	0x5A	start byte
1	0xA7	Ping response code
2		Protocol bugfix
3		Protocol minor
4		Protocol major
5		Protocol name = 'P' (0x50)
6		Options low
7		Options high
8		CRC16 low
9		CRC16 high

7.3.7.3 Framing Packet

The framing packet is used for flow control and error detection, and it (the framing packet) wraps command and data packets as well.

The framing packet described in this section is used for serial peripherals including LPUART, I2C and SPI. The USB HID peripheral does not use framing packets. Instead, the packetization inherent in the USB protocol itself is used. Please refer to the USB peripheral section for details.

Table 7-8. Framing Packet Format

Byte #	Value	Parameter	
0	0x5A	start byte	
1		packetType	
2		length_low	Length is a 16-bit field that specifies the entire command or data packet size in bytes.
3		length_high	
4		crc16_low	This is a 16-bit field. The CRC16 value covers entire framing packet, including the start byte and command or data packets, but does not include the CRC bytes. See the CRC16 algorithm after this table.
5		crc16_high	
6 . . .n		Command or Data packet payload	

A special framing packet that contains only a start byte and a packet type is used for synchronization between the host and target.

Table 7-9. Special Framing Packet Format

Byte #	Value	Parameter
0	0x5A	start byte
1	0xA*n*	packetType

The Packet Type field specifies the type of the packet from one of the defined types (below):

Table 7-10. packetType Field

packetType	Name	Description
0xA1	kFramingPacketType_Ack	The previous packet was received successfully; the sending of more packets is allowed.
0xA2	kFramingPacketType_Nak	The previous packet was corrupted and must be re-sent.
0xA3	kFramingPacketType_AckAbort	Data phase is being aborted.
0xA4	kFramingPacketType_Command	The framing packet contains a command packet payload.
0xA5	kFramingPacketType_Data	The framing packet contains a data packet payload.

Table continues on the next page...

Table 7-10. packetType Field (continued)

packetType	Name	Description
0xA6	kFramingPacketType_Ping	Sent to verify the other side is alive. Also used for UART autobaud.
0xA7	kFramingPacketType_PingResponse	A response to Ping; contains the framing protocol version number and options.

This device uses the Cyclic Redundancy Check module (CRC) to perform the CRC algorithm. See the CRC chapter for more details.

7.3.7.4 Command packet

The command packet carries a 32-bit command header and a list of 32-bit parameters.

Table 7-11. Command Packet Format

Command Packet Format (32 bytes)										
Command Header (4 bytes)				28 bytes for Parameters (Max 7 parameters)						
Tag	Flags	Rsvd	Param Count	Param1 (32-bit)	Param2 (32-bit)	Param3 (32-bit)	Param4 (32-bit)	Param5 (32-bit)	Param6 (32-bit)	Param7 (32-bit)
byte 0	byte 1	byte 2	byte 3							

Table 7-12. Command Header Format

Byte #	Command Header Field	
0	Command or Response tag	The command header is 4 bytes long, with these fields.
1	Flags	
2	Reserved. Should be 0x00.	
3	ParameterCount	

The header is followed by 32-bit parameters up to the value of the ParameterCount field specified in the header. Because a command packet is 32 bytes long, only 7 parameters can fit into the command packet.

Command packets are also used by the target to send responses back to the host. As mentioned earlier, command packets and data packets are embedded into framing packets for all of the transfers.

Table 7-13. Commands that are supported

Command	Name
0x01	FlashEraseAll

Table continues on the next page...

Table 7-13. Commands that are supported (continued)

Command	Name
0x02	FlashEraseRegion
0x03	ReadMemory
0x04	WriteMemory
0x05	FillMemory
0x06	FlashSecurityDisable
0x07	GetProperty
0x08	ReceiveSBFile
0x09	Execute
0x10	FlashReadResource
0x11	ConfigureQuadSpi
0x0A	Call
0x0B	Reset
0x0C	SetProperty
0x0D	FlashEraseAllUnsecure
0x0E	FlashProgramOnce
0x0F	FlashReadOnce

Table 7-14. Responses that are supported

Response	Name
0xA0	GenericResponse
0xA7	GetPropertyResponse (used for sending responses to GetProperty command only)
0xA3	ReadMemoryResponse (used for sending responses to ReadMemory command only)
0xAF	FlashReadOnceResponse (used for sending responses to FlashReadOnce command only)
0xB0	FlashReadResourceResponse (used for sending responses to FlashReadResource command only)

Flags: Each command packet contains a Flag byte. Only bit 0 of the flag byte is used. If bit 0 of the flag byte is set to 1, then data packets will follow in the command sequence. The number of bytes that will be transferred in the data phase is determined by a command-specific parameter in the parameters array.

ParameterCount: The number of parameters included in the command packet.

Parameters: The parameters are word-length (32 bits). With the default maximum packet size of 32 bytes, a command packet can contain up to 7 parameters.

7.3.7.5 Data packet

The data packet carries just the data, either host sending data to target, or target sending data to host. The data transfer direction is determined by the last command sent from the host. The data packet is also wrapped within a framing packet, to ensure the correct packet data is received.

The contents of a data packet are simply the data itself. There are no other fields, so that the most data per packet can be transferred. Framing packets are responsible for ensuring that the correct packet data is received.

7.3.7.6 Response packet

The responses are carried using the same command packet format wrapped with framing packet data. Types of responses include:

- GenericResponse
- GetPropertyResponse
- ReadMemoryResponse
- FlashReadOnceResponse
- FlashReadResourceResponse

GenericResponse: After the Kinetis Bootloader has processed a command, the bootloader will send a generic response with status and command tag information to the host. The generic response is the last packet in the command protocol sequence. The generic response packet contains the framing packet data and the command packet data (with generic response tag = 0xA0) and a list of parameters (defined in the next section). The parameter count field in the header is always set to 2, for status code and command tag parameters.

Table 7-15. GenericResponse Parameters

Byte #	Parameter	Descripton
0 - 3	Status code	The Status codes are errors encountered during the execution of a command by the target (Kinetis Bootloader). If a command succeeds, then a kStatus_Success code is returned. Table 7-61, Kinetis Bootloader Status Error Codes, lists the status codes returned to the host by the Kinetis Bootloader for KL82 ROM.
4 - 7	Command tag	The Command tag parameter identifies the response to the command sent by the host.

GetPropertyResponse: The GetPropertyResponse packet is sent by the target in response to the host query that uses the GetProperty command. The GetPropertyResponse packet contains the framing packet data and the command packet data, with the command/response tag set to a GetPropertyResponse tag value (0xA7).

The parameter count field in the header is set to greater than 1, to always include the status code and one or many property values.

Table 7-16. GetPropertyResponse Parameters

Byte #	Value	Parameter
0 - 3		Status code
4 - 7		Property value
. . .		. . .
		Can be up to maximum 6 property values, limited to the size of the 32-bit command packet and property type.

ReadMemoryResponse: The ReadMemoryResponse packet is sent by the target in response to the host sending a ReadMemory command. The ReadMemoryResponse packet contains the framing packet data and the command packet data, with the command/response tag set to a ReadMemoryResponse tag value (0xA3), the flags field set to kCommandFlag_HasDataPhase (1).

The parameter count set to 2 for the status code and the data byte count parameters shown below.

Table 7-17. ReadMemoryResponse Parameters

Byte #	Parameter	Descripton
0 - 3	Status code	The status of the associated Read Memory command.
4 - 7	Data byte count	The number of bytes sent in the data phase.

FlashReadOnceResponse: The FlashReadOnceResponse packet is sent by the target in response to the host sending a FlashReadOnce command. The FlashReadOnceResponse packet contains the framing packet data and the command packet data, with the command/response tag set to a FlashReadOnceResponse tag value (0xAF), and the flags field set to 0. The parameter count is set to 2 plus *the number of words* requested to be read in the FlashReadOnceCommand.

Table 7-18. FlashReadOnceResponse Parameters

Byte #	Value	Parameter
0 – 3		Status Code

Table continues on the next page...

Table 7-18. FlashReadOnceResponse Parameters (continued)

4 – 7		Byte count to read
…		…
		Can be up to 20 bytes of requested read data.

The FlashReadResourceResponse packet is sent by the target in response to the host sending a FlashReadResource command. The FlashReadResourceResponse packet contains the framing packet data and command packet data, with the command/response tag set to a FlashReadResourceResponse tag value (0xB0), and the flags field set to kCommandFlag_HasDataPhase (1).

Table 7-19. FlashReadResourceResponse Parameters

Byte #	Value	Parameter
0 – 3		Status Code
4 – 7		Data byte count

7.3.8 Bootloader Command API

All Kinetis Bootloader command APIs follow the command packet format that is wrapped by the framing packet, as explained in previous sections.

- For a list of commands supported by the Kinetis Bootloader in KL82 ROM, see Table 7-2, Commands supported.
- For a list of status codes returned by the Kinetis Bootloader in KL82 ROM, see Table 7-61, Kinetis Bootloader Status Error Codes.

NOTE

All the examples in this section depict byte traffic on serial peripherals that use framing packets. USB HID transactions use the USB HID report packets instead of the serial framing packets shown in this section. Please refer to the HID reports section for details of the USB HID packet structure.

7.3.8.1 Call command

The Call command will execute a function that is written in memory at the address sent in the command. The address needs to be a valid memory location residing in accessible flash (internal or external) or in RAM. The command supports the passing of one 32-bit

argument. Although the command supports a stack address, at this time the call will still take place using the current stack pointer. After execution of the function, a 32-bit return value will be returned in the generic response message.

Table 7-20. Parameters for Call Command

Byte #	Command
0 - 3	Call address
4 - 7	Argument word
8 - 11	Stack pointer

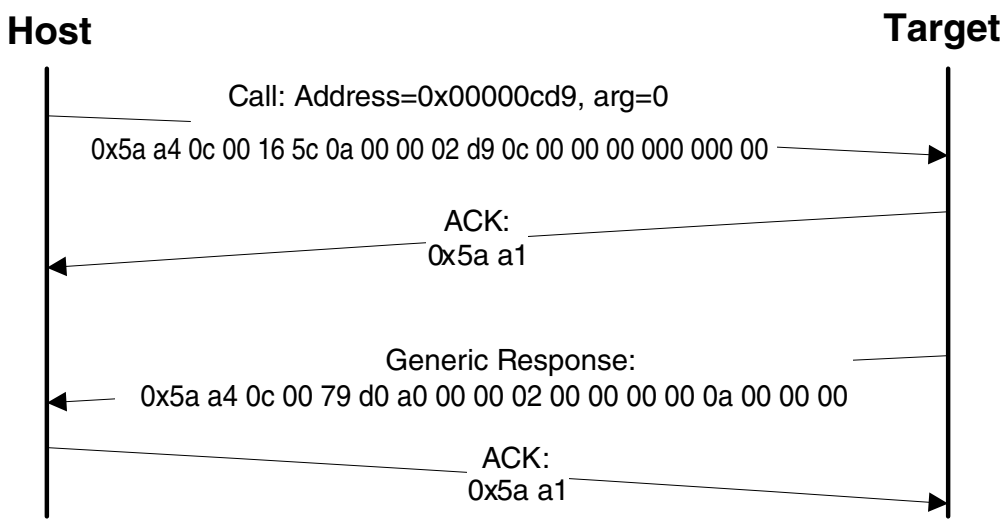


Figure 7-7. Protocol Sequence for Call Command

Response: The target (Kinetis Bootloader) will return a GenericResponse packet with a status code either set to the return value of the function called or set to kStatus_InvalidArgument (105).

7.3.8.2 ConfigureQuadSPI command

The ConfigureQuadSPI command will configure the QuadSPI device using a pre-programmed configuration image. The parameters passed in the command are the QuadSPI memory ID, which should always be 1 for the current release of the bootloader, and then the memory address from which the configuration data can be loaded from.

Options for loading the data can be a scenario where the configuration data is written to a RAM or flash location and then this command will direct the bootloader to use the data at that location to configure the QuadSPI.

Table 7-21. Parameters for ConfigureQuadSPI Command

Byte #	Command
0 – 3	Flash Memory ID (Should always be 1)
4 – 7	Configuration block address

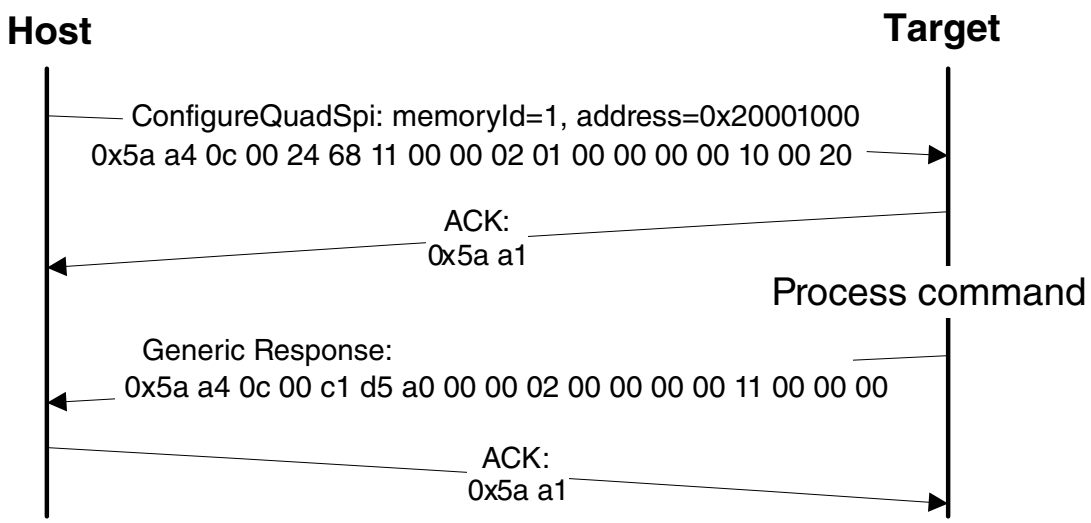


Figure 7-8. Protocol Sequence for Configure QuadSPI Command

Response: The target (Kinetis Bootloader) will return a GenericResponse packet with a status code either set to kStatus_Success upon successful execution of the command, or set to an appropriate error code.

7.3.8.3 ReceiveSBFile command

The ReceiveSBFile command (ReceiveSbFile) will start the transfer of an SB file to the target. The command only specifies the size in bytes of the SB file that will be sent in the data phase. The SB file will be processed as it is received by the bootloader.

Table 7-22. Parameters for ReceiveSBFile Command

Byte #	Command
0 - 3	Byte count

Data Phase: The Receive SB file command has a data phase; the host will send data packets until the number of bytes of data specified in the byteCount parameter of the Receive SB File command are received by the target.

Response: The target (Bootloader) will return a GenericResponse packet with a status code set to the kStatus_Success upon successful execution of the command, or set to an appropriate error code.

7.3.8.4 Execute command

The execute command results in the bootloader setting the program counter to the code at the provided jump address, R0 to the provided argument, and a Stack pointer to the provided stack pointer address. Prior to the jump, the system is returned to the reset state.

The Jump address, function argument pointer, and stack pointer are the parameters required for the Execute command.

Table 7-23. Parameters for Execute Command

Byte #	Command
0 - 3	Jump address
4 - 7	Argument word
8 - 11	Stack pointer address

The Execute command has no data phase.

Response: Before executing the Execute command, the target (Kinetis Bootloader) will validate the parameters and return a GenericResponse packet with a status code either set to kStatus_Success or an appropriate error status code.

7.3.8.5 Reset command

The Reset command will result in bootloader resetting the chip.

The Reset command requires no parameters.

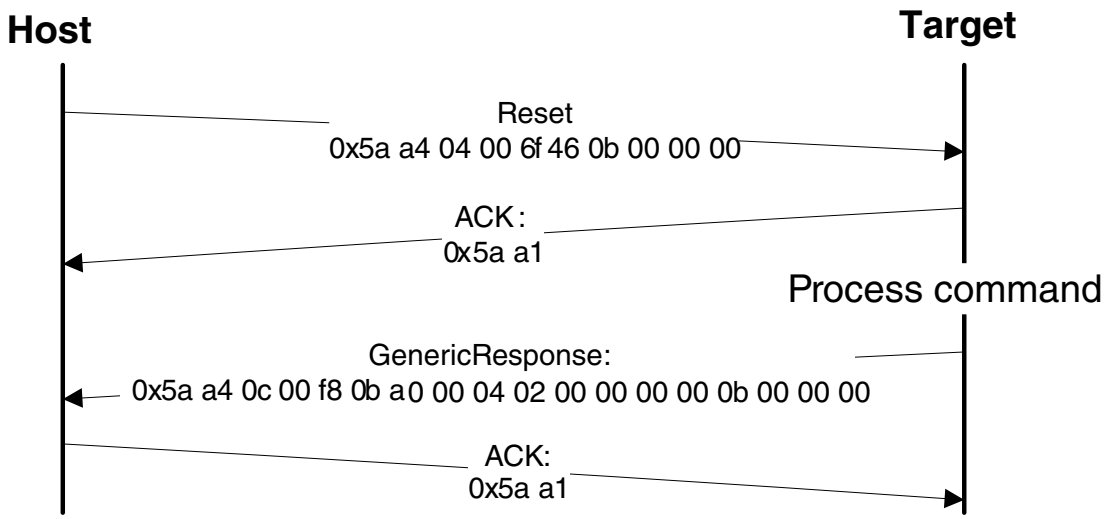


Figure 7-9. Protocol Sequence for Reset Command

Table 7-24. Reset Command Packet Format (Example)

Reset	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x04 0x00
	crc16	0x6F 0x46
Command packet	commandTag	0x0B - reset
	flags	0x00
	reserved	0x00
	parameterCount	0x00

The Reset command has no data phase.

Response: The target (Kinetis Bootloader) will return a GenericResponse packet with status code set to kStatus_Success, before resetting the chip.

7.3.8.6 GetProperty command

The GetProperty command is used to query the bootloader about various properties and settings. Each supported property has a unique 32-bit tag associated with it. The tag occupies the first parameter of the command packet. The target returns a GetPropertyResponse packet with the property values for the property identified with the tag in the GetProperty command.

Properties are the defined units of data that can be accessed with the GetProperty or SetProperty commands. Properties may be read-only or read-write. All read-write properties are 32-bit integers, so they can easily be carried in a command parameter.

For a list of properties and their associated 32-bit property tags supported by the Kinetis Bootloader in KL82 ROM, see Table 7-56.

The 32-bit property tag is the only parameter required for GetProperty command.

Table 7-25. Parameters for GetProperty Command

Byte #	Command
0 - 3	Property tag
4 - 7	External memory identifier, which is only valid when using the GetProperty command for an external memory

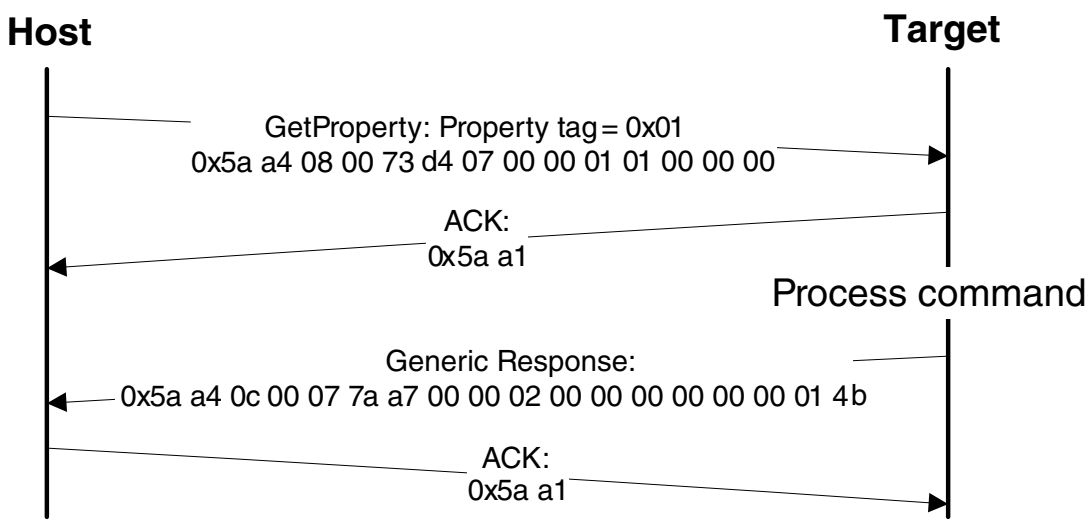


Figure 7-10. Protocol Sequence for GetProperty Command

Table 7-26. GetProperty Command Packet Format (Example)

GetProperty	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x08 0x00
	crc16	0x73 0xD4
Command packet	commandTag	0x07 – GetProperty
	flags	0x00
	reserved	0x00
	parameterCount	0x01
	propertyTag	0x00000001 - CurrentVersion

The GetProperty command has no data phase.

Response: In response to a GetProperty command, the target will send a GetPropertyResponse packet with the response tag set to 0xA7. The parameter count indicates the number of parameters sent for the property values, with the first parameter showing status code 0, followed by the property value(s). The next table shows an example of a GetPropertyResponse packet.

Table 7-27. GetProperty Response Packet Format (Example)

GetPropertyResponse	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x0c 0x00 (12 bytes)
	crc16	0x07 0x7a
Command packet	responseTag	0xA7
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	status	0x00000000
	propertyValue	0x0000014b - CurrentVersion

7.3.8.7 SetProperty command

The SetProperty command is used to change or alter the values of the properties or options in the Kinetis Bootloader ROM. However, the SetProperty command can only change the value of properties that are writable—see Table 7-56, Properties used by Get/SetProperty Commands. If you try to set a value for a read-only property, then the Kinetis Bootloader will return an error.

The property tag and the new value to set are the 2 parameters required for the SetProperty command.

Table 7-28. Parameters for SetProperty Command

Byte #	Command
0 - 3	Property tag
4 - 7	Property value

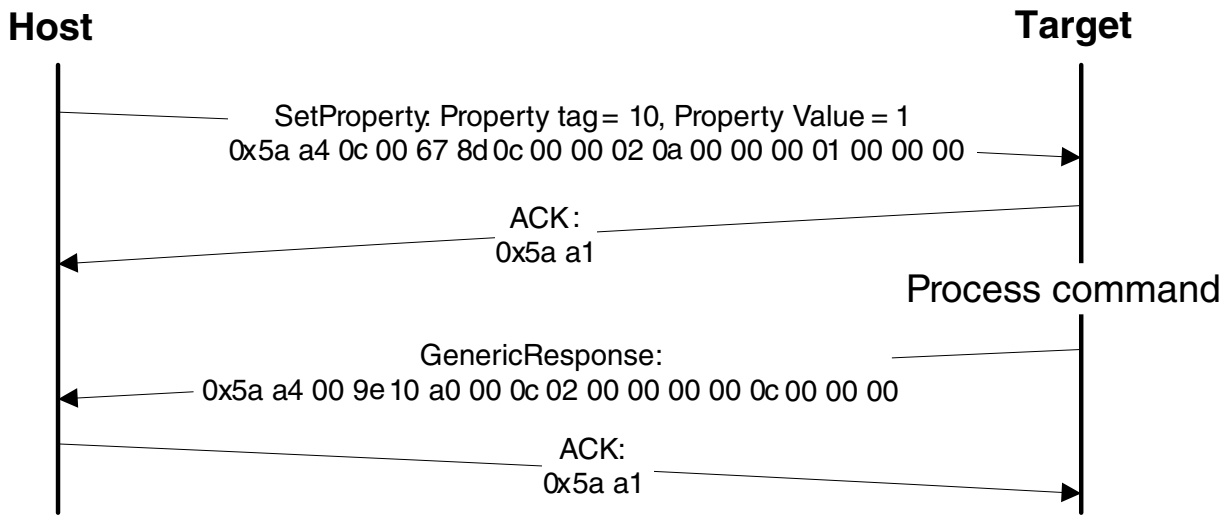


Figure 7-11. Protocol Sequence for SetProperty Command

Table 7-29. SetProperty Command Packet Format (Example)

SetProperty	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x0C 0x00
	crc16	0x67 0x8D
Command packet	commandTag	0x0C – SetProperty with property tag 10
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	propertyTag	0x0000000A - VerifyWrites
	propertyValue	0x00000001

The SetProperty command has no data phase.

Response: The target (Kinetis Bootloader) will return a GenericResponse packet with one of following status codes:

Table 7-30. SetProperty Response Status Codes

Status Code
kStatus_Success
kStatus_ReadOnly
kStatus_UnknownProperty
kStatus_InvalidArgument

7.3.8.8 FlashEraseAll command

The FlashEraseAll command performs an erase of the entire flash memory. If any flash regions are protected, then the FlashEraseAll command will fail and return an error status code. Executing the FlashEraseAll command will release flash security if it (flash security) was enabled, by setting the FTFA_FSEC register. However, the FSEC field of the flash configuration field is erased, so unless it is reprogrammed, the flash security will be re-enabled after the next system reset. The Command tag for FlashEraseAll command is 0x01 set in the commandTag field of the command packet.

The FlashEraseAll command requires no parameters.

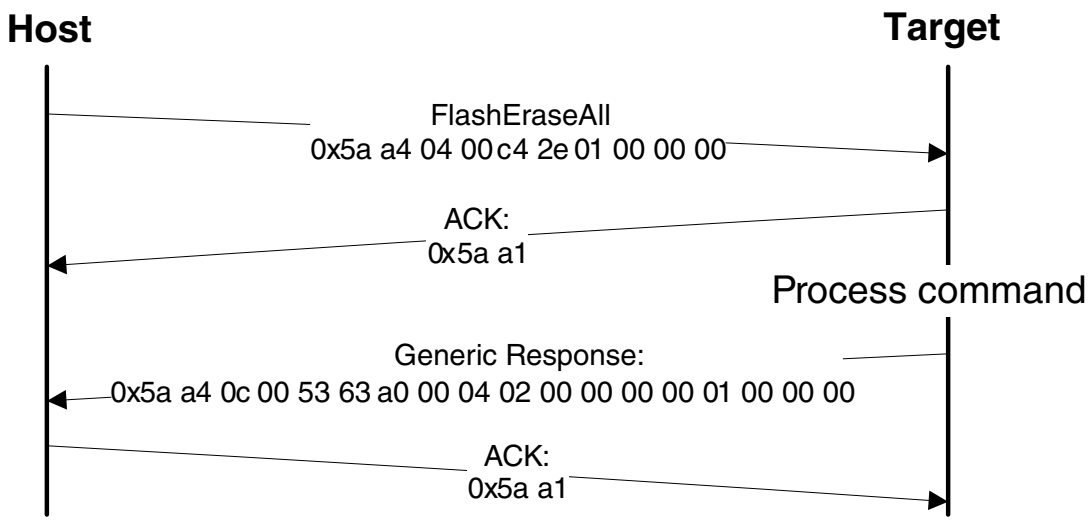


Figure 7-12. Protocol Sequence for FlashEraseAll Command

Table 7-31. FlashEraseAll Command Packet Format (Example)

FlashEraseAll	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x04 0x00
	crc16	0xC4 0x2E
Command packet	commandTag	0x01 - FlashEraseAll
	flags	0x00
	reserved	0x00
	parameterCount	0x00

The FlashEraseAll command has no data phase.

Response: The target (Kinetis Bootloader) will return a GenericResponse packet with status code either set to kStatus_Success for successful execution of the command, or set to an appropriate error status code.

7.3.8.9 FlashEraseRegion command

The FlashEraseRegion command performs an erase of one or more sectors of the flash memory or a specified range of flash within the connected SPI flash devices.

The start address and number of bytes are the 2 parameters required for the FlashEraseRegion command. The start and byte count parameters must be , or the FlashEraseRegion command will fail and return kStatus_FlashAlignmentError (0x101). If the region specified does not fit in the flash memory space, the FlashEraseRegion command will fail and return kStatus_FlashAddressError (0x102). If any part of the region specified is protected, the FlashEraseRegion command will fail and return kStatus_MemoryRangeInvalid (0x10200).

Table 7-32. Parameters for FlashEraseRegion Command

Byte #	Parameter
0 - 3	Start address
4 - 7	Byte count

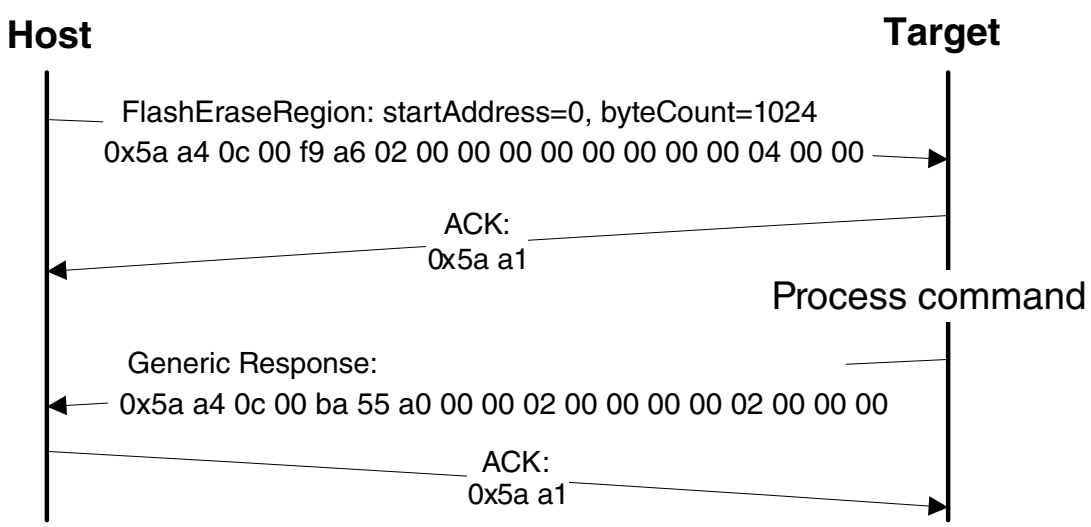


Figure 7-13. Protocol Sequence for FlashEraseRegion Command

Table 7-33. FlashEraseRegion Command Packet Format (Example)

FlashEraseRegion	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x0C 0x00
	crc16	0xF9 0x A6
Command packet	commandTag	0x02, kCommandTag_FlashEraseRegion
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	startAddress	0x00 0x00 0x00 0x00 (0x0000_0000)
	byte count	0x00 0x04 0x00 0x00 (0x400)

The FlashEraseRegion command has no data phase.

Response: The target (Kinetis Bootloader) will return a GenericResponse packet with one of following error status codes.

Table 7-34. FlashEraseRegion Response Status Codes

Status Code
kStatus_Success (0x0)
kStatus_MemoryRangeInvalid (0x10200)
kStatus_FlashAlignmentError (0x101)
kStatus_FlashAddressError (0x102)
kStatus_FlashAccessError (0x103)
kStatus_FlashProtectionViolation (0x104)
kStatus_FlashCommandFailure (0x105)

7.3.8.10 FlashEraseAllUnsecure command

The FlashEraseAllUnsecure command performs a mass erase of the flash memory, including protected sectors. Flash security is immediately disabled if it (flash security) was enabled, and the FSEC byte in the flash configuration field at address 0x40C is programmed to 0xFE. However, if the mass erase enable option in the FSEC field is disabled, then the FlashEraseAllUnsecure command will fail.

The FlashEraseAllUnsecure command requires no parameters.

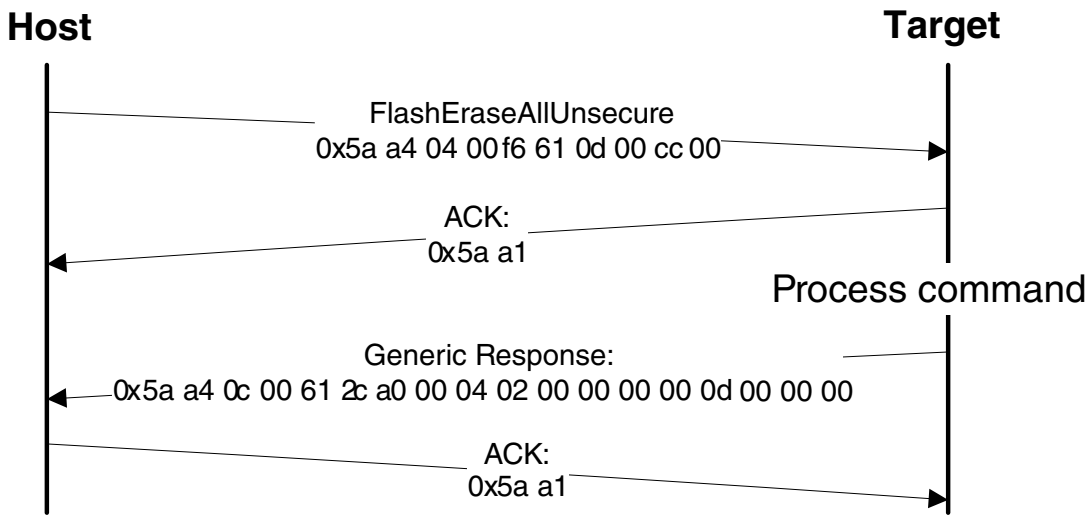


Figure 7-14. Protocol Sequence for FlashEraseAll Command

Table 7-35. FlashEraseAllUnsecure Command Packet Format (Example)

FlashEraseAllUnsecure	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x04 0x00
	crc16	0xF6 0x61
Command packet	commandTag	0x0D - FlashEraseAllUnsecure
	flags	0x00
	reserved	0x00
	parameterCount	0x00

The FlashEraseAllUnsecure command has no data phase.

Response: The target (Kinetis Bootloader) will return a GenericResponse packet with status code either set to kStatus_Success for successful execution of the command, or set to an appropriate error status code.

7.3.8.11 FlashProgramOnce command

The FlashProgramOnce command writes data (that is provided in a command packet) to a specified range of bytes in the program once field. Special care must be taken when writing to the program once field.

- The program once field only supports programming once, so any attempted to reprogram a program once field will get an error response.
- Writing to the program once field requires the byte count to be 4-byte aligned or 8-byte aligned.

The FlashProgramOnce command uses 3 parameters: index, byteCount, data.

Table 7-36. Parameters for FlashProgramOnce Command

Byte #	Command
0 - 3	Index of program once field
4 - 7	Byte count (must be evenly divisible by 4)
8 - 11	Data
12 - 16	Data

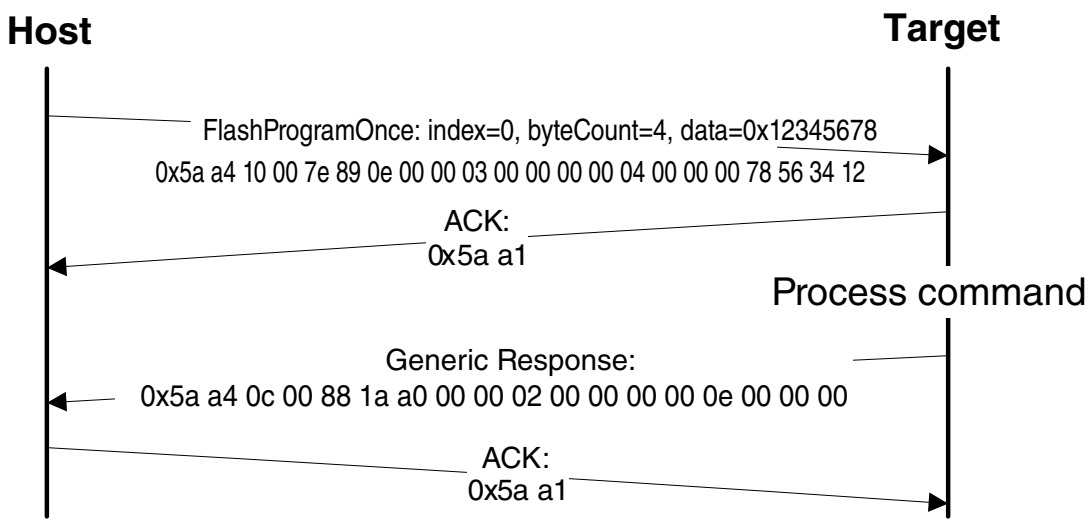


Figure 7-15. Protocol Sequence for FlashProgramOnce Command

Table 7-37. FlashProgramOnce Command Packet Format (Example)

FlashProgramOnce	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x10 0x00
	crc16	0x7E4 0x89
Command packet	commandTag	0x0E – FlashProgramOnce
	flags	0
	reserved	0
	parameterCount	3
	index	0x0000_0000

Table continues on the next page...

Table 7-37. FlashProgramOnce Command Packet Format (Example) (continued)

FlashProgramOnce	Parameter	Value
	byteCount	0x0000_0004
	data	0x1234_5678

Response: upon successful execution of the command, the target (Kinetis Flashloader) will return a GenericResponse packet with a status code set to kStatus_Success, or to an appropriate error status code.

7.3.8.12 FlashReadOnce command

The FlashReadOnce command returns the contents of the program once field by given index and byte count. The FlashReadOnce command uses 2 parameters: index and byteCount.

Table 7-38. Parameters for FlashReadOnce Command

Byte #	Parameter	Description
0 - 3	index	Index of the program once field (to read from)
4 - 7	byteCount	Number of bytes to read and return to the caller

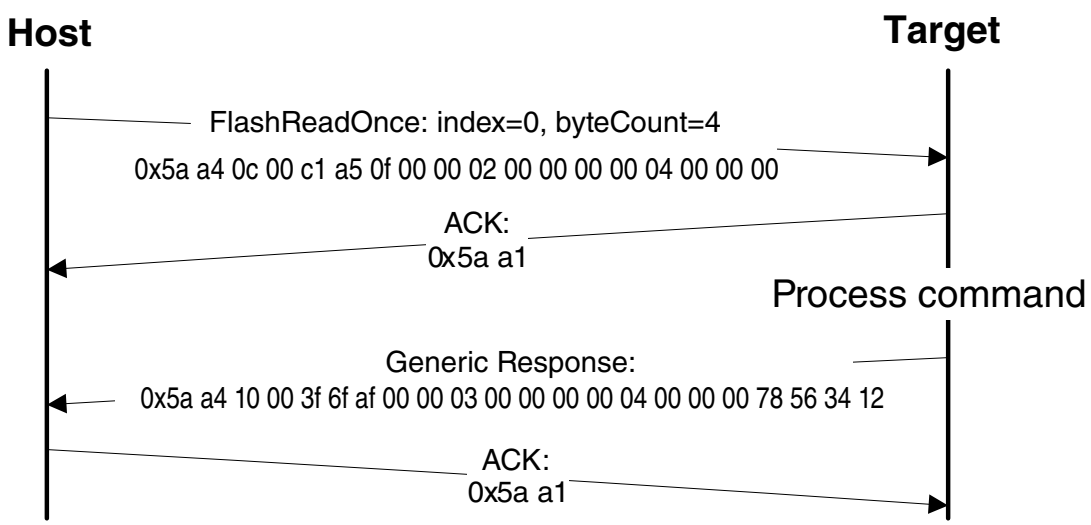


Figure 7-16. Protocol Sequence for FlashReadOnce Command

Table 7-39. FlashReadOnce Command Packet Format (Example)

FlashReadOnce	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4
	length	0x0C 0x00
	crc	0xC1 0xA5
Command packet	commandTag	0x0F – FlashReadOnce
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	index	0x0000_0000
	byteCount	0x0000_0004

Table 7-40. FlashReadOnce Response Format (Example)

FlashReadOnce Response	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4
	length	0x10 0x00
	crc	0x3F 0x6F
Command packet	commandTag	0xAF
	flags	0x00
	reserved	0x00
	parameterCount	0x03
	status	0x0000_0000
	byteCount	0x0000_0004
	data	0x1234_5678

Response: upon successful execution of the command, the target (Kinetis Flashloader) will return a FlashReadOnceResponse packet with a status code set to kStatus_Success, a byte count and corresponding data read from Program Once Field upon successful execution of the command, or will return with a status code set to an appropriate error status code and a byte count set to 0.

7.3.8.13 FlashReadResource command

The FlashReadResource command returns the contents of the IFR field or Flash firmware ID, by given offset, byte count, and option. The FlashReadResource command uses 3 parameters: start address, byteCount, option.

Table 7-41. Parameters for FlashReadResource Command

Byte #	Parameter	Command
0 - 3	start address	Start address of specific non-volatile memory to be read
4 - 7	byteCount	Byte count to be read
8 - 11	option	0: IFR 1: Flash firmware ID

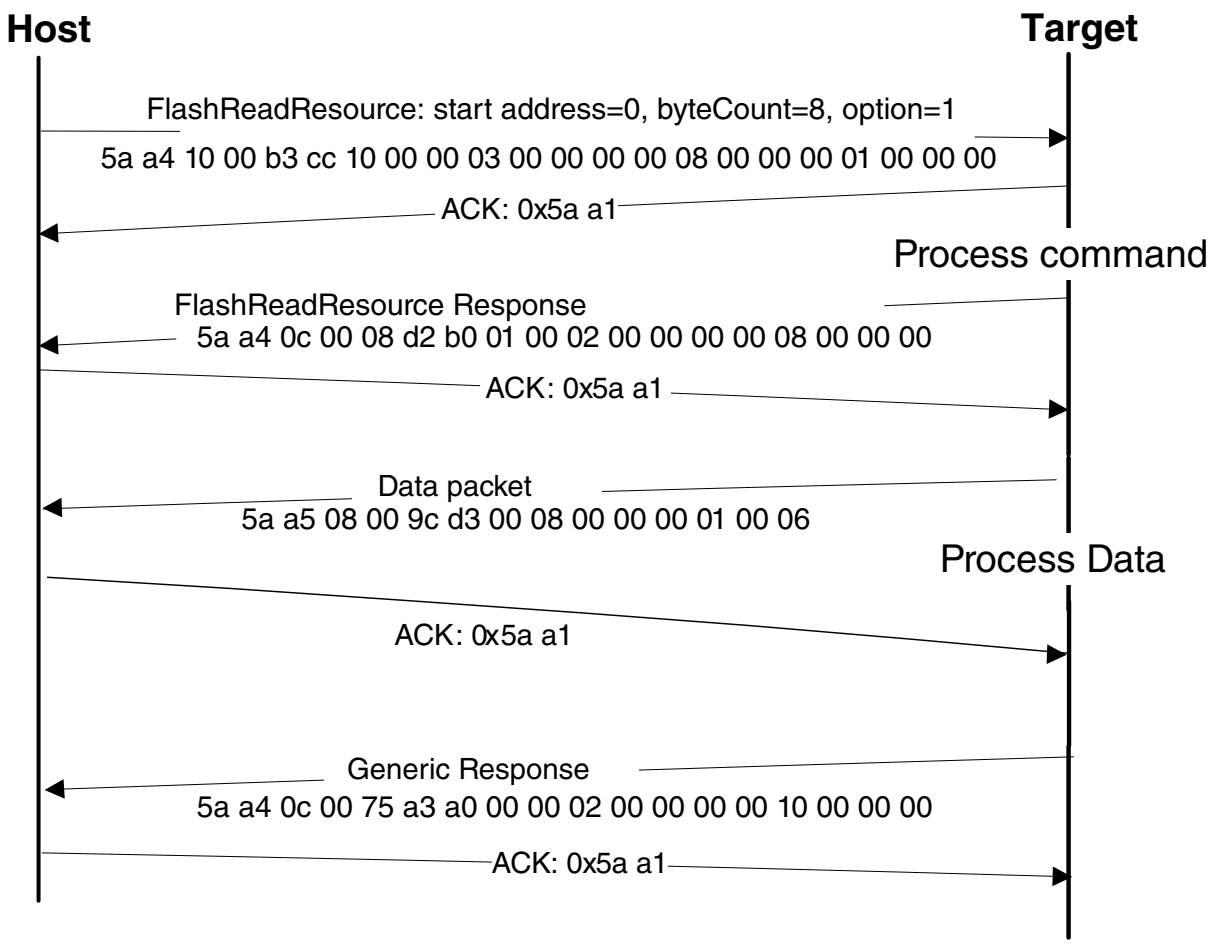


Figure 7-17. Protocol Sequence for FlashReadResource Command

Table 7-42. FlashReadResource Command Packet Format (Example)

FlashReadResource	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4
	length	0x10 0x00
	crc	0xB3 0xCC
Command packet	commandTag	0x10 – FlashReadResource
	flags	0x00

Table continues on the next page...

Table 7-42. FlashReadResource Command Packet Format (Example) (continued)

FlashReadResource	Parameter	Value
	reserved	0x00
	parameterCount	0x03
	startAddress	0x0000_0000
	byteCount	0x0000_0008
	option	0x0000_0001

Table 7-43. FlashReadResource Response Format (Example)

FlashReadResource Response	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4
	length	0x0C 0x00
	crc	0xD2 0xB0
Command packet	commandTag	0xB0
	flags	0x01
	reserved	0x00
	parameterCount	0x02
	status	0x0000_0000
	byteCount	0x0000_0008

Data phase: The FlashReadResource command has a data phase. Because the target (Kinetis Bootloader) works in slave mode, the host must pull data packets until the number of bytes of data *specified in the byteCount parameter of FlashReadResource command* are received by the host.

7.3.8.14 FlashSecurityDisable command

The FlashSecurityDisable command performs the flash security disable operation, by comparing the 8-byte backdoor key (provided in the command) against the backdoor key stored in the flash configuration field (at address 0x400 in the flash).

The backdoor low and high words are the only parameters required for FlashSecurityDisable command.

Table 7-44. Parameters for FlashSecurityDisable Command

Byte #	Command
0 - 3	Backdoor key low word

Table continues on the next page...

Table 7-44. Parameters for FlashSecurityDisable Command (continued)

Byte #	Command
4 - 7	Backdoor key high word

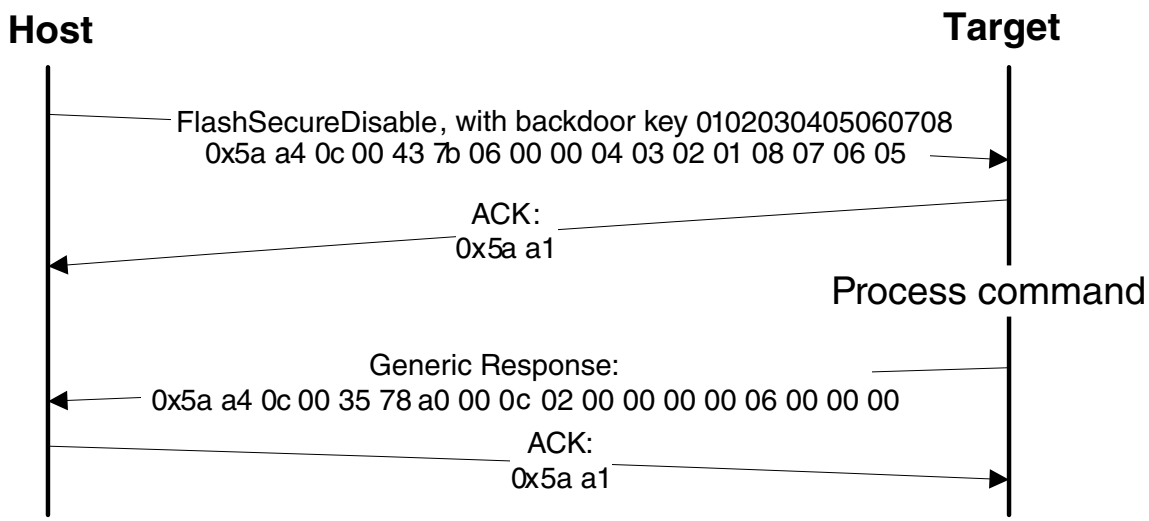


Figure 7-18. Protocol Sequence for FlashSecurityDisable Command

Table 7-45. FlashSecurityDisable Command Packet Format (Example)

FlashSecurityDisable	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x0C 0x00
	crc16	0x43 0x7B
Command packet	commandTag	0x06 - FlashSecurityDisable
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	Backdoorkey_low	0x04 0x03 0x02 0x01
	Backdoorkey_high	0x08 0x07 0x06 0x05

The FlashSecurityDisable command has no data phase.

Response: The target (Kinetis Bootloader) will return a GenericResponse packet with a status code either set to kStatus_Success upon successful execution of the command, or set to an appropriate error status code.

7.3.8.15 FillMemory command

The FillMemory command fills a range of bytes in memory with a data pattern. It follows the same rules as the WriteMemory command. The difference between FillMemory and WriteMemory is that a data pattern is included in FillMemory command parameter, and there is no data phase for the FillMemory command, while WriteMemory does have a data phase.

Table 7-46. Parameters for FillMemory Command

Byte #	Command
0 - 3	Start address of memory to fill
4 - 7	Number of bytes to write with the pattern • The start address should be 32-bit aligned. • The number of bytes must be evenly divisible by 4.
8 - 11	32-bit pattern

- To fill with a byte pattern (8-bit), the byte must be replicated 4 times in the 32-bit pattern.
- To fill with a short pattern (16-bit), the short value must be replicated 2 times in the 32-bit pattern.

For example, to fill a byte value with 0xFE, the word pattern would be 0xFEFEFEFE; to fill a short value 0x5AFE, the word pattern would be 0x5AFE5AFE.

Special care must be taken when writing to flash.

- First, any flash sector written to must have been previously erased with a FlashEraseAll, FlashEraseRegion, or FlashEraseAllUnsecure command.
- Writing to flash requires the start address to be .
- If the VerifyWrites property is set to true, then writes to flash will also perform a flash verify program operation.

When writing to RAM, the start address need not be aligned, and the data will not be padded.

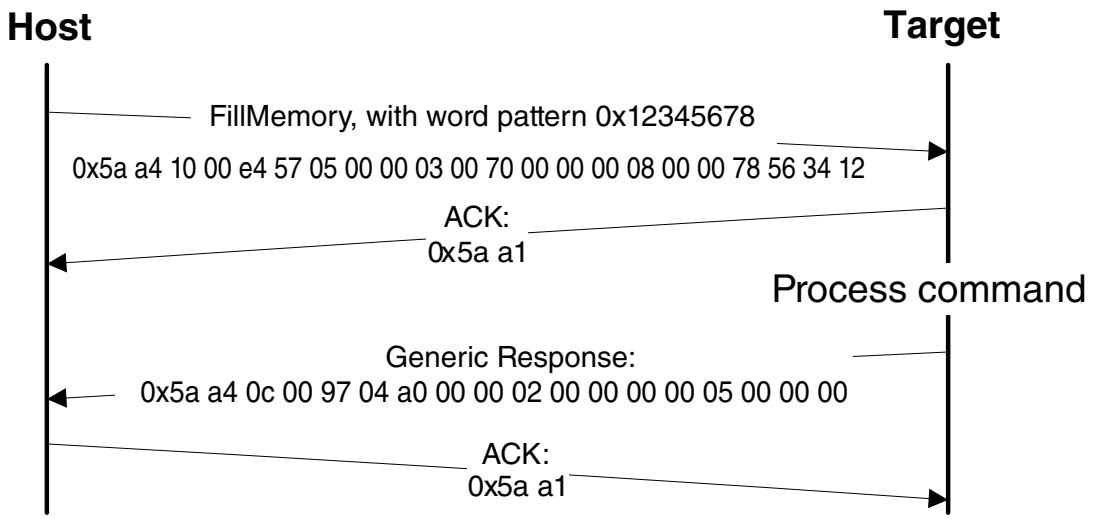


Figure 7-19. Protocol Sequence for FillMemory Command

Table 7-47. FillMemory Command Packet Format (Example)

FillMemory	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x10 0x00
	crc16	0xE4 0x57
Command packet	commandTag	0x05 – FillMemory
	flags	0x00
	Reserved	0x00
	parameterCount	0x03
	startAddress	0x00007000
	byteCount	0x00000800
	patternWord	0x12345678

The FillMemory command has no data phase.

Response: upon successful execution of the command, the target (Kinetis Flashloader) will return a GenericResponse packet with a status code set to kStatus_Success, or to an appropriate error status code.

7.3.8.16 WriteMemory command

The WriteMemory command writes data provided in the data phase to a specified range of bytes in memory (flash or RAM or QuadSPI memory). However, if flash protection is enabled, then writes to protected sectors will fail.

Special care must be taken when writing to flash.

- First, any flash sector written to must have been previously erased with a FlashEraseAll, FlashEraseRegion, or FlashEraseAllUnsecure command.
- Writing to flash requires the start address to be .
- The byte count will be rounded up to a multiple of , and the trailing bytes will be filled with the flash erase pattern (0xff).
- If the VerifyWrites property is set to true, then writes to flash will also perform a flash verify program operation.

When writing to RAM, the start address need not be aligned, and the data will not be padded.

The start address and number of bytes are the 2 parameters required for WriteMemory command.

Table 7-48. Parameters for WriteMemory Command

Byte #	Command
0 - 3	Start address
4 - 7	Byte count

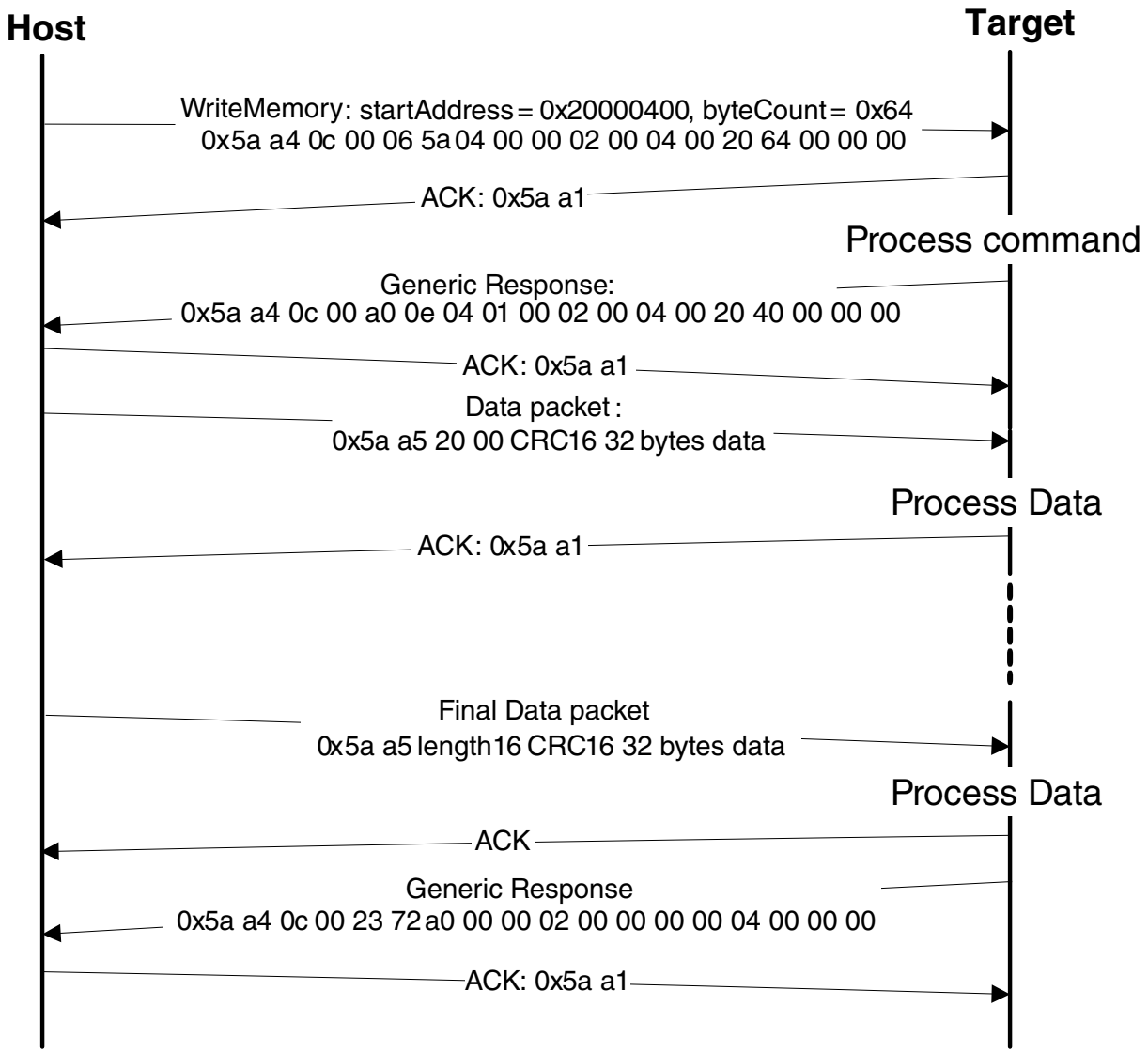


Figure 7-20. Protocol Sequence for WriteMemory Command

Table 7-49. WriteMemory Command Packet Format (Example)

WriteMemory	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x0C 0x00
	crc16	0x06 0x5A
Command packet	commandTag	0x04 - writeMemory
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	startAddress	0x20000400
	byteCount	0x00000064

Data Phase: The WriteMemory command has a data phase; the host will send data packets until the number of bytes of data specified in the byteCount parameter of the WriteMemory command are received by the target.

Response: The target (Kinetis Bootloader) will return a GenericResponse packet with a status code set to kStatus_Success upon successful execution of the command, or to an appropriate error status code.

7.3.8.17 Read memory command

The ReadMemory command returns the contents of memory at the given address, for a specified number of bytes. This command can read any region of memory accessible by the CPU and not protected by security.

The start address and number of bytes are the 2 parameters required for ReadMemory command.

Table 7-50. Parameters for read memory command

Byte	Parameter	Description
0-3	Start address	Start address of memory to read from
4-7	Byte count	Number of bytes to read and return to caller

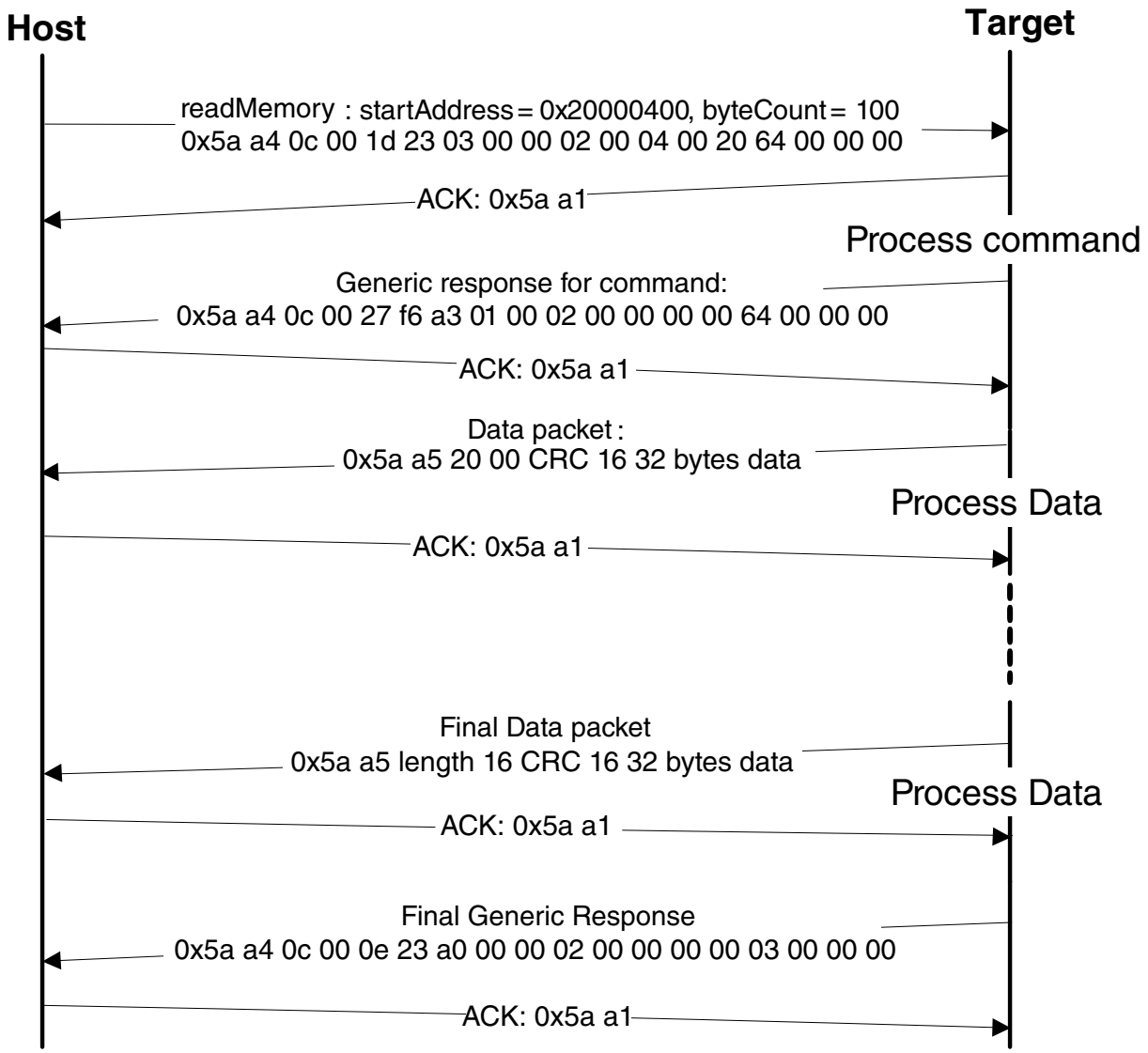


Figure 7-21. Command sequence for read memory

ReadMemory	Parameter	Value
Framing packet	Start byte	0x5A0xA4,
	packetType	kFramingPacketType_Command
	length	0x0C 0x00
	crc16	0x1D 0x23
Command packet	commandTag	0x03 - readMemory
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	startAddress	0x20000400
	byteCount	0x00000064

Data Phase: The ReadMemory command has a data phase. Since the target (Kinetis Bootloader) works in slave mode, the host need pull data packets until the number of bytes of data specified in the byteCount parameter of ReadMemory command are received by host.

Response: The target (Kinetis Bootloader) will return a GenericResponse packet with a status code either set to kStatus_Success upon successful execution of the command, or set to an appropriate error status code.

7.3.9 Bootloader Exit state

The Kinetis Bootloader tries to reconfigure the system back to the reset state in the following situations:

- After completion of an Execute command, but before jumping to the specified entry point.
- After a peripheral detection timeout, but before jumping to the application entry point.

7.4 Peripherals Supported

This section describes the peripherals supported by the Kinetis ROM Bootloader. To use an interface for bootloader communications, the peripheral must be enabled in the BCA, as shown in Table 7-3. If the BCA is invalid (such as all 0xFF bytes), then all peripherals will be enabled by default.

7.4.1 I2C Peripheral

The Kinetis Bootloader in ROM supports loading data into flash via the I2C peripheral, where the I2C peripheral serves as the I2C slave. A 7-bit slave address is used during the transfer.

Customizing an I2C slave address is also supported. This feature is enabled if the Bootloader Configuration Area (BCA) (shown in Table 7-3) is enabled (tag field is filled with ‘kcfg’) and the i2cSlaveAddress field is filled with a value other than 0xFF. If the I2C secondary slave address select pin is asserted, then 0x12 is used as the default I2C slave address; otherwise 0x10 is used as the default I2C slave address.

The maximum supported I2C baud rate depends on corresponding clock configuration field in the BCA. Typical supported baud rate is 400 kbps with factory settings. Actual supported baud rate may be lower or higher than 400 kbps, depending on the actual value of the clockFlags and the clockDivider fields.

Because the I2C peripheral serves as an I2C slave device, each transfer should be started by the host, and each outgoing packet should be fetched by the host.

- An incoming packet is sent by the host with a selected I2C slave address and the direction bit is set as write.
- An outgoing packet is read by the host with a selected I2C slave address and the direction bit is set as read.
- 0x00 will be sent as the response to host if the target is busy with processing or preparing data.

The following flow charts demonstrate the communication flow of how the host reads ping packet, ACK and response from the target.

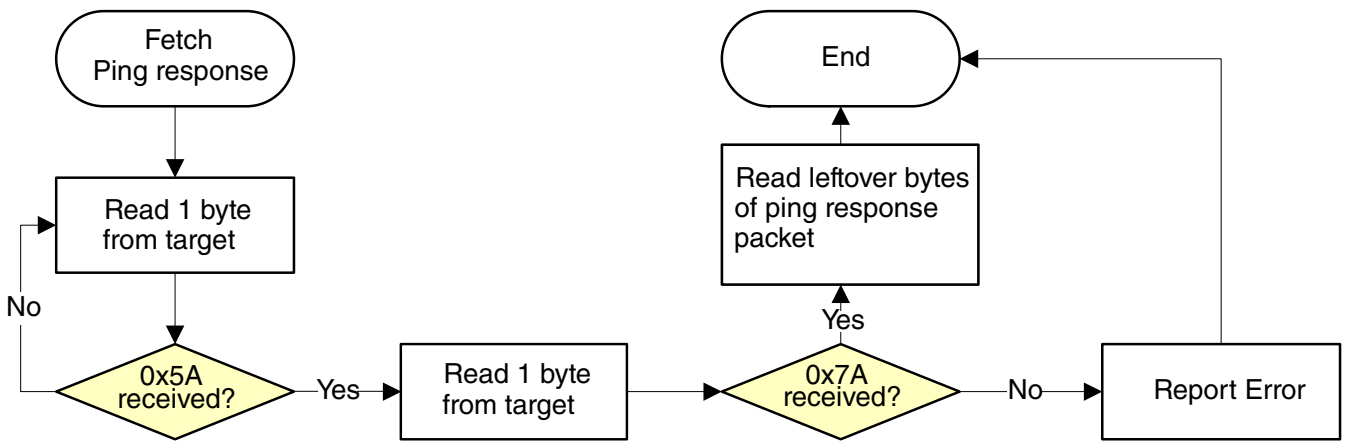


Figure 7-22. Host reads ping response from target via I2C

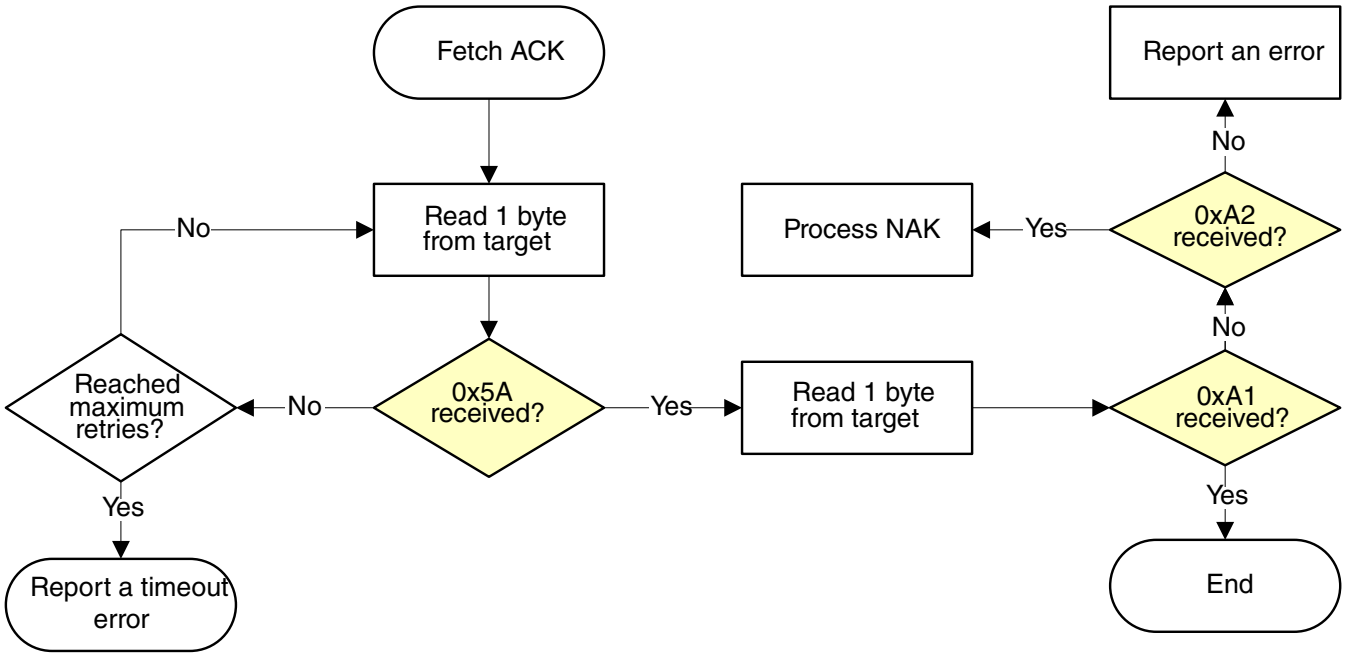


Figure 7-23. Host reads ACK packet from target via I2C

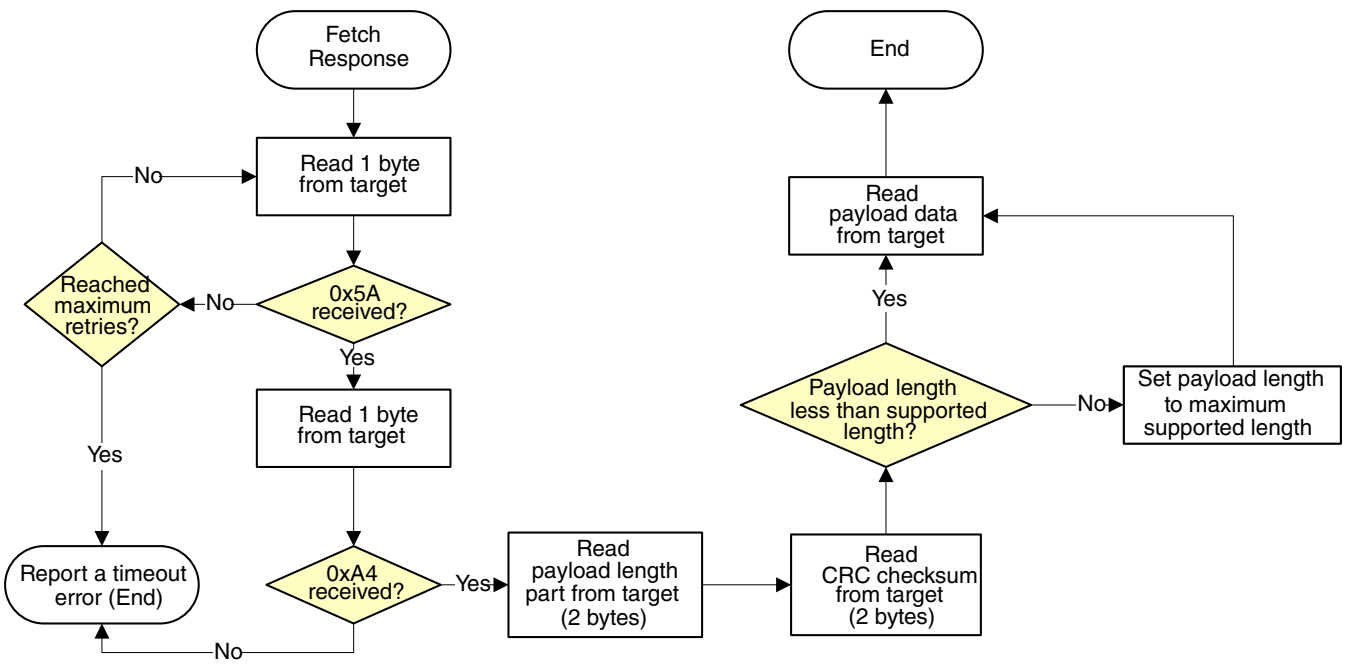


Figure 7-24. Host reads response from target via I2C

7.4.2 SPI Peripheral

The Kinetis Bootloader in KL82 ROM supports loading data into flash via the SPI peripheral, where the SPI peripheral serves as a SPI slave.

Maximum supported baud rate of SPI depends on the clock configuration fields in the Bootloader Configuration Area (BCA) shown in Table 7-3. The typical supported baud rate is 400 kbps with the factory settings. The actual baud rate is lower or higher than 400 kbps, depending on the actual value of the clockFlags and clockDivider fields in the BCA.

The SPI peripheral uses the following bus attributes:

- Clock Phase = 1 (Second Edge)
- Clock Polarity = 1 (Active Low)

Because the SPI peripheral in KL82 ROM serves as a SPI slave device, each transfer should be started by the host, and each outgoing packet should be fetched by the host.

The transfer on SPI is slightly different from I2C:

- Host will receive 1 byte after it sends out any byte.
- Received bytes should be ignored when host is sending out bytes to target
- Host starts reading bytes by sending 0x00s to target
- The byte 0x00 will be sent as response to host if target is under the following conditions:
  - Processing incoming packet
  - Preparing outgoing data
  - Received invalid data

The SPI bus configuration is:

- Phase = 1; data is sampled on rising edges
- Polarity = 1; idle is high
- MSB is transmitted first

For any transfer where the target does not have actual data to send, the target (slave) is responsible for ensuring that 0x00 bytes will be returned to the host (master). The host uses framing packets to identify real data and not "dummy" 0x00 bytes (which do not have framing packets).

The following flowcharts demonstrate how the host reads a ping response, an ACK and a command response from target via SPI.

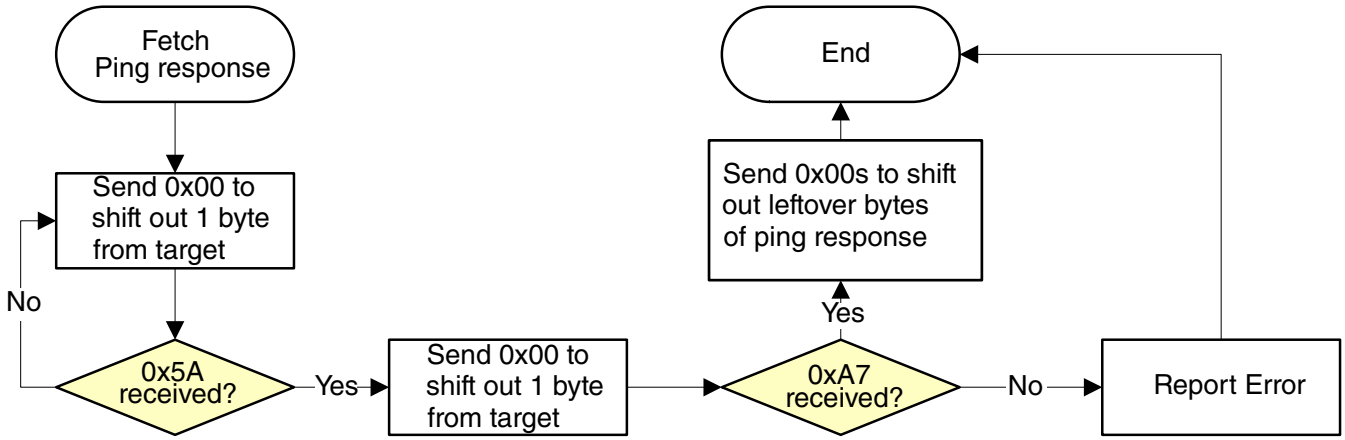


Figure 7-25. Host reads ping packet from target via SPI

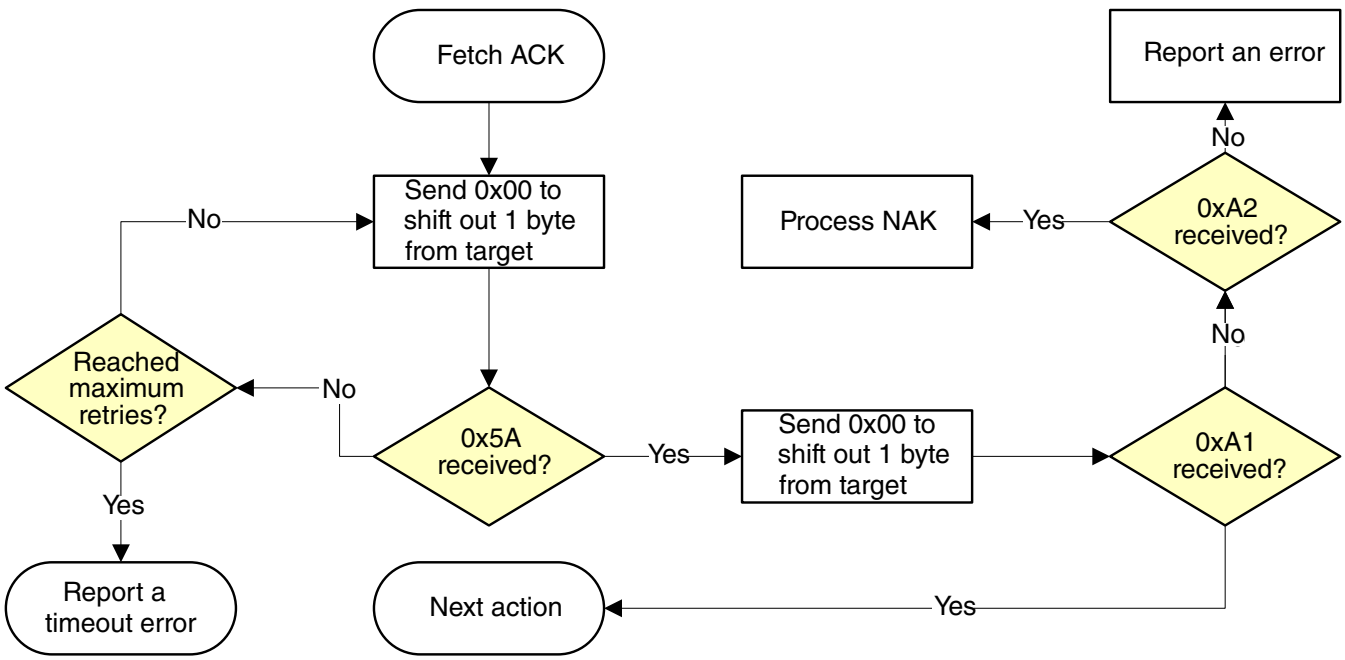


Figure 7-26. Host reads ACK from target via SPI

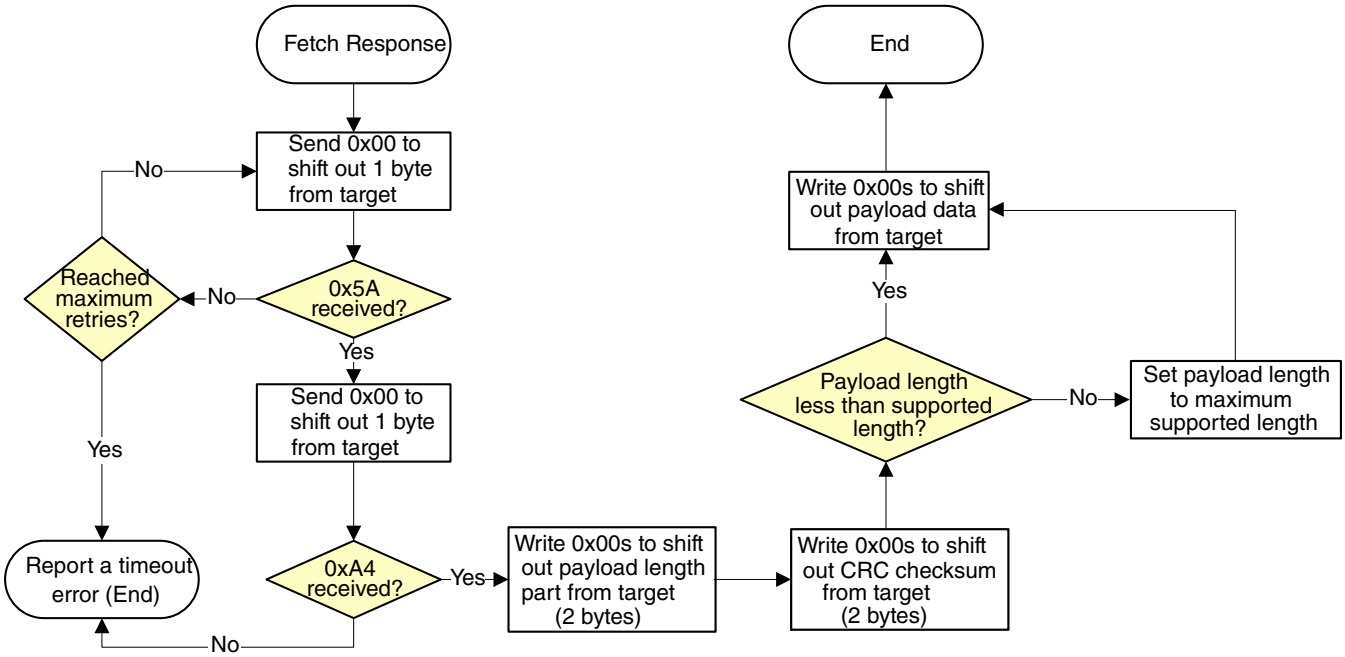


Figure 7-27. Host reads response from target via SPI

7.4.3 QuadSPI Peripheral

The Kinetis Bootloader supports read, write, and erase external SPI flash devices (QuadSPI memory) via the QuadSPI module. It supports booting directly to external SPI flash and XIP in QuadSPI memory. Before accessing external SPI flash devices, the QuadSPI module must be configured properly, using the QSPI configuration block.

7.4.3.1 QSPI configuration block

The QSPI config block (QCB) provides many configuration parameters, which are intended to support many types of serial flash. All fields in the QSPI config block must be configured according to the specific flash device provided by your specific vendor, and all of them are related to the configuration for registers in the QuadSPI module. Also see the QuadSPI chapter.

NOTE

To correctly configure the QuadSPI, **all *unused* QuadSPI configuration fields should be set to 0**.

Table 7-51. Configuration fields in QSPI config block

Offset	Size (bytes)	Configuration Field	Description
0x00 – 0x03	4	tag	A magic number to verify whether the QSPI config block (QCB) is valid. Must be set to 'kqcf' [31:24] - 'f' (0x66) [23:16] - 'c' (0x63) [15: 8] - 'q'(0x71) [7: 0] - 'k'(0x6B)
0x04 – 0x07	4	version	Version number of the QSPI config block [31:24] - name: must be 'Q' (0x51) [23:16] - major: must be 1 [15: 8] - minor: must be 0 [7: 0] - bugfix: must be 0
0x08 – 0x0b	4	lengthInBytes	Size of QSPI config block, in bytes Must be 512
0x0c – 0x0f	4	dqs_loopback	Enable DQS loopback support 0 DQS loopback is disabled 1 DQS loopback is enabled, the DQS loopback mode is determined by subsequent '***dqs_loopback_internal***' field
0x10 – 0x1b	12	-	Reserved
0x1c – 0x1f	4	device_mode_config_en	Configure work mode Enable for external SPI flash devices 0 Disabled - ROM will not configure work mode of external flash devices. 1 Enabled - ROM will configure work mode of external flash devices, based on "device_cmd" and the LUT entry indicated by" write_cmd_ipcr".
0x20 – 0x23	4	device_cmd	Command to configure the work mode of external flash devices. Effective only if "device_mode_config_en" is set to 1. It also depends on your specific external SPI flash device.
0x24 – 0x27	4	write_cmd_ipcr	IPCR pointed to LUT index for quad mode enablement Value = index << 24
0x28 – 0x2b	4	word_addressable	Word Addressable 0 Byte-addressable serial flash mode 1 Word-addressable serial flash mode
0x2c – 0x2f	4	cs_hold_time	Serial flash CS hold time, in number of flash clock cycles
0x30 – 0x33	4	cs_setup_time	Serial flash CS setup time, in number of flash clock cycles
0x34 – 0x37	4	sflash_A1_size	Size of external flash connected to ports of QSPI0A and QSPI0A_CS0, in bytes

Table continues on the next page...

Table 7-51. Configuration fields in QSPI config block (continued)

Offset	Size (bytes)	Configuration Field	Description
0x38 – 0x3b	4	sflash_A2_size	Size of external flash connected to ports of QSPI0A and quadSPI0A_CS1, in bytes sflash_A2_size field must be set to 0 if the serial flash device is not present.
0x3c – 0x3f	4	sflash_B1_size	Size of external flash connected to ports of QSPI0B and quadSPI0B_CS0, in bytes sflash_B1_size field must be set to 0 if the serial flash device is not present.
0x40 – 0x43	4	sflash_B2_size	Size of external flash connected to ports of QSPI0B and quadSPI0B_CS1, in bytes sflash_B2_size field must be set to 0 if the serial flash device is not present.
0x44 – 0x47	4	sclk_freq	Frequency of QuadSPI serial clock 1 0 Low frequency 1 Mid frequency 2 High frequency
0x48 – 0x4b	4	busy_bit_offset	Busy bit offset in status register of Serial flash [31:16] : 0 - Busy flag in status register is 1 when flash devices are busy. 1 - Busy flag in status register is 0 when flash devices are busy. [15:0]: The offset of busy flag in status register; valid range is 0 - 31.
0x4c – 0x4f	4	sflash_type	Type of serial flash 0 Single mode 1 Dual mode 2 Quad mode 3 Octal mode
0x50 – 0x53	4	sflash_port	Port enablement for QuadSPI module 0 Only pins for QSPI0A are enabled 1 Pins for both QSPI0A and QSPI0B are enabled
0x54 – 0x57	4	ddr_mode_enable	Enable DDR mode 0 DDR mode is disabled 1 DDR mode is enabled
0x58 – 0x5b	4	dqs_enable	Enable DQS 0 DQS is disabled 1 DQS is enabled
0x5c – 0x5f	4	parallel_mode_enable	Enable Parallel Mode

Table continues on the next page...

Table 7-51. Configuration fields in QSPI config block (continued)

Offset	Size (bytes)	Configuration Field	Description
			0 Parallel mode is disabled 1 Parallel mode is enabled[1]
0x60 – 0x63	4	portA_cs1	Enable QuadSPI0A_CS1 0 QuadSPI0A_CS1 is disabled 1 QuadSPI0A_CS1 is enabled portA_cs1 field must be set to 1 if sflash_A2_size is not equal to 0.
0x64 – 0x67	4	portB_cs1	Enable QuadSPI0B_CS1 0 QuadSPI0B_CS1 is disabled 1 QuadSPI0B_CS1 is enabled portB_cs1 field must be set to 1 if sflash_B2_size is not equal to 0.
0x68 – 0x6b	4	fsphs	Full Speed Phase selection for SDR instructions 0 Select sampling at non-inverted clock 1 Select sampling at inverted clock
0x6c – 0x6f	4	fsdly	Full Speed Delay selection for SDR instructions 0 One clock cycle delay 1 Two clock cycles delay.
0x70 – 0x73	4	ddrsmp	DDR sampling point Valid range: 0 - 7
0x74 – 0x173	4	look_up_table	Look-up-table for sequences of instructions
0x174 – 0x177	4	column_address_space	Column Address Space Defines the width of the column address
0x178 – 0x17b	4	config_cmd_en	Enable additional configuration command 0 Additional configuration command is not needed 1 Additional configuration command is needed
0x17c – 0x18b	16	config_cmds	IPCR arrays for each connected SPI flash All fields must be set to 0 if config_cmd_en is not asserted.
0x18c - 0x19b	16	config_cmds_args	Command arrays needed to be transferred to external spi flash All fields must be set to 0 if config_cmd_en is not asserted.
0x19c – 0x19f	4	differential_clock_pin_enable	Enable differential flash clock pin 0 Differential flash clock pin is disabled 1 Differential flash clock pin is enabled
0x1a0 – 0x1a3	4	flash_CK2_clock_pin_enable	Enable Flash CK2 Clock pin 0 Flash CK2 Clock pin is disabled

Table continues on the next page...

Table 7-51. Configuration fields in QSPI config block (continued)

Offset	Size (bytes)	Configuration Field	Description
			1 Flash CK2 Clock pin is enabled
0x1a4 – 0x1a7	4	dqs_inverse_sel	Select clock source for internal DQS generation 0 Use 1x internal reference clock for DQS generation 1 Use inverse 1x internal reference clock for DQS generation
0x1a8 – 0x1ab	4	dqs_latency_enable	DQS Latency Enable 0 DQS latency disabled 1 DQS feature with latency included enabled
0x1ac – 0x1af	4	dqs_loopback_internal	DQS loopback from internal DQS signal or DQS Pad 0 DQS loopback is sent to DQS pad first and then looped back to QuadSPI 1 DQS loopback from internal DQS signal directly
0x1b0 – 0x1b3	4	dqs_phase_sel	Select Phase Shift for internal DQS generation 0 No Phase shift 1 Select 45° phase shift 2 Select 90° phase shift 3 Select 135° phase shift
0x1b4 – 0x1b7	4	dqs_fa_delay_chain_sel	Delay chain tap number selection for QuadSPI0A DQS Valid range: 0 - 63
0x1b8 – 0x1bb	4	dqs_fb_delay_chain_sel	Delay chain tap number selection for QuadSPI0B DQS Valid range: 0 - 63
0x1bc – 0x1c3	8	-	Reserved
0x1c4 – 0x1c7	4	page_size	Page size of external SPI flash.[1] Page size of all SPI flash devices must be the same
0x1c8 – 0x1cb	4	sector_size	Sector size of external SPI flash.[1] Sector size of all SPI flash devices must be the same.
0x1cc - 0x1cf	4	timeout_milliseconds	Timeout in terms of milliseconds. 0 Timeout check is disabled. **NOTE:** If the time that the external SPI device is busy is more than this timeout value, then the QuadSPI driver returns a timeout.
0x1d0 – 0x1d3	4	ips_cmd_second_divider	Second divider for IPs command based on QSPI_MCR[SCLKCFG]; the maximum value of QSPI_MCR[SCLKCFG] depends on the specific device.
0x1d4 – 0x1d7	4	need_multi_phase	0 Only 1 phase is necessary to access external flash devices 1 Multiple phases are necessary to erase/program external flash devices

Table continues on the next page...

Table 7-51. Configuration fields in QSPI config block (continued)

Offset	Size (bytes)	Configuration Field	Description
0x1d8 – 0x1db	4	is_spansion_hyperflash	0 External flash devices is not in the Spansion Hyperflash family 1 External flash devices is in the Spansion Hyperflash family
0x1dc – 0x1df	4	pre_read_status_cmd_address_offset[2]	Additional address for the PreReadStatus command. Set this field to 0xFFFF FFFF if it is not required.
0x1e0 – 0x1e3	4	pre_unlock_cmd_address_offset[2]	Additional address for PreWriteEnable command. Set this field to 0xFFFF FFFF if it is not required.
0x1e4 – 0x1e7	4	unlock_cmd_address_offset[2]	Additional address for WriteEnable command. Set this field to 0xFFFF FFFF if it is not required.
0x1e8 – 0x1eb	4	pre_program_cmd_address_offset[2]	Additional address for PrePageProgram command. Set this field to 0xFFFF FFFF if it is not required.
0x1ec – 0x1ef	4	pre_erase_cmd_address_offset[2]	Additional address for PreErase command. Set this field to 0xFFFF FFFF if it is not required.
0x1f0 – 0x1f3	4	erase_all_cmd_address_offset[2]	Additional address for EraseAll command. Set this field to 0xFFFF FFFF if it is not required.
0x1f4 – 0x1ff	12	-	Reserved

1. If parallel mode is enabled, then page size and sector size must be twice the actual size.
2. These fields are effective only if “need_multi_phase” field is set to 1.

7.4.3.2 Look-up-table

The look-up table (LUT) is a part of the QCB, and contains sequences for instructions, such as read and write instructions. The Kinetis Bootloader defines LUT entries to support erase, program and read operations.

NOTE

The sequence in each LUT entry is target-specific. See the datasheet or reference manual of the corresponding serial flash device.

Table 7-52. Look-up table entries for bootloader

Index	Field	Description
0	Read	Sequence for read instructions
1	WriteEnable	Sequence for WriteEnable instructions
2	EraseAll	Sequence for EraseAll instructions
3	ReadStatus	Sequence for ReadStatus instructions
4	PageProgram	Sequence for Page Program instructions
6	PreErase[1]	Sequence for Pre-Erase instructions

Table continues on the next page...

Table 7-52. Look-up table entries for bootloader (continued)

Index	Field	Description
7	SectorErase	Sequence for Sector Erase
8	Dummy	Sequence for dummy operation if needed. For example, if continuous read is configured in index 0, then the dummy LUT should be configured to force the external SPI flash to exit continuous read mode. If a dummy operation is not required, then this LUT entry must be set to 0.
9	PreWriteEnable[1]	Sequence for Pre-WriteEnable instructions
10	PrePageProgram[1]	Sequence for Pre-PageProgram instructions
11	PreReadStatus[1]	Sequence for Pre-ReadStatus instructions

1. If these LUT entries are are not required, then they are allowed to be used for other purposes.

7.4.3.3 Configure QuadSPI module

The Kinetis Bootloader is able to access external SPI devices via the QuadSPI module, but only after the QuadSPI module is configured. There are 2 ways to configure the QuadSPI module:

- Configure QuadSPI module at runtime
- Configure QuadSPI module at start-up

Table 7-53. Configuring the QuadSPI module

Configure QuadSPI at	Procedure	Clock updates during QuadSPI module configuration
runtime	1. Use a WriteMemory command to program the QCB to either a region of RAM or internal flash. 2. Use the ConfigQuadSPI command to configure the QuadSPI module with the QCB that was programmed before. 3. After the above operations, the QuadSPI module has been set to an expected mode specified by the QCB, so the Kinetis bootloader is now able to access all connected SPI flash devices.	**If QuadSPI module is configured at runtime:** The System Core clock will not be updated if the QuadSPI module is configured at runtime; only QUADSPI_MCR [SCLKCFG] is updated according to sclk_freq field within the QCB. In this case, the clock source for QuadSPI module is MCGFLL , (QUADSPI0_SOCCR [QSPISRC] equals 4).
start-up	The steps of configuring QuadSPI at startup is based on the runtime procedure, if the QCB is not present at address 0 of the 1st external SPI flash device. 1. Configure the QuadSPI module at runtime (procedure above). 2. Erase the 1st sector of the 1st connected external SPI flash device using the FlashEraseRegion command. 3. Program the QCB to address 0 of the 1st connected external SPI flash device using the WriteMemory command.	**If QuadSPI module is configured at start-up:** The System Core clock will be updated to 72 MHz, if the QuadSPI module is configured at start-up. In this case, the clock source of the QuadSPI module switches to MCGFLL. The corresponding registers are updated with the values listed in the table *Register value updates when the QuadSPI module is configured at start-up*.

Table 7-53. Configuring the QuadSPI module

Configure QuadSPI at	Procedure	Clock updates during QuadSPI module configuration
	NOTE: For some types of SPI flash devices (like Spansion Hyperflash) which do not support basic reads (0x03) with 24-bit addresses, an alternative is available: for this step, program the QCB to internal flash, set the "qspiConfigBlockPointer" in the BCA to the start address of QCB, and program the BCA to 0x3c0. 4. Update BOOTSRC_SEL field (bits [7:6]) in FOPTregister at the address 0x40D to "0b'10", which means "boot from ROM with QuadSPI configured". 5. Reset the target. 6. After start-up, ROM code reads the QCB from address 0 of the external SPI flash and then configures the QuadSPI according to the QCB. 7. Now, the Kinetis Bootloader is able to access all connected SPI flash devices. The QuadSPI module will be configured automatically out of reset, if the QCB is already present and the BOOTSRC_SEL field (bits [7:6]) in FOPTregister at the address 0x40D equals to "0'b10".	

NOTE

The user application boot from QuadSPI in XIP mode should not change the QuadSPI source clock from what ROM has configured (as shown in the previous table); otherwise a hard fault may occur. However, the QuadSPI source clocks (listed in the next table) can be changed successfully, if the application avoids shutting down the QSPI clock during clock switching; for example, if the clock switch related codes are relocated in either internal flash or SRAM.

Table 7-54. Register value updates when the QuadSPI module is configured at start-up

Register	Field	Value	Description
MCG_C7	OSCSEL	2	IRC48M is chosen as external reference clock
MCG_C2	RANGE	2	
MCG_C1	FRDIV	6	Combined with MCG_C2[RANGE] to divide IRC48M with 1280
MCG_C4	DMX32	0	
MCG_C4	DRST_DRS	2	Combined with MCG_C4[DMX32] to multiply divided clock with 1920, then MCGOUT clock is equal to 72 MHz.
MCG_C6	PLLS	0	Disable PLL
MCG_C1	IREFS	0	Switch to external reference clock

Table continues on the next page...

Table 7-54. Register value updates when the QuadSPI module is configured at start-up (continued)

Register	Field	Value	Description
SIM_CLKDIV1	OUTDIV1	0	
SIM_CLKDIV1	OUTDIV2	1	
SIM_CLKDIV1	OUTDIV3	1	
SIM_CLKDIV1	OUTDIV4	3	
QUADSPII0_SOCCR	QSPISRC	1	MCGFLL clock is selected as the clock source for QuadSPI module

7.4.3.4 Access external SPI flash devices using QuadSPI module

The Kinetis Bootloader supports access to external SPI flash devices using the following commands:

- **Flash-erase-all:** This command can erase all SPI flash devices defined in the QCB. For example, if “flash-erase-all 1”, the 1 represents the source of the erasure command is QuadSPI memory.
- **Flash-erase-region:** This command can erase a specified range of flash within connected SPI flash devices. For example “flash-erase-region 0x68000000 0x10000”.
- **Write-memory:** The Kinetis Bootloader calls the Write-memory command to program specified data to a given region of connected SPI flash devices. For example, “write-memory 0x68001000 led_demo.bin”.
- **Read-memory:** The Kinetis Bootloader calls the Read-memory command to read data from a given region of connected SPI flash devices. For example, “read-memory 0x68000000 1024 temp.bin”.

These commands return error codes.

Table 7-55. Status Error Codes for accessing QuadSPI memory

Error Code	Value	Description
kStatus_Success	0	Operation succeeded without error
kStatus_QspiFlashSizeError	400	Size of external SPI flash is invalid
kStatus_QspiFlashAlignmentError	401	Start Address for program is not page-aligned
kStatus_QspiFlashAddressError	402	The address is invalid
kStatus_QspiFlashCommandFailure	403	The operation failed
kStatus_QspiNotConfigured	405	QSPI module is not successfully configured
kStatus_QspiCommandNotSupported	406	The command is not supported under certain modes
kStatus_QspiCommandTimeout	407	The time that the external SPI device is busy more than the timeout value (timeout_milliseconds).
kStatus_QspiWriteFailure	408	QSPI module cannot perform a program command at the current clock frequency

7.4.3.5 Boot directly from QuadSPI

The Kinetis Bootloader supports booting directly from QuadSPI. To boot directly from QuadSPI, the following conditions must be met:

- The bootFlags field in BCA is set to 0xFE, which means "boot directly from QuadSPI".
- The BOOTSRC_SEL field (bits [7:6]) in the FOPT register at address 0x40D is set to “0’b10”, which means "boot from ROM with QuadSPI configured".
- User application is valid.
- QuadSPI configuration block (QCB) is valid
- CRC check passed if the CRC check feature is enabled.

7.4.4 USB peripheral

The Kinetis Bootloader in the ROM supports loading data into flash via the USB peripheral. The target is implemented as a USB HID class.

USB HID does not use framing packets; instead the packetization inherent in the USB protocol itself is used. The ability for the device to NAK Out transfers (until they can be received) provides the required flow control; the built-in CRC of each USB packet provides the required error detection.

7.4.4.1 Clock configuration

The KL82 ROM supports the crystal-less USB feature. If the USB peripheral is enabled, then the ROM enables the 48-MHz IRC. The ROM also enables the USB clock recovery feature (by setting USB_CLK_RECOVER_CTRL[CLOCK_RECOVER_EN] to 1 and USB_CLK_RECOVER_IRC_EN[IRC_EN] to 1).

7.4.4.2 Device descriptor

The Kinetis ROM configures the default USB VID/PID/Strings as below:

Default VID/PID:

- VID = 0x15A2
- PID = 0x0073

Default Strings:

- Manufacturer [1] = "Freescale Semiconductor Inc."
- Product [2] = "Kinetis Bootloader"

You can customize the USB VID/PID/Strings with the Bootloader Configuration Area (BCA) of the flash. See Table 7-3. For example, the USB VID and PID can be customized by writing the new VID to the usbVid(BCA + 0x14) field and the new PID to the usbPid(BCA + 0x16) field of the BCA in flash. To change the USB strings, you need to prepare a structure (like the one shown below) in the flash, and then write the address of the g_languages structure to the usbStringsPointer(BCA + 0x18) field of the BCA.

```
g_languages = { USB_STR_0,
sizeof(USB_STR_0),
(uint_16)0x0409,
(const uint_8 **)g_string_descriptors,
g_string_desc_size};
the USB_STR_0, g_string_descriptors and g_string_desc_size are defined as below.
USB_STR_0[4] =    {0x02,
0x03,
0x09,
0x04
};
g_string_descriptors[4] =
{ USB_STR_0,
USB_STR_1,
USB_STR_2,
USB_STR_3};
g_string_desc_size[4] =
{ sizeof(USB_STR_0),
sizeof(USB_STR_1),
sizeof(USB_STR_2),
sizeof(USB_STR_3)};
```

You can make your own structure of USB_STR_1, USB_STR_2, USB_STR_3:

- USB_STR_1 is used for the manufacturer string.
- USB_STR_2 is used for the product string.
- USB_STR_3 is used for the serial number string.

By default, the 3 strings are defined as below:

```
USB_STR_1[] =
{  sizeof(USB_STR_1),
USB_STRING_DESCRIPTOR,
'F',0,
'r',0,
'e',0,
'e',0,
's',0,
'c',0,
'a',0,
'l',0,
'e',0,
' ',0,
'S',0,
'e',0,
'm',0,
'i',0,
'c',0,
'o',0,
```

```
'n',0,
'd',0,
'u',0,
'c',0,
't',0,
'o',0,
'r',0,
' ',0,
'I',0,
'n',0,
'c',0,
'.',0
};

USB_STR_2[] =
{  sizeof(USB_STR_2),
USB_STRING_DESCRIPTOR,
'M',0,
'K',0,
' ',0,
'M',0,
'a',0,
's',0,
's',0,
' ',0,
'S',0,
't',0,
'o',0,
'r',0,
'a',0,
'g',0,
'e',0
};

USB_STR_3[] =
{  sizeof(USB_STR_3),
USB_STRING_DESCRIPTOR,
'0',0,
'1',0,
'2',0,
'3',0,
'4',0,
'5',0,
'6',0,
'7',0,
'8',0,
'9',0,
'A',0,
'B',0,
'C',0,
'D',0,
'E',0,
'F',0
};
```

7.4.4.3 Endpoints

The HID peripheral uses 3 endpoints:

- Control (0)
- Interrupt IN (1)
- Interrupt OUT (2)

The Interrupt OUT endpoint is optional for HID class devices, but the Kinetis Bootloader uses it as a pipe, where the firmware can NAK send requests from the USB host.

7.4.4.4 HID reports

There are 4 HID reports defined and used by the bootloader USB HID peripheral. The report ID determines the direction and type of packet sent in the report; otherwise, the contents of all reports are the same.

Report ID	Packet Type	Direction
1	Command	OUT
2	Data	OUT
3	Command	IN
4	Data	IN

For all reports, these properties apply:

Usage Min	1
Usage Max	1
Logical Min	0
Logical Max	255
Report Size	8
Report Count	34

Each report has a maximum size of 34 bytes. This is derived from the minimum bootloader packet size of 32 bytes, plus a 2-byte report header that indicates the length (in bytes) of the packet sent in the report.

NOTE

In the future, the maximum report size may be increased, to support transfers of larger packets. Alternatively, additional reports may be added with larger maximum sizes.

The actual data sent in all of the reports looks like:

0	Report ID
1	Packet Length LSB
2	Packet Length MSB
3	Packet[0]

Table continues on the next page...

4	Packet[1]
5	Packet[2]
	...
N+3-1	Packet[N-1]

This data includes the Report ID, which is required if more than one report is defined in the HID report descriptor. The actual data sent and received has a maximum length of 35 bytes. The Packet Length header is written in little-endian format, and it is set to the size (in bytes) of the packet sent in the report. This size does not include the Report ID or the Packet Length header itself. During a data phase, a packet size of 0 indicates a data phase abort request from the receiver.

7.5 Get/SetProperty Command Properties

This section lists the properties of the GetProperty and SetProperty commands.

Table 7-56. Properties used by Get/SetProperty Commands, sorted by Value

Property	Writable	Tag Value	Size	Descripion
CurrentVersion	No	01h	4	Current bootloader version.
AvailablePeripherals	No	02h	4	The set of peripherals supported on this chip.
FlashStartAddress	No	03h	4	Start address of program flash.
FlashSizeInBytes	No	04h	4	Size in bytes of program flash.
FlashSectorSize	No	05h	4	The size in bytes of one sector of program flash. This is the minimum erase size.
FlashBlockCount	No	06h	4	Number of blocks in the flash array.
AvailableCommands	No	07h	4	The set of commands supported by the bootloader.
VerifyWrites	Yes	0Ah	4	Controls whether the bootloader will verify writes to flash. VerifyWrites feature is enabled by default. 0 - No verification is done. 1 - Enable verification.
MaxPacketSize	No	0Bh	4	Maximum supported packet size for the currently active peripheral interface.
ReservedRegions	No	0Ch	16	List of memory regions reserved by the bootloader. Returned as value pairs (<start-address-of-region>, <end-address-of-region>). • If HasDataPhase flag is not set, then the Response packet parameter count indicates the number of pairs. • If HasDataPhase flag is set, then the second parameter is the number of bytes in the data phase.
RAMStartAddress	No	0Eh	4	Start address of RAM

Table continues on the next page...

Table 7-56. Properties used by Get/SetProperty Commands, sorted by Value (continued)

Property	Writable	Tag Value	Size	Descripion
RAMSizeInBytes	No	0Fh	4	Size in bytes of RAM
SystemDeviceId	No	10h	4	Value of the Kinetis System Device Identification register.
FlashSecurityState	No	11h	4	Indicates whether Flash security is enabled 0 - Flash security is disabled 1 - Flash security is enabled
ExternalMemoryAttributes	No	19h	24	List of attributes supported by the specified memory Id (1=QuadSpi0). See description for the return value in the section ExternalMemoryAttributes Property.

7.5.1 Property Definitions

Get/Set property definitions are provided in this section.

7.5.1.1 CurrentVersion Property

The value of this property is a 4-byte structure containing the current version of the bootloader.

Table 7-57. Fields of CurrentVersion property:

Bits	[31:24]	[23:16]	[15:8]	[7:0]
Field	Name = 'K' (0x4B)	Major version	Minor version	Bugfix version

7.5.1.2 AvailablePeripherals Property

The value of this property is a bitfield that lists the peripherals supported by the bootloader and the hardware on which it is running.

Table 7-58. Peripheral bits:

Bit	[31:7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
Peripheral	Reserved	Reserved	Reserved	USB HID	Reserved	SPI Slave	I2C Slave	LPUART

If the peripheral is available, then the corresponding bit will be set in the property value. All reserved bits must be set to 0.

7.5.1.3 AvailableCommands Property

This property value is a bitfield with set bits indicating the commands enabled in the bootloader. Only commands that can be sent from the host to the target are listed in the bitfield. Response commands such as GenericResponse are excluded.

The bit number that identifies whether a command is present is the command's tag value minus 1. 1 is subtracted from the command tag because the lowest command tag value is 0x01. To get the bit mask for a given command, use this expression:

```
mask = 1 << (tag - 1)
```

Table 7-59. Command bits:

Bit	[31:17]	[16]	[15]	[14]	[13]	[12]	[11]	[10]	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
Command	Reserved	ConfigureQuadSPI	FlashReadResource	FlashReadOnce	FlashProgramOnce	FlashEraseAllUnsecure	SetProperty	Reset	Call	Execute	ReceiveSBFile	GetProperty	FlashSecurityDisable	FillMemory	WriteMemory	ReadMemory	FlashEraseRegion	FlashEraseAll

7.5.1.4 ExternalMemoryAttributes Property

The value returned by this property is a 24-byte data structure containing available external memory attributes: start address, total size in KB, page size, sector size, and block size. Below is the breakdown of 24-byte structure:

Table 7-60. Fields of ExternalMemoryAttributes property:

Field Offset	Field Description
0 - 3	The value returned is a bitmap showing the supported attributes for the external memory, with the corresponding bit-field set. 0x00000001 - start address 0x00000002 - total size 0x00000004 - page size 0x00000008 - sector size 0x00000010 - block size
4 - 7	Start address of external memory

Table continues on the next page...

Table 7-60. Fields of ExternalMemoryAttributes property: (continued)

Field Offset	Field Description
8 - 11	Total size of external memory in terms of kilobytes
12 - 15	Page size of external memory in terms of bytes
16 - 19	Sector size of external memory in terms of bytes
20 - 23	Block size of external memory in terms of bytes

7.6 SB File Decryption Support

The bootloader supports the loading and flashing of applications using an encrypted SB file format. Using the ReceiveSbFile command, the host can send an SB file that contains various commands and data that will be executed on the Kinetis bootloader (the target). An unencrypted SB file can be flashed to the target only when flash security is disabled. When flash security is enabled, all memory writes and reads are disabled, with the exception being an encrypted SB file that can be sent to the target. The SB file is decrypted using an AES-128 style key (16 bytes). Before you send the encrypted file, the key must be written to the target using the FlashProgramOnce command. The SB key begins at offset 0x30 in the program once registers.

For example, for KL81 and KL82 devices, to program a key of 000102030405060708090A0B0C0D0E0F:

- FlashProgramOnce to index 0x30 with 4 bytes of 0x03020100
- FlashProgramOnce to index 0x31 with 4 bytes of 0x07060504
- FlashProgramOnce to index 0x32 with 4 bytes of 0x0B0A0908
- FlashProgramOnce to index 0x33 with 4 bytes of 0x0F0E0D0C

If no key is programmed in these registers, then an attempt is made to decrypt the file using an all-zero key.

7.6.1 Decryption using LTC

SB file decryption is performed by the LTC module; see the LTC chapter for more details.

7.7 CRC-32 Check on Application Data

Using CRC-32 and a given address range, the ROM bootloader supports performing an application integrity check. To properly configure this functionality, the following fields in the bootloader configuration area must be set:

- Set crcStartAddress to the start address that should be used for the CRC check.
- Set crcByteCount to the number of bytes to run the CRC check on, from the start address.
- Set crcExpectedValue to the value that the CRC calculation should result in.

Considerations:

- If all of the above fields are unset (all 0xFF bytes for crcStartAddress, crcByteCount, and crcExpectedValue), then the ROM bootloader returns kStatus_AppCrcCheckInvalid.
- If any one of the above fields are set (crcStartAddress, crcByteCount, and crcExpectedValue), then the ROM bootloader checks if the given address range of the application is valid and if the application just resides in internal flash or external QSPI flash:
  - If false, then the bootloader returns kStatus_AppCrcCheckOutOfRange.
  - If true, then the CRC check occurs. If the CRC check fails, then the bootloader returns kStatus_AppCrcCheckFailed; if the CRC check succeeds, then it returns kStatus_AppCrcCheckPassed.
  - If the bootloader returns kStatus_AppCrcCheckOutOfRange or kStatus_AppCrcCheckFailed, then an external pin (PTC5) will also be asserted, to indicate CRC check failure.
  - Only if kStatus_AppCrcCheckPassed is returned, will the application be jumped to; otherwise the bootloader will stay active, and wait for further commands.
- The CRC32 algorithm in ROM uses a polynomial (0x04C1_1DB7), and the initial seed is 0xFFFF_FFFF.

7.8 Kinetis Bootloader Status Error Codes

This section describes the status error codes that the Kinetis Bootloader returns to the host.

Table 7-61. Kinetis Bootloader Status Error Codes, sorted by Value

Error Code	Value	Description
kStatus_Success	0	Operation succeeded without error.
kStatus_Fail	1	Operation failed with a generic error.
kStatus_ReadOnly	2	Requested value cannot be changed because it is read-only.
kStatus_OutOfRange	3	Requested value is out of range.
kStatus_InvalidArgument	4	The requested command's argument is undefined.
kStatus_Timeout	5	A timeout occurred.
kStatus_FlashSizeError	100	Not used.
kStatus_FlashAlignmentError	101	Address or length does not meet required alignment.
kStatus_FlashAddressError	102	Address or length is outside addressable memory.
kStatus_FlashAccessError	103	The FTFA_FSTAT[ACCERR] bit is set.
kStatus_FlashProtectionViolation	104	The FTFA_FSTAT[FPVIOL] bit is set.
kStatus_FlashCommandFailure	105	The FTFA_FSTAT[MGSTAT0] bit is set.
kStatus_FlashUnknownProperty	106	Unknown Flash property.
kStatus_FlashEraseKeyError	107	The key provided does not match the programmed flash key.
kStatus_FlashRegionExecuteOnly	108	The area of flash is protected as execute only.
kStatus_I2C_SlaveTxUnderrun	200	I2C Slave TX Underrun error.
kStatus_I2C_SlaveRxOverrun	201	I2C Slave RX Overrun error.
kStatus_I2C_AribtrationLost	202	I2C Arbitration Lost error.
kStatus_SPI_SlaveTxUnderrun	300	SPI Slave TX Underrun error.
kStatus_SPI_SlaveRxOverrun	301	SPI Slave RX Overrun error.
kStatus_SPI_Timeout	302	SPI tranfser timed out.
kStatus_SPI_Busy	303	SPI instance is already busy performing a transfer.
kStatus_SPI_NoTransferInProgress	304	Attempt to abort a transfer when no transfer was in progress.
kStatus_QspiFlashSizeError	400	The detected flash size is incorrect.
kStatus_QspiFlashAlignmentError	401	The address that is being written to is not properly aligned.
kStatus_QspiFlashAddressError	402	The QSPI address is invalid.
kStatus_QspiFlashCommandFailure	403	The QSPI command failed.
kStatus_QspiFlashUnknownProperty	404	The property is not supported.
kStatus_QspiNotConfigured	405	The QSPI has not been configured.
kStatus_QspiCommandNotSupported	406	The operation is not supported.
kStatus_QspiCommandTimeout	407	The time that external SPI devices are busy is more than the specified timeout value.
kStatus_QspiWriteFailure	408	The QSPI module cannot perform a program command at the current clock frequency.
kStatus_UnknownCommand	10000	The requested command value is undefined.
kStatus_SecurityViolation	10001	Command is disallowed because flash security is enabled.
kStatus_AbortDataPhase	10002	Abort the data phase early.
kStatus_Ping	10003	Internal: received ping during command phase.
kStatusRomLdrSectionOverrun	10100	The loader has finished processing the SB file.
kStatusRomLdrSignature	10101	The signature of the SB file is incorrect.

Table continues on the next page...

Table 7-61. Kinetis Bootloader Status Error Codes, sorted by Value (continued)

Error Code	Value	Description
kStatusRomLdrSectionLength	10102	The section length in chunks is invalid.
kStatusRomLdrUnencryptedOnly	10103	An encrypted SB file has been sent and decryption support is not available.
kStatusRomLdrEOFReached	10104	The end of the SB file has been reached.
kStatusRomLdrChecksum	10105	The checksum of a command tag block is invalid.
kStatusRomLdrCrc32Error	10106	The CRC-32 of the data for a load command is incorrect.
kStatusRomLdrUnknownCommand	10107	An unknown command was found in the SB file.
kStatusRomLdrIdNotFound	10108	There was no bootable section found in the SB file.
kStatusRomLdrDataUnderrun	10109	The SB state machine is waiting for more data.
kStatusRomLdrJumpReturned	10110	The function that was jumped to by the SB file has returned.
kStatusRomLdrCallFailed	10111	The call command in the SB file failed.
kStatusRomLdrKeyNotFound	10112	A matching key was not found in the SB file's key dictionary to unencrypt the section.
kStatusRomLdrSecureOnly	10113	The SB file sent is unencrypted and security on the target is enabled.
kStatusRomLdrResetReturned	10114	The SB file reset operation has unexpectedly returned.
kStatusMemoryRangeInvalid	10200	Memory range conflicts with a protected region.
kStatus_UnknownProperty	10300	The requested property value is undefined.
kStatus_ReadOnlyProperty	10301	The requested property value cannot be written.
kStatus_InvalidPropertyValue	10302	The specified property value is invalid.
kStatus_AppCrcCheckPassed	10400	CRC check is valid and passed.
kStatus_AppCrcCheckFailed	10401	CRC check is valid but failed.
kStatus_AppCrcCheckInactive	10402	CRC check is inactive.
kStatus_AppCrcCheckInvalid	10403	CRC check is invalid, because the BCA is invalid or the CRC parameters are unset (all 0xFF bytes).
kStatus_AppCrcCheckOutOfRange	10404	CRC check is valid but addresses are out of range.

Chapter 8
Power Management

8.1 Introduction

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

8.2 Clocking modes

Information found here describes the various clocking modes supported on this device.

8.2.1 Partial Stop

Partial Stop is a clocking option that can be taken instead of entering Stop mode and is configured in the SMC Stop Control Register (SMC_STOPCTRL). The Stop mode is only partially entered, which leaves some additional functionality alive at the expense of higher power consumption. Partial Stop can be entered from either Run mode or VLP Run mode.

When configured for PSTOP2, only the core and system clocks are gated and the bus clock remains active. The bus masters and bus slaves clocked by the system clock enter Stop mode, but the bus slaves clocked by bus clock remain in Run (or VLP Run) mode. The clock generators in the MCG and the on-chip regulator in the PMC also remain in Run (or VLP Run) mode. Exit from PSTOP2 can be initiated by a reset, an asynchronous interrupt from a bus master or bus slave clocked by the system clock, or a synchronous interrupt from a bus slave clocked by the bus clock. If configured, a DMA request (using the asynchronous DMA wakeup) can also be used to exit Partial Stop for the duration of a DMA transfer before the device is transitioned back into PSTOP2.

When configured for PSTOP1, both the system clock and bus clock are gated. All bus masters and bus slaves enter Stop mode, but the clock generators in the MCG and the on-chip regulator in the PMC remain in Run (or VLP Run) mode. Exit from PSTOP1 can be initiated by a reset or an asynchronous interrupt from a bus master or bus slave. If configured, an asynchronous DMA request can also be used to exit Partial Stop for the duration of a DMA transfer before the device is transitioned back into PSTOP1.

PSTOP1 is functionally similar to Stop mode, but offers faster wake-up at the expense of higher power consumption. Another benefit is that it keeps all of the MCG clocks enabled, which can be useful for some of the asynchronous peripherals that can remain functional in Stop modes.

8.2.2 DMA Wakeup

The DMA can be configured to wake the device on a DMA request whenever it is placed in Stop mode. The wake-up is configured per DMA channel and is supported in Compute Operation, PSTOP, STOP, and VLPS low power modes.

When a DMA wake-up is detected in PSTOP, STOP or VLPS then the device will initiate a normal exit from the low power mode. This can include restoring the on-chip regulator and internal power switches, enabling the clock generators in the MCG, enabling the system and bus clocks (but not the core clock) and negating the stop mode signal to the bus masters and bus slaves. The only difference is that the CPU will remain in the low power mode with the CPU clock disabled.

During Compute Operation, a DMA wake-up will initiate a normal exit from Compute Operation. This includes enabling the clocks and negating the stop mode signal to the bus masters and bus slaves. The core clock always remains enabled during Compute Operation.

Since the DMA wakeup will enable the clocks and negate the stop mode signals to all bus masters and slaves, software needs to ensure that bus masters and slaves that are not involved with the DMA wake-up and transfer remain in a known state. That can be accomplished by disabling the modules before entry into the low power mode or by setting the Doze enable bit in selected modules.

Once the DMA request that initiated the wake-up negates and the DMA completes the current transfer, the device will transition back to the original low-power mode. This includes requesting all non-CPU bus masters to enter Stop mode and then requesting bus slaves to enter Stop mode. In STOP and VLPS modes the MCG and PMC would then also enter their appropriate modes.

NOTE

If the requested DMA transfer cannot cause the DMA request to negate then the device will remain in a higher power state until the low power mode is fully exited.

An enabled DMA wake-up can cause an aborted entry into the low power mode, if the DMA request asserts during the stop mode entry sequence (or reentry if the request asserts during a DMA wakeup) and can cause the SMC to assert its Stop Abort flag. Once the DMA wake-up completes, entry into the low power mode will restart.

An interrupt that occurs during a DMA wake-up will cause an immediate exit from the low power mode (this is optional for Compute Operation) without impacting the DMA transfer.

A DMA wake-up can be generated by either a synchronous DMA request or an asynchronous DMA request. Not all peripherals can generate an asynchronous DMA request in stop modes, although in general if a peripheral can generate synchronous DMA requests and also supports asynchronous interrupts in stop modes, then it can generate an asynchronous DMA request.

8.2.3 Compute Operation

Compute Operation is an execution or compute-only mode of operation that keeps the CPU enabled with full access to the SRAM and Flash read port, but places all other bus masters and bus slaves into their stop mode. Compute Operation can be enabled in either Run mode, High Speed Run mode or VLP Run mode.

NOTE

Do not enter any stop mode without first exiting Compute Operation.

Because Compute Operation reuses the stop mode logic (including the staged entry with bus masters disabled before bus slaves), any bus master or bus slave that can remain functional in stop mode also remains functional in Compute Operation, including generation of asynchronous interrupts and DMA requests. When enabling Compute Operation in Run mode, module functionality for bus masters and slaves is the equivalent of STOP mode. When enabling Compute Operation in VLP Run mode, module functionality for bus masters and slaves is the equivalent of VLPS mode. The MCG, PMC, SRAM and Flash read port are not affected by Compute Operation, although the Flash register interface is disabled.

During Compute Operation, the AIPS peripheral and external memory () space and QSPI is disabled and attempted accesses generate bus errors. The private peripheral bus (PPB) remains accessible during Compute Operation, including the MCM, System Control Space (SCS) (for NVIC), and SysTick. Although access to the GPIO registers is supported, the GPIO port data input registers do not return valid data since clocks are disabled to the Port Control and Interrupt modules. By writing to the GPIO port data output registers, it is possible to control those GPIO ports that are configured as output pins.

Compute Operation is controlled by the CPO register in the MCM, which is only accessible to the CPU. Setting or clearing the CPOREQ bit in the MCM initiates entry or exit into Compute Operation. Compute Operation can also be configured to exit automatically on detection of an interrupt, which is required in order to service most interrupts. Only the core system interrupts (exceptions, including NMI and SysTick) and any edge sensitive interrupts can be serviced without exiting Compute Operation.

When entering Compute Operation, the CPOACK status bit indicates when entry has completed. When exiting Compute Operation in Run mode, the CPOACK status bit negates immediately. When exiting Compute Operation in VLP Run mode, the exit is delayed to allow the PMC to handle the change in power consumption. This delay means the CPOACK bit is polled to determine when the AIPS peripheral space can be accessed without generating a bus error.

The DMA wakeup is also supported during Compute Operation and causes the CPOACK status bit to clear and the AIPS peripheral space to be accessible for the duration of the DMA wakeup. At the completion of the DMA wakeup, the device transitions back into Compute Operation.

8.2.4 Peripheral Doze

Several peripherals support a Peripheral Doze mode, where a register bit can be used to disable the peripheral for the duration of a low-power mode. The flash memory can also be placed in a low-power state during Peripheral Doze via a register bit in the SIM.

Peripheral Doze is defined to include all of the modes of operation listed below.

- The CPU is in Wait mode.
- The CPU is in Stop mode, including the entry sequence and for the duration of a DMA wakeup.
- The CPU is in Compute Operation, including the entry sequence and for the duration of a DMA wakeup.

Peripheral Doze can therefore be used to disable selected bus masters or slaves for the duration of WAIT or VLPW mode. It can also be used to disable selected bus slaves immediately on entry into any stop mode (or Compute Operation), instead of waiting for the bus masters to acknowledge the entry as part of the stop entry sequence. Finally, it can be used to disable selected bus masters or slaves that should remain inactive during a DMA wakeup.

If the flash memory is not being accessed during WAIT and PSTOP modes, then the Flash Doze mode can be used to reduce power consumption, at the expense of a slightly longer wake-up when executing code and vectors from flash. It can also be used to reduce power consumption during Compute Operation when executing code and vectors from SRAM.

8.2.5 Clock Gating

To conserve power, the clocks to most modules can be turned off using the SIM module registers.

8.3 Power modes description

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 VLRP modes, 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.

Stop mode entry is not supported directly from HSRun and requires transition to Run prior to an attempt to enter a Stop mode.

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 8-1. Chip power modes

Chip mode	Description	Core mode	Normal recovery method
Run (Normal Run)	Allows maximum performance of chip. • Default mode out of reset • On-chip voltage regulator is on.	Run	—
Wait (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
Stop (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. Only MCG_Lite modes BLPILIRC and BLPEEXT can be used in VLPR. • Reduced frequency Flash access mode (1 MHz) • LVD off • In BLPILIRC clock mode, only the fast internal reference oscillator (IRC8M) is available to provide a low power nominal 4 MHz source for the core with the nominal bus and flash clock required to be <800 kHz • Alternatively, BLPEEXT clock mode can be used with an external clock or the crystal oscillator providing the clock source.	Run	—
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 OSC, LPTMR, LPUART, RTC, CMP, TSI can be used. • TPM and LPUART can optionally be enabled if their clock source is enabled. • 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
LLS3 (Low-Leakage Stop)	State retention power mode • Most peripherals are in state retention mode (with clocks stopped), but OSC, LLWU, LPTMR, RTC, CMP, TSI can be used. • NVIC is disabled; LLWU is used to wake up.	Sleep Deep	Wake-up Interrupt[1]

Table continues on the next page...

Table 8-1. Chip power modes (continued)

Chip mode	Description	Core mode	Normal recovery method
	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 • All SRAM is operating (content retained and I/O states held).		
LLS2 (Low-Leakage Stop)	State retention power mode • Most peripherals are in state retention mode (with clocks stopped), but OSC, LLWU, LPTMR, RTC, CMP, TSI can be used. • NVIC is disabled; LLWU is used to wake up. **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 • The system RAM3 partition is powered down. The system RAM1 partition, internal logic and I/O states are retained.	Sleep Deep	Wake-up Interrupt[1]
VLLS3 (Very Low-Leakage Stop3)	• Most peripherals are disabled (with clocks stopped), but OSC, LLWU, LPTMR, RTC, CMP, TSI can be used. • NVIC is disabled; LLWU is used to wake up. • SRAM_U and SRAM_L remain powered on (content retained and I/O states held).	Sleep Deep	Wake-up Reset[-1]
VLLS2 (Very Low-Leakage Stop3)	• Most peripherals are disabled (with clocks stopped), but OSC, LLWU, LPTMR, RTC, CMP, TSI can be used. • NVIC is disabled; LLWU is used to wake up. • 32K of SRAM_U remains powered on (content retained and I/O states held).	Sleep Deep	Wake-up Reset[-1]
VLLS1 (Very Low-Leakage Stop1)	• Most peripherals are disabled (with clocks stopped), but OSC, LLWU, LPTMR, RTC, CMP, TSI can be used. • NVIC is disabled; LLWU is used to wake up. • All of SRAM_U and SRAM_L are powered off. • The 32-byte system register file remains powered for customer-critical data • The 16-byte system register file remains powered for customer-critical data	Sleep Deep	Wake-up Reset[-1]
VLLS0 (Very Low-Leakage Stop 0)	• Most peripherals are disabled (with clocks stopped), but LLWU, LPTMR, RTC, TSI can be used. • NVIC is disabled; LLWU is used to wake up. • All of SRAM_U and SRAM_L are powered off. • The 32-byte system register file remains powered for customer-critical data • The 16-byte system register file remains powered for customer-critical data • LPO disabled, optional POR brown-out detection	Sleep Deep	Wake-up Reset[-1]

1. Resumes Normal Run mode operation by executing the LLWU interrupt service routine.

8.4 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.

For LLS3, LLS2, VLLS3, VLLS2 and VLLS1 modes, the wake-up sources are limited to LLWU generated wake-ups, LPTMR, CMP, RTC, NMI_b pin, or RESET_b pin assertions. When the NMI_b pin or RESET_b pin have been disabled through associated FTFA_FOPT settings, then these pins are ignored as wakeup sources. The wake-up flow from VLLSx is always through reset.

Before executing the WFI instruction, the last register written to must be read back. This ensures that all register writes associated with setting up the low power mode being entered have completed before the MCU enters the low power mode. Failure to do this may result in the low power mode not being entered correctly.

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.

On VLLS recoveries, the I/O pins continue to be held in a static state after code execution begins, allowing software to reconfigure the system before unlocking the I/O. RAM is retained in VLLS3 only.

8.5 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 mode(s) are the lowest power stop modes based on amount of logic or memory that is required to be retained by the application.

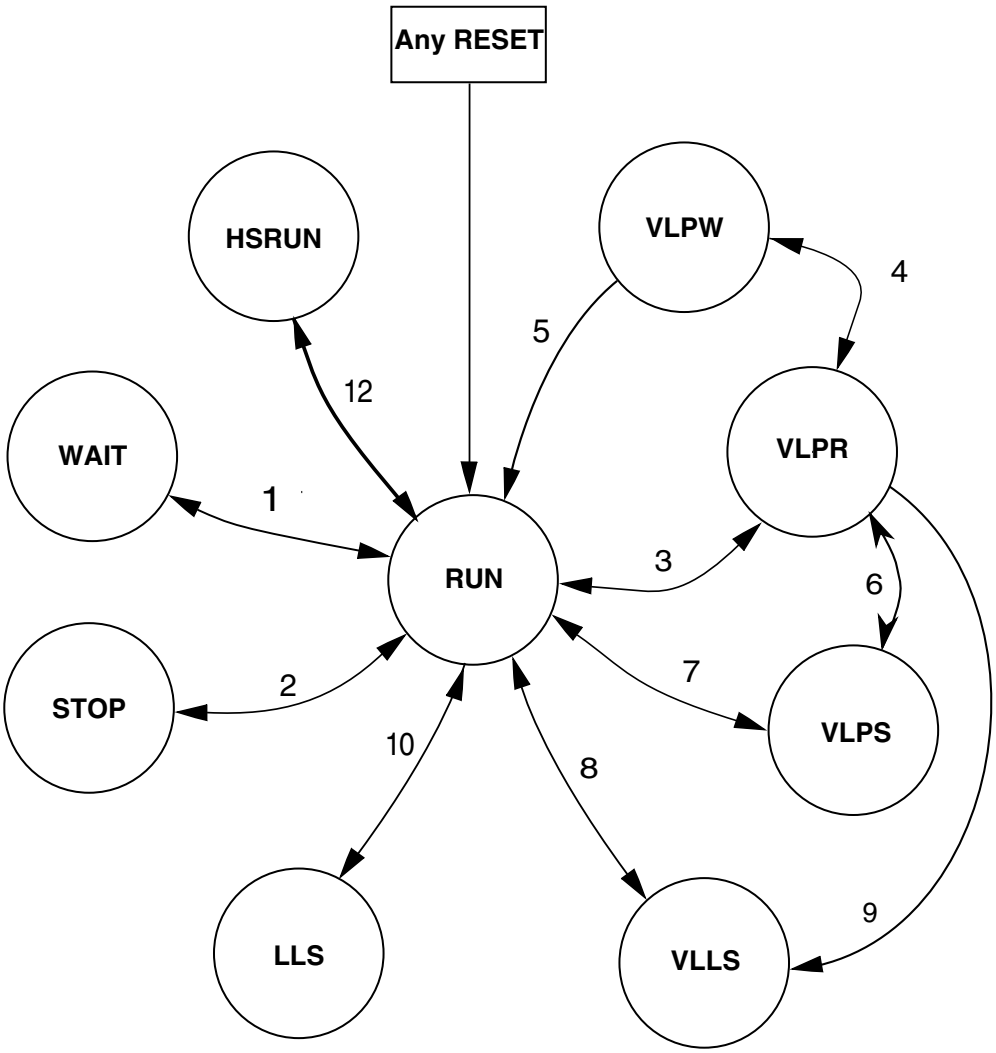


Figure 8-1. Power mode state transition diagram

8.6 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-M0+ core immediately.
- Polls stop acknowledge indications from the non-core crossbar masters (DMA), supporting peripherals (SPI, PIT, TRNG) 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.
- SCG 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-M0+ 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.

8.7 Flash Program Restrictions

The flash memory on this device should not be programmed or erased while operating VLPR power modes.

8.8 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. The standard behavior is shown with some exceptions for Compute Operation (CPO) and Partial Stop2 (PSTOP2).

(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.
- Async operation = Fully functional with alternate clock source, provided the selected clock source remains enabled
- static = Module register states and associated memories are retained.
- powered = Memory is powered to retain contents.
- low power = Memory is powered to retain contents in a lower power state

- OFF = Modules are powered off; module is in reset state upon wakeup. For clocks, OFF means disabled.
- wakeup = Modules can serve as a wakeup source for the chip.

Table 8-2. Module operation in low power modes

Modules	VLPR	VLPW	Stop	VLPS	LLSx	VLLSx
Core modules						
NVIC	FF	FF	static	static	static	OFF
System modules						
Mode Controller	FF	FF	FF	FF	FF	FF
LLWU[1]	static	static	static	static	FF	FF[2]
Regulator	low power	low power	ON	low power	low power	low power in VLLS2/3, OFF in VLLS0/1
LVD	disabled	disabled	ON	disabled	disabled	disabled
Brown-out Detection	ON	ON	ON	ON	ON	ON in VLLS1/2/3, optionally disabled in VLLS0[3]
DMA	FF Async operation in CPO	FF	Async operation	Async operation	static	OFF
Watchdog	FF	FF	FF	FF	static	OFF
EWM	FF static in CPO	static	static FF in PSTOP2	static	static	OFF
Clocks						
1kHz LPO	ON	ON	ON	ON	ON	ON in VLLS1/3,optional be disabled bySMC_STOPCTRL [LPOPO] ,OFF in VLLS0
System oscillator (OSC)	OSCERCLK max of 16MHz crystal; low range/low power (30~40 kHz)	OSCERCLK max of 16MHz crystal; low range/low power(30~40 kHz)	OSCERCLK optional	OSCERCLK max of 16MHz crystal; low range/low power (30~40 kHz)	OSCERCLK max of 16MHz crystal; low range/low power (30~40 kHz)	OSCERCLK max of 16MHz crystal inVLLS1/3, OFF inVLLS0; low range/low powerin VLLS1/3, OFF in VLLS0
MCG	4 MHz IRC	4 MHz IRC	static - MCGIRCLK optional ; PLL optionally on but gated	static - MCGIRCLK optional (4 MHz IRC only).	static - no clock output	OFF
Core clock	4 MHz max	OFF	OFF	OFF	OFF	OFF

Table continues on the next page...

Table 8-2. Module operation in low power modes (continued)

Modules	VLPR	VLPW	Stop	VLPS	LLSx	VLLSx
Platform clock	4 MHz max	4 MHz max	OFF	OFF	OFF	OFF
System clock	4 MHz max OFF in CPO	4 MHz max	OFF	OFF	OFF	OFF
Bus clock	1 MHz max OFF in CPO	1 MHz max	OFF 24 MHz max in PSTOP2 from RUN 1 MHz max in PSTOP2 from VLPR	OFF	OFF	OFF
Memory and memory interfaces						
Flash	1 MHz max access - no program/erase No register access in CPO	low power	low power	low power	OFF	OFF
System RAM (SRAM_U and SRAM_L)	low power	low power	low power	low power	low power in LLS3, partial in LLS2	low power in VLLS3, partial in VLLS2; otherwise OFF
VBAT Register file[4]	powered	powered	powered	powered	powered	powered
System Register files	powered	powered	powered	powered	powered	powered
QSPI	FF, static in CPO	FF	static [5]	static	static	OFF
Communication interfaces						
USB FS/LS	static, wakeup on resume	static, wakeup on resume	static, wakeup on resume	static, wakeup on resume	static	OFF
LPUART	4 Mbps Async operation in CPO	4 Mbps	Async operation FF in PSTOP2	Async operation	static	OFF
SPI	1 Mbit/s (slave) 2 Mbit/s (master) static in CPO	1 Mbit/s (slave) 2 Mbit/s (master)	static	static	static	OFF
I^2C	200 kbit/s static, address match wakeup in CPO	200 kbit/s	static, address match wakeup FF in PSTOP2	static, address match wakeup	static	OFF
EMVSIM	FF	FF	static	static	static	OFF
FLEXIO	FF	FF	Async operation FF in PSTOP2	Async operation	static	OFF
Security						

Table continues on the next page...

Table 8-2. Module operation in low power modes (continued)

Modules	VLPR	VLPW	Stop	VLPS	LLSx	VLLSx
CRC	FF static in CPO	FF	static	static	static	OFF
TRNG	FF static in CPO	static	static	static	static	OFF
LTC	FF, static in CPO	FF	static	static	static	OFF
Timers						
TPM	FF	FF	Async operation FF in PSTOP2	Async operation	static	OFF
PIT	FF static in CPO	FF	static ?? FF in PSTOP2	static ??	static	OFF
LPTMR	FF	FF	Async operation FF in PSTOP2	Async operation	Async operation	Async operation[6]
RTC - 32kHz OSC[4]	FF Async operation in CPO	FF	Async operation FF in PSTOP2	Async operation	Async operation[7]	Async operation[7]
Analog						
16-bit ADC	FF ADACK and ALTCLK clocks only in CPO	FF	ADACK, ALTCLK, and ALTCLK2 clocks only FF in PSTOP2	ADACK and ALTCLK clocks only	static	OFF
CMP[8]	FF HS or LS compare in CPO	FF	HS or LS compare FF in PSTOP2	HS or LS compare	LS compare	LS compare in VLLS1/2/3, OFF in VLLS0
6-bit DAC	FF static in CPO	FF	static FF in PSTOP2	static	static	static, OFF in VLLS0
VREF	FF	FF	FF	FF	static	OFF
12-bit DAC	FF static in CPO	FF	static FF in PSTOP2	static	static	static
Human-machine interfaces						
GPIO	FF GPIO write only in CPO	FF	static output, wakeup input FF in PSTOP2	static output, wakeup input	static, pins latched	OFF, pins latched
TSI0	FF Async operation in CPO	Async operation	Async operation[9] FF in PSTOP2	Async operation[9]	Async operation[9]	Async operation[9]

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 SMC_STOPCTRL[PORPO] bit in the SMC module controls this option.

4. These components remain powered in BAT power mode.
5. the QSPI module may require specific instructions to enter low power prior to the MCU entering low power.
6. System OSC and LPO clock sources are not available in VLLS0. Pulse counting is available in all modes.
7. RTC_CLKOUT is not available. CLKOUT32K can be configured as an alternate path of supplying 32 kHz.
8. CMP in stop or VLPS supports high speed or low speed external pin to pin or external pin to DAC compares. CMP in LLSx 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, LLSx, or VLLSx modes.
9. TSI wakeup from all low power modes is limited to a single selectable pin.

Chapter 9
Security

9.1 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.

9.2 Flash security

The flash module provides security information to the MCU based on the state held by the FTFA_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. 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 (SWD), as well as user code execution of Flash Controller commands. When the flash is secured (FTFA_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.

9.3 Security interactions with other modules

The flash security settings are used by the chip 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.

9.3.1 Security interactions with debug

When flash security is active, the SWD 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 10
Debug

10.1 Introduction

This debug of this device is based on the ARM CoreSight™ architecture and is configured to provide the maximum flexibility as allowed by the restrictions of the pinout and other available resources.

It provides register and memory accessibility from the external debugger interface, basic run/halt control plus 2 breakpoints and 2 watchpoints.

Only one debug interface is supported:

- Serial Wire Debug (SWD)

10.2 Debug port pin descriptions

The debug port pins default after POR to their SWD functionality.

Table 10-1. Serial wire debug pin description

Pin name	Type	Description
SWD_CLK	Input	Serial wire clock This pin is the clock for debug logic when in the serial wire debug mode. This pin is pulled down internally.
SWD_DIO	Input / Output	Serial wire debug data input/output The SWD_DIO pin is used by an external debug tool for communication and device control. This pin is pulled up internally.

10.3 SWD 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 figure found here.

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 using SWD. The MDM-AP is accessible as Debug Access Port 1 with the available registers shown in this table.

Table 10-2. MDM-AP register summary

Address	Register	Description
0x0100_0000	Status	See MDM-AP status register
0x0100_0004	Control	See MDM-AP control register
0x0100_00FC	IDR	Read-only identification register that always reads as 0x001C_0020

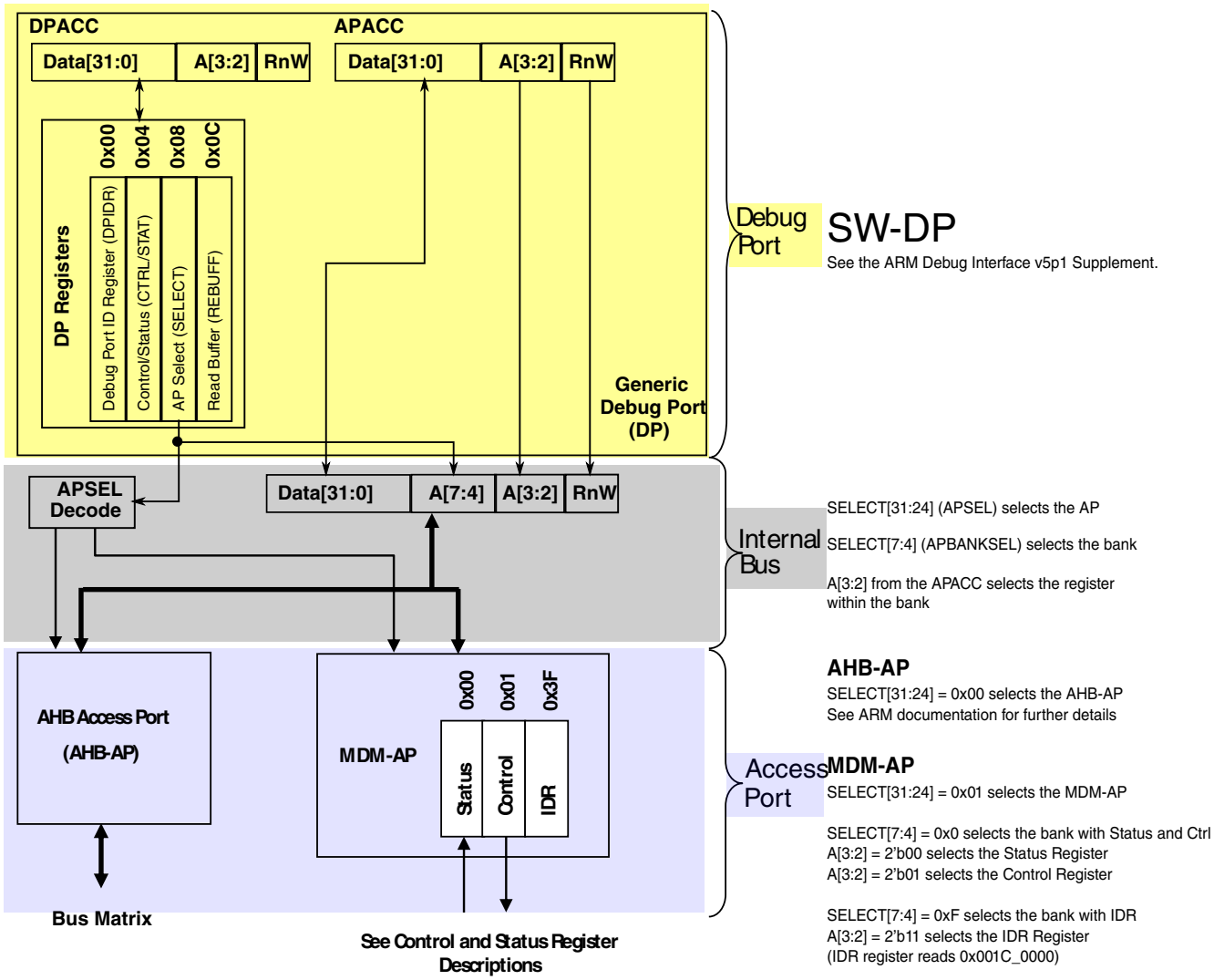


Figure 10-1. MDM AP Addressing

10.3.1 MDM-AP control register

Table 10-3. 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. 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.

Table continues on the next page...

Table 10-3. MDM-AP control register assignments (continued)

Bit	Name	Secure[1]	Description
			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. 0 Normal operation - release the Core from reset along with the rest of the system at the end of system reset sequencing. 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. This bit holds the system in reset when VLLSx modes are exited to allow the debugger time to re-initialize debug IP before the debug session continues. 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. 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 system being held in reset following a VLLSx recovery 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. 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. 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.
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. 0 The timestamp counter continues to count assuming trace is enabled. (default) 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

10.3.2 MDM-AP status register

Table 10-4. 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. 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. 0 System is in reset 1 System is not in reset
4	Reserved	
5	Mass Erase Enable	Indicates if the MCU can be mass erased or not 0 Mass erase is disabled 1 Mass erase is enabled
6	Backdoor Access Key Enable	Indicates if the MCU has the backdoor access key enabled. 0 Disabled 1 Enabled
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. 0 Low Power Stop Mode is not enabled 1 Low Power Stop Mode is enabled 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. This bit is used to throttle SWD_CLK 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. 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.

Table continues on the next page...

Table 10-4. MDM-AP status register assignments (continued)

Bit	Name	Description
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. 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. SLEEPING==0 and SLEEPDEEP==1 indicates stop or VLPS mode.
19 – 31	Reserved for future use	Always read 0.

10.4 Debug resets

The debug system receives the following sources of reset:

- 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 field 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.

10.5 Micro trace buffer (MTB)

The Micro Trace Buffer (MTB) provides a simple execution trace capability for the Cortex-M0+ processor.

When enabled, the MTB records changes in program flow reported by the Cortex-M0+ processor, via the execution trace interface, into a configurable region of the SRAM. Subsequently, an off-chip debugger may extract the trace information, which would allow reconstruction of an instruction flow trace. The MTB does not include any form of load/store data trace capability or tracing of any other information.

In addition to providing the trace capability, the MTB also operates as a simple AHB-Lite SRAM controller. The system bus masters, including the processor, have read/write access to all of the SRAM via the AHB-Lite interface, allowing the memory to be also used to store program and data information. The MTB simultaneously stores the trace information into an attached SRAM and allows bus masters to access the memory. The MTB ensures that trace information write accesses to the SRAM take priority over accesses from the AHB-Lite interface.

The MTB includes trace control registers for configuring and triggering the MTB functions. The MTB also supports triggering via TSTART and TSTOP control functions in the MTB DWT module.

10.6 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 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.

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.

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 to 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.

10.7 Debug and security

When security is enabled (FTFA_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 (FTFA_FSEC[MEEN] = 10), attempts to mass erase via the debug interface are blocked.

Chapter 11
Signal Multiplexing and Signal Descriptions

11.1 Signal multiplexing 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.

11.2 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.

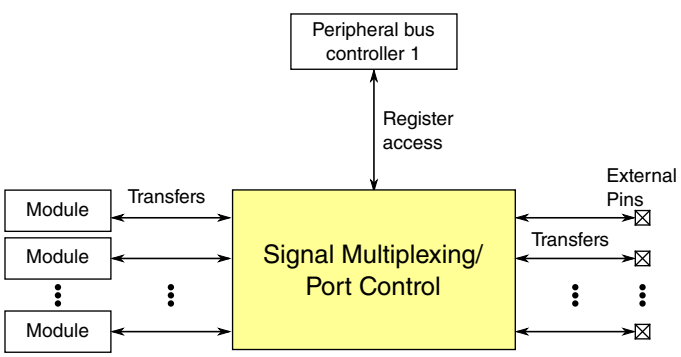


Figure 11-1. Signal multiplexing integration

Table 11-1. 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 11-1. Reference links to related information (continued)

Topic	Related module	Reference
Clocking		Clock Distribution
Register access	Peripheral bus controller	Peripheral bridge

11.2.1 Port control and interrupt module features

- 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.

Table 11-2. Ports summary

Feature	Port A	Port B	Port C	Port D	Port E
Pull select control	Yes	Yes	Yes	Yes	Yes
Pull select at reset	PTA1-PTA5=Pull up, Others=Pull down	Pull down	Pull down	Pull down	Pull down
Pull enable control	Yes	Yes	Yes	Yes	Yes
Pull enable at reset	PTA0-PTA5=Enabled; Others=Disabled	Disabled	Disabled	Disabled	Disabled
Slew rate enable control	Yes	Yes	Yes	Yes	Yes
Slew rate enable at reset	Disabled	Disabled	Disabled	Disabled	Disabled
Passive filter enable control	PTA4=Yes; Others=No	No	No	No	No
Passive filter enable at reset	Disabled	Disabled	Disabled	Disabled	Disabled
Open drain enable control	Yes	Yes	Yes	Yes	Yes
Open drain enable at reset	Disabled	Disabled	Disabled	Disabled	Disabled
Pin mux control	Yes	Yes	Yes	Yes	Yes
Pin mux at reset	PTA0-PTA4=ALT7; Others=ALT0	ALT0	ALT0	ALT0	ALT0
Lock bit	Yes	Yes	Yes	Yes	Yes
Interrupt and DMA request	Yes	Yes	Yes	Yes	Yes
Digital glitch filter	No	No	No	No	No

NOTE

Port E has separate power domain to support QSPI flash

11.2.2 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, see Clock distribution.

11.2.3 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.

11.3 Pinous

11.3.1 KL82 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.

121 MAP BGA	100 LQFP	80 LQFP	64 MAP BGA	64 LQFP	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
B1	1	1	A1	1	PTE0	DISABLED		PTE0	SPI1_PCS1	LPUART1_TX		QSPI0A_DATA3	I2C1_SDA	RTC_CLKOUT
C2	2	2	B1	2	PTE1/LLWU_P0	DISABLED		PTE1/LLWU_P0	SPI1_SCK	LPUART1_RX		QSPI0A_SCLK	I2C1_SCL	SPI1_SIN
C1	3	3	C5	3	PTE2/LLWU_P1	DISABLED		PTE2/LLWU_P1	SPI1_SOUT	LPUART1_CTS_b		QSPI0A_DATA0		SPI1_SCK
D2	4	4	D2	4	PTE3	DISABLED		PTE3	SPI1_PCS2	LPUART1_RTS_b		QSPI0A_DATA2		SPI1_SOUT
F7	5	5	C4	5	VSS	VSS	VSS							
E5	6	6	D3	6	VDDIO_E	VDDIO_E	VDDIO_E							

121 MAP BGA	100 LQFP	80 LQFP	64 MAP BGA	64 LQFP	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
D1	7	7	E2	7	PTE4/ LLWU_P2	DISABLED		PTE4/ LLWU_P2	SPI1_SIN			QSPI0A_ DATA1		
E2	8	8	D1	8	PTE5	DISABLED		PTE5	SPI1_PCS0			QSPI0A_ SS0_B		USB0_SOF_ OUT
E1	9	—	—	—	PTE6/ LLWU_P16	DISABLED		PTE6/ LLWU_P16	SPI1_PCS3			QSPI0B_ DATA3		
F3	10	9	—	—	PTE7	DISABLED		PTE7				QSPI0B_ SCLK		QSPI0A_ SS1_B
F2	11	10	—	—	PTE8	DISABLED		PTE8				QSPI0B_ DATA0		
F1	12	—	—	—	PTE9/ LLWU_P17	DISABLED		PTE9/ LLWU_P17				QSPI0B_ DATA2		
G2	13	—	—	—	PTE10/ LLWU_P18	DISABLED		PTE10/ LLWU_P18				QSPI0B_ DATA1		
G1	14	11	—	—	PTE11	DISABLED		PTE11				QSPI0B_ SS0_B		QSPI0A_ DQS
—	15	12	—	—	VDDIO_E	VDDIO_E	VDDIO_E							
—	16	13	—	9	VSS	VSS	VSS							
H3	—	—	F3	—	VSS	VSS	VSS							
H2	17	14	E1	10	USB0_DP	USB0_DP	USB0_DP							
H1	18	15	F1	11	USB0_DM	USB0_DM	USB0_DM							
J1	19	16	F2	12	USB_VDD	USB_VDD	USB_VDD							
J2	20	—	—	—	NC	NC	NC							
—	21	—	—	—	NC									
K2	—	—	—	—	ADC0_DP0	ADC0_DP0	ADC0_DP0							
K1	—	—	—	—	ADC0_DM0	ADC0_DM0	ADC0_DM0							
F5	22	17	G2	13	VDDA	VDDA	VDDA							
G5	23	18	H3	14	VREFH	VREFH	VREFH							
G6	24	19	H2	15	VREFL	VREFL	VREFL							
F6	25	20	G1	16	VSSA	VSSA	VSSA							
L2	26	21	H1	17	ADC0_DP1	ADC0_DP1	ADC0_DP1							
L1	27	22	G3	18	ADC0_DM1	ADC0_DM1	ADC0_DM1							
L3	28	23	F4	19	VREF_OUT/ CMP0_IN5/ ADC0_SE22	VREF_OUT/ CMP0_IN5/ ADC0_SE22	VREF_OUT/ CMP0_IN5/ ADC0_SE22							
K4	29	24	G4	20	DAC0_OUT/ ADC0_SE23	DAC0_OUT/ ADC0_SE23	DAC0_OUT/ ADC0_SE23							
H6	—	—	—	—	NC	NC	NC							
K5	30	25	F5	21	RTC_ WAKEUP_B	RTC_ WAKEUP_B	RTC_ WAKEUP_B							
L4	31	26	H4	22	XTAL32	XTAL32	XTAL32							
L5	32	27	H5	23	EXTAL32	EXTAL32	EXTAL32							
K6	33	28	G5	24	VBAT	VBAT	VBAT							
—	34	—	—	—	VDD	VDD	VDD							

121 MAP BGA	100 LQFP	80 LQFP	64 MAP BGA	64 LQFP	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
—	35	—	—	—	VSS	VSS	VSS							
L7	36	29	D4	25	PTA0	SWD_CLK	TSI0_CH1	PTA0	LPUART0_ CTS_b	TPM0_CH5		FXIO0_D10	EMVSIM0_ CLK	SWD_CLK
H8	37	30	D5	26	PTA1	TSI0_CH2	TSI0_CH2	PTA1	LPUART0_ RX			FXIO0_D11	EMVSIM0_ IO	
J7	38	31	E5	27	PTA2	TSI0_CH3	TSI0_CH3	PTA2	LPUART0_ TX			FXIO0_D12	EMVSIM0_ PD	
H9	39	32	H6	28	PTA3	SWD_DIO	TSI0_CH4	PTA3	LPUART0_ RTS_b	TPM0_CH0		FXIO0_D13	EMVSIM0_ RST	SWD_DIO
J8	40	33	G6	29	PTA4/ LLWU_P3	NMI_b	TSI0_CH5	PTA4/ LLWU_P3		TPM0_CH1		FXIO0_D14	EMVSIM0_ VCCEN	NMI_b
K7	41	—	—	—	PTA5	DISABLED		PTA5	USB0_ CLKIN	TPM0_CH2		FXIO0_D15		
L10	—	—	—	—	VDD	VDD	VDD							
K10	—	—	—	—	VSS	VSS	VSS							
J9	—	—	—	—	PTA10/ LLWU_P22	DISABLED		PTA10/ LLWU_P22		TPM2_CH0	EMVSIM1_ VCCEN	FXIO0_D16		
H7	—	—	—	—	PTA11/ LLWU_P23	DISABLED		PTA11/ LLWU_P23		TPM2_CH1		FXIO0_D17		
K8	42	—	—	—	PTA12	DISABLED		PTA12		TPM1_CH0		FXIO0_D18		
L8	43	—	—	—	PTA13/ LLWU_P4	DISABLED		PTA13/ LLWU_P4		TPM1_CH1		FXIO0_D19		
K9	44	34	—	—	PTA14	DISABLED		PTA14	SPI0_PCS0	LPUART0_ TX		FXIO0_D20		
L9	45	35	—	—	PTA15	DISABLED		PTA15	SPI0_SCK	LPUART0_ RX		FXIO0_D21		
J10	46	36	—	—	PTA16	DISABLED		PTA16	SPI0_SOUT	LPUART0_ CTS_b		FXIO0_D22		
H10	47	37	—	—	PTA17	DISABLED		PTA17	SPI0_SIN	LPUART0_ RTS_b		FXIO0_D23		
E6	48	38	H7	30	VDD	VDD	VDD							
G7	49	39	G7	31	VSS	VSS	VSS							
L11	50	40	H8	32	PTA18	EXTAL0	EXTAL0	PTA18			TPM_ CLKIN0			
K11	51	41	G8	33	PTA19	XTAL0	XTAL0	PTA19			TPM_ CLKIN1		LPTMR0_ ALT1/ LPTMR1_ ALT1	
J11	52	42	F8	34	RESET_b	RESET_b	RESET_b							
H11	—	—	—	—	PTA29	DISABLED		PTA29						
G11	53	43	E6	35	PTB0/ LLWU_P5	ADC0_SE8/ TSI0_CH0	ADC0_SE8/ TSI0_CH0	PTB0/ LLWU_P5	I2C0_SCL	TPM1_CH0				FXIO0_D0
G10	54	44	—	—	PTB1	ADC0_SE9/ TSI0_CH6	ADC0_SE9/ TSI0_CH6	PTB1	I2C0_SDA	TPM1_CH1				FXIO0_D1

121 MAP BGA	100 LQFP	80 LQFP	64 MAP BGA	64 LQFP	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
G9	55	—	—	—	PTB2	ADC0_ SE12/ TSI0_CH7	ADC0_ SE12/ TSI0_CH7	PTB2	I2C0_SCL	LPUART0_ RTS_b				FXIO0_D2
G8	56	—	—	—	PTB3	ADC0_ SE13/ TSI0_CH8	ADC0_ SE13/ TSI0_CH8	PTB3	I2C0_SDA	LPUART0_ CTS_b				FXIO0_D3
B11	—	45	F7	36	PTB4	DISABLED		PTB4	EMVSIM1_ IO					
C11	—	46	F6	37	PTB5	DISABLED		PTB5	EMVSIM1_ CLK					
F11	—	47	E7	38	PTB6	DISABLED		PTB6	EMVSIM1_ VCCEN					
E11	—	48	E8	39	PTB7	DISABLED		PTB7	EMVSIM1_ PD					
D11	—	49	D7	40	PTB8	DISABLED		PTB8	EMVSIM1_ RST					
E10	57	—	—	—	PTB9	DISABLED		PTB9	SPI1_PCS1					
D10	58	—	—	—	PTB10	DISABLED		PTB10	SPI1_PCS0					FXIO0_D4
C10	59	50	—	—	PTB11	DISABLED		PTB11	SPI1_SCK					FXIO0_D5
L6	60	—	—	—	VSS	VSS	VSS							
E7	61	—	—	—	VDD	VDD	VDD							
B10	62	51	—	—	PTB16	TSI0_CH9	TSI0_CH9	PTB16	SPI1_SOUT	LPUART0_ RX	TPM_ CLKIN0		EWM_IN	
E9	63	52	—	—	PTB17	TSI0_CH10	TSI0_CH10	PTB17	SPI1_SIN	LPUART0_ TX	TPM_ CLKIN1		EWM_OUT_ b	
D9	64	53	D6	41	PTB18	TSI0_CH11	TSI0_CH11	PTB18		TPM2_CH0				FXIO0_D6
C9	65	54	C7	42	PTB19	TSI0_CH12	TSI0_CH12	PTB19		TPM2_CH1				FXIO0_D7
F10	66	—	—	—	PTB20	DISABLED		PTB20					CMP0_OUT	FXIO0_D8
F9	67	—	—	—	PTB21	DISABLED		PTB21						FXIO0_D9
F8	68	—	—	—	PTB22	DISABLED		PTB22						FXIO0_D10
E8	69	—	—	—	PTB23	DISABLED		PTB23		SPI0_PCS5				FXIO0_D11
B9	70	55	D8	43	PTC0	ADC0_ SE14/ TSI0_CH13	ADC0_ SE14/ TSI0_CH13	PTC0	SPI0_PCS4	EXTRG_IN	USB0_SOF_ OUT			FXIO0_D12
D8	71	56	C6	44	PTC1/ LLWU_P6	ADC0_ SE15/ TSI0_CH14	ADC0_ SE15/ TSI0_CH14	PTC1/ LLWU_P6	SPI0_PCS3	LPUART1_ RTS_b	TPM0_CH0			FXIO0_D13
C8	72	57	B7	45	PTC2	ADC0_ SE4b/ TSI0_CH15	ADC0_ SE4b/ TSI0_CH15	PTC2	SPI0_PCS2	LPUART1_ CTS_b	TPM0_CH1			
B8	73	58	C8	46	PTC3/ LLWU_P7	DISABLED		PTC3/ LLWU_P7	SPI0_PCS1	LPUART1_ RX	TPM0_CH2	CLKOUT		
—	74	59	E3	47	VSS	VSS	VSS							
—	75	60	E4	48	VDD	VDD	VDD							
A8	76	61	B8	49	PTC4/ LLWU_P8	DISABLED		PTC4/ LLWU_P8	SPI0_PCS0	LPUART1_ TX	TPM0_CH3			

121 MAP BGA	100 LQFP	80 LQFP	64 MAP BGA	64 LQFP	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
D7	77	62	A8	50	PTC5/ LLWU_P9	DISABLED		PTC5/ LLWU_P9	SPI0_SCK	LPTMR0_ ALT2/ LPTMR1_ ALT2			CMP0_OUT	TPM0_CH2
C7	78	63	A7	51	PTC6/ LLWU_P10	CMP0_IN0	CMP0_IN0	PTC6/ LLWU_P10	SPI0_SOUT	EXTRG_IN				FXIO0_D14
B7	79	64	B6	52	PTC7	CMP0_IN1	CMP0_IN1	PTC7	SPI0_SIN	USB0_SOF_ OUT				FXIO0_D15
A7	80	65	A6	53	PTC8	CMP0_IN2	CMP0_IN2	PTC8						FXIO0_D16
D6	81	66	B5	54	PTC9	CMP0_IN3	CMP0_IN3	PTC9						FXIO0_D17
C6	82	67	B4	55	PTC10	DISABLED		PTC10	I2C1_SCL					FXIO0_D18
C5	83	68	A5	56	PTC11/ LLWU_P11	DISABLED		PTC11/ LLWU_P11	I2C1_SDA					FXIO0_D19
B6	84	69	—	—	PTC12	DISABLED		PTC12			TPM_ CLKIN0			
A6	85	70	—	—	PTC13	DISABLED		PTC13			TPM_ CLKIN1			
A5	86	—	—	—	PTC14	DISABLED		PTC14						FXIO0_D20
B5	87	—	—	—	PTC15	DISABLED		PTC15						FXIO0_D21
—	88	—	—	—	VSS	VSS	VSS							
—	89	—	—	—	VDD	VDD	VDD							
D5	—	71	—	—	PTC16	DISABLED		PTC16						
C4	90	72	—	—	PTC17	DISABLED		PTC17						
B4	—	—	—	—	PTC18	DISABLED		PTC18						
A4	—	—	—	—	PTC19	DISABLED		PTC19						
D4	91	73	C3	57	PTD0/ LLWU_P12	DISABLED		PTD0/ LLWU_P12	SPI0_PCS0	LPUART2_ RTS_b				FXIO0_D22
D3	92	74	A4	58	PTD1	ADC0_SE5b	ADC0_SE5b	PTD1	SPI0_SCK	LPUART2_ CTS_b				FXIO0_D23
C3	93	75	C2	59	PTD2/ LLWU_P13	DISABLED		PTD2/ LLWU_P13	SPI0_SOUT	LPUART2_ RX				I2C0_SCL
B3	94	76	B3	60	PTD3	DISABLED		PTD3	SPI0_SIN	LPUART2_ TX				I2C0_SDA
A3	95	77	A3	61	PTD4/ LLWU_P14	DISABLED		PTD4/ LLWU_P14	SPI0_PCS1	LPUART0_ RTS_b	TPM0_CH4		EWM_IN	SPI1_PCS0
A2	96	78	C1	62	PTD5	ADC0_SE6b	ADC0_SE6b	PTD5	SPI0_PCS2	LPUART0_ CTS_b	TPM0_CH5		EWM_OUT_ b	SPI1_SCK
B2	97	79	B2	63	PTD6/ LLWU_P15	ADC0_SE7b	ADC0_SE7b	PTD6/ LLWU_P15	SPI0_PCS3	LPUART0_ RX				SPI1_SOUT
—	98	—	—	—	VSS	VSS	VSS							
—	99	—	—	—	VDD	VDD	VDD							
A1	100	80	A2	64	PTD7	DISABLED		PTD7		LPUART0_ TX				SPI1_SIN
A10	—	—	—	—	PTD8/ LLWU_P24	DISABLED		PTD8/ LLWU_P24	I2C0_SCL					FXIO0_D24

121 MAP BGA	100 LQFP	80 LQFP	64 MAP BGA	64 LQFP	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
A9	—	—	—	—	PTD9	DISABLED		PTD9	I2C0_SDA					FXIO0_D25
E4	—	—	—	—	PTD10	DISABLED		PTD10						FXIO0_D26
E3	—	—	—	—	PTD11/ LLWU_P25	DISABLED		PTD11/ LLWU_P25						FXIO0_D27
F4	—	—	—	—	PTD12	DISABLED		PTD12						FXIO0_D28
G3	—	—	—	—	PTD13	DISABLED		PTD13						FXIO0_D29
G4	—	—	—	—	PTD14	DISABLED		PTD14						FXIO0_D30
H4	—	—	—	—	PTD15	DISABLED		PTD15						FXIO0_D31
A11	—	—	—	—	NC	NC	NC							
J6	—	—	—	—	NC	NC	NC							
J4	—	—	—	—	NC	NC	NC							
H5	—	—	—	—	NC	NC	NC							
J3	—	—	—	—	NC	NC	NC							
J5	—	—	—	—	NC	NC	NC							
K3	—	—	—	—	NC	NC	NC							
121	100	80	64	64										

11.3.2 KL82 Pinouts

The below figures show the pinout diagrams 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	
A	PTD7	PTD5	PTD4/ LLWU_P14	PTC19	PTC14	PTC13	PTC8	PTC4/ LLWU_P8	PTD9	PTD8/ LLWU_P24	NC	A
B	PTE0	PTD6/ LLWU_P15	PTD3	PTC18	PTC15	PTC12	PTC7	PTC3/ LLWU_P7	PTC0	PTB16	PTB4	B
C	PTE2/ LLWU_P1	PTE1/ LLWU_P0	PTD2/ LLWU_P13	PTC17	PTC11/ LLWU_P11	PTC10	PTC6/ LLWU_P10	PTC2	PTB19	PTB11	PTB5	C
D	PTE4/ LLWU_P2	PTE3	PTD1	PTD0/ LLWU_P12	PTC16	PTC9	PTC5/ LLWU_P9	PTC1/ LLWU_P6	PTB18	PTB10	PTB8	D
E	PTE6/ LLWU_P16	PTE5	PTD11/ LLWU_P25	PTD10	VDDIO_E	VDD	VDD	PTB23	PTB17	PTB9	PTB7	E
F	PTE9/ LLWU_P17	PTE8	PTE7	PTD12	VDDA	VSSA	VSS	PTB22	PTB21	PTB20	PTB6	F
G	PTE11	PTE10/ LLWU_P18	PTD13	PTD14	VREFH	VREFL	VSS	PTB3	PTB2	PTB1	PTB0/ LLWU_P5	G
H	USB0_DM	USB0_DP	VSS	PTD15	NC	NC	PTA11/ LLWU_P23	PTA1	PTA3	PTA17	PTA29	H
J	USB_VDD	NC	NC	NC	NC	NC	PTA2	PTA4/ LLWU_P3	PTA10/ LLWU_P22	PTA16	RESET_b	J
K	ADC0_DM0	ADC0_DP0	NC	DAC0_OUT/ ADC0_SE23	RTC_WAKEUP_B	VBAT	PTA5	PTA12	PTA14	VSS	PTA19	K
L	ADC0_DM1	ADC0_DP1	VREF_OUT/ CMP0_IN5/ ADC0_SE22	XTAL32	EXTAL32	VSS	PTA0	PTA13/ LLWU_P4	PTA15	VDD	PTA18	L
	1	2	3	4	5	6	7	8	9	10	11	

Figure 11-2. KL82 121-pin MAPBGA pinout diagram

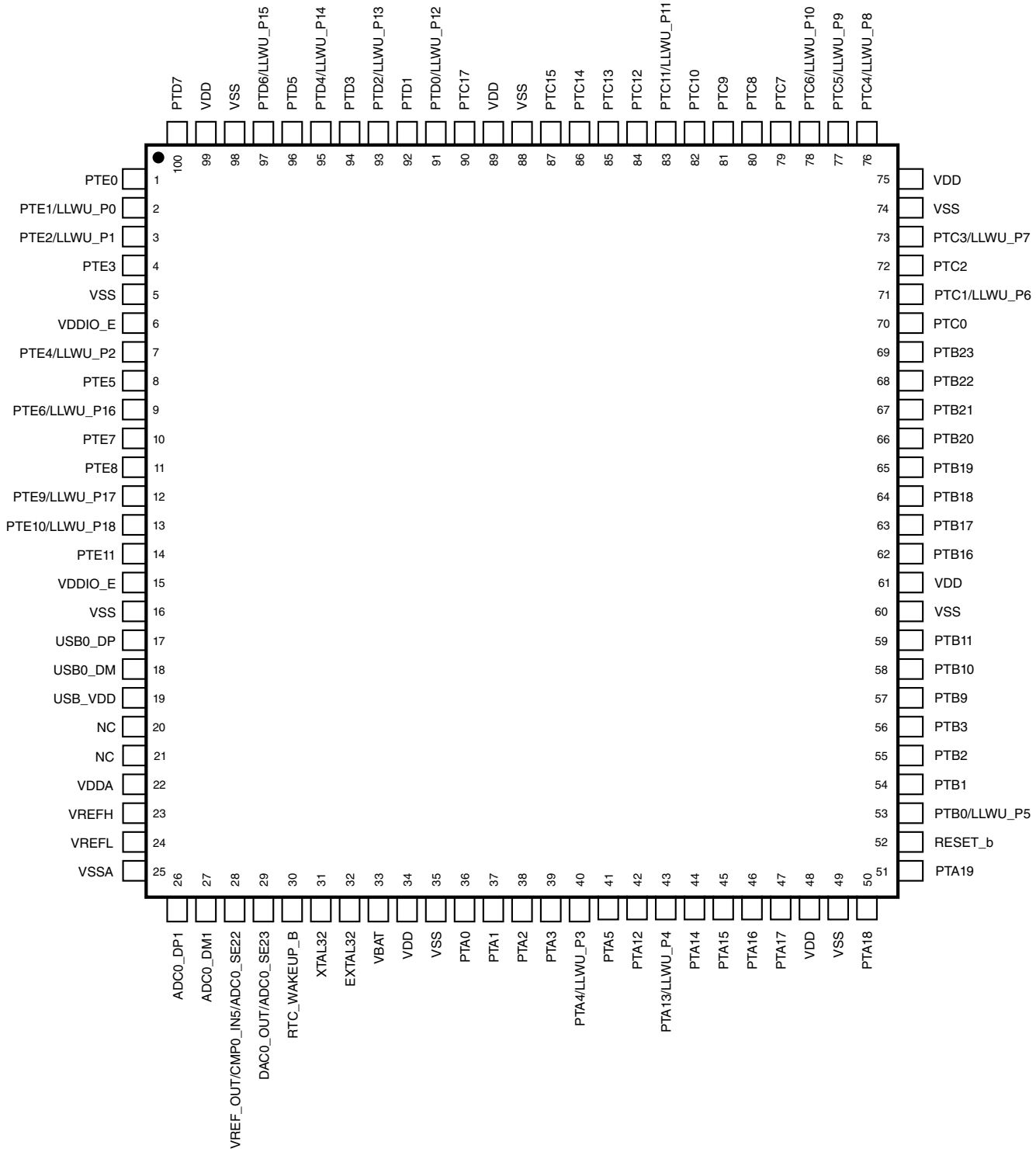


Figure 11-3. KL82 100-pin LQFP pinout diagram

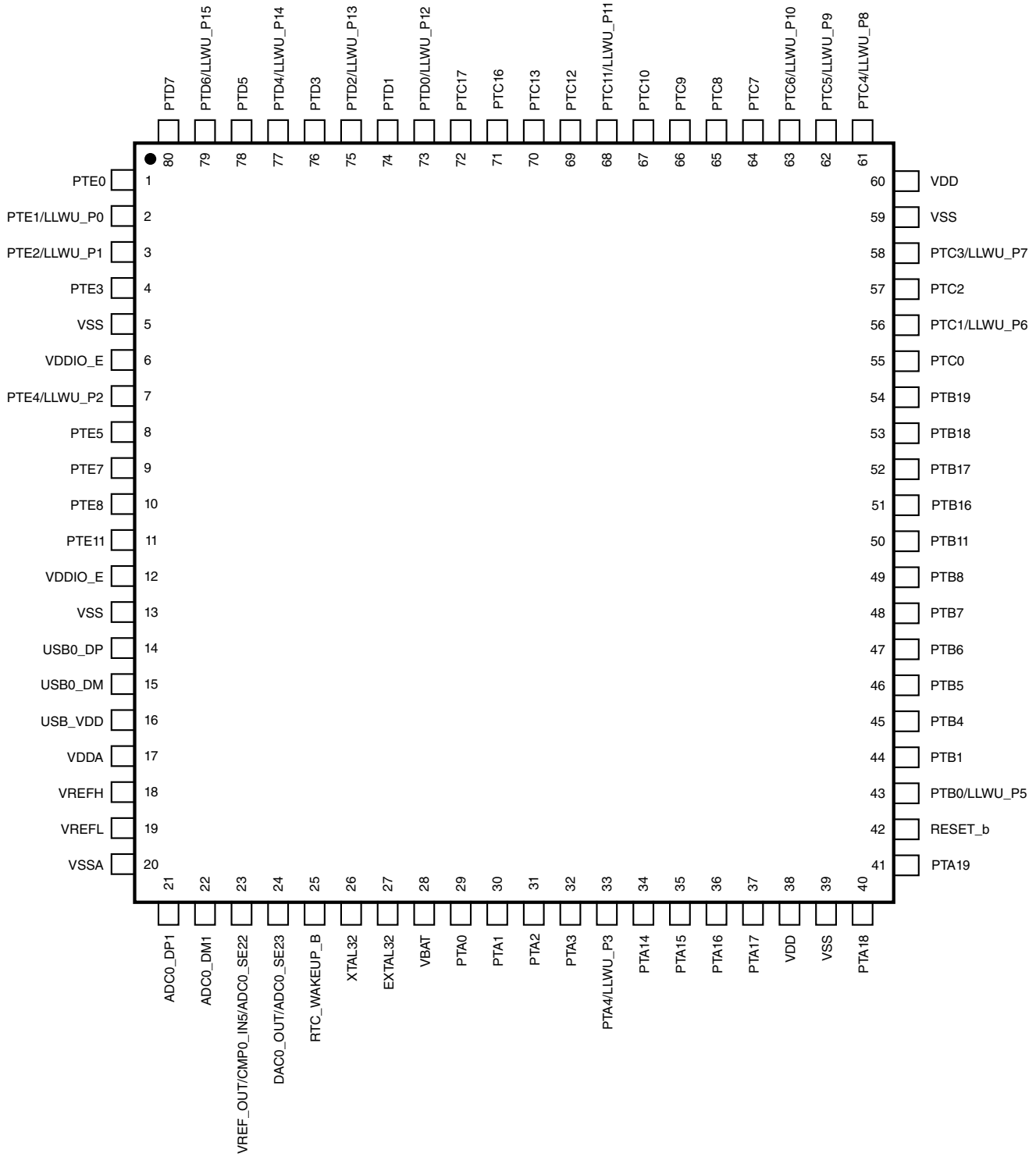
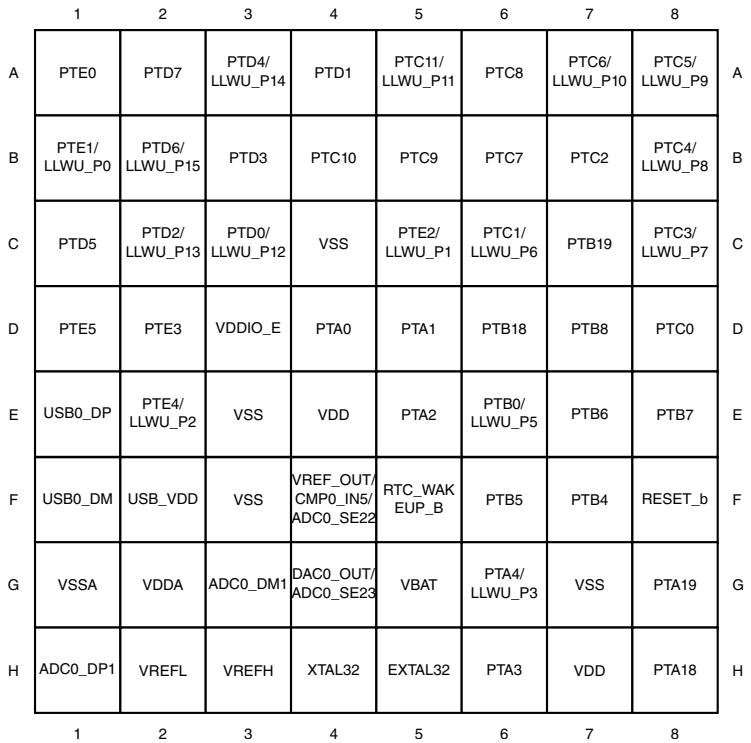


Figure 11-4. KL82 80-pin LQFP pinout diagram



	1	2	3	4	5	6	7	8	
A	PTE0	PTD7	PTD4/ LLWU_P14	PTD1	PTC11/ LLWU_P11	PTC8	PTC6/ LLWU_P10	PTC5/ LLWU_P9	A
B	PTE1/ LLWU_P0	PTD6/ LLWU_P15	PTD3	PTC10	PTC9	PTC7	PTC2	PTC4/ LLWU_P8	B
C	PTD5	PTD2/ LLWU_P13	PTD0/ LLWU_P12	VSS	PTE2/ LLWU_P1	PTC1/ LLWU_P6	PTB19	PTC3/ LLWU_P7	C
D	PTE5	PTE3	VDDIO_E	PTA0	PTA1	PTB18	PTB8	PTC0	D
E	USB0_DP	PTE4/ LLWU_P2	VSS	VDD	PTA2	PTB0/ LLWU_P5	PTB6	PTB7	E
F	USB0_DM	USB_VDD	VSS	VREF_OUT/ CMP0_IN5/ ADC0_SE22	RTC_WAKEUP_B	PTB5	PTB4	RESET_b	F
G	VSSA	VDDA	ADC0_DM1	DAC0_OUT/ ADC0_SE23	VBAT	PTA4/ LLWU_P3	VSS	PTA19	G
H	ADC0_DP1	VREFL	VREFH	XTAL32	EXTAL32	PTA3	VDD	PTA18	H
	1	2	3	4	5	6	7	8	

Figure 11-5. KL82 64-pin MAPBGA pinout diagram

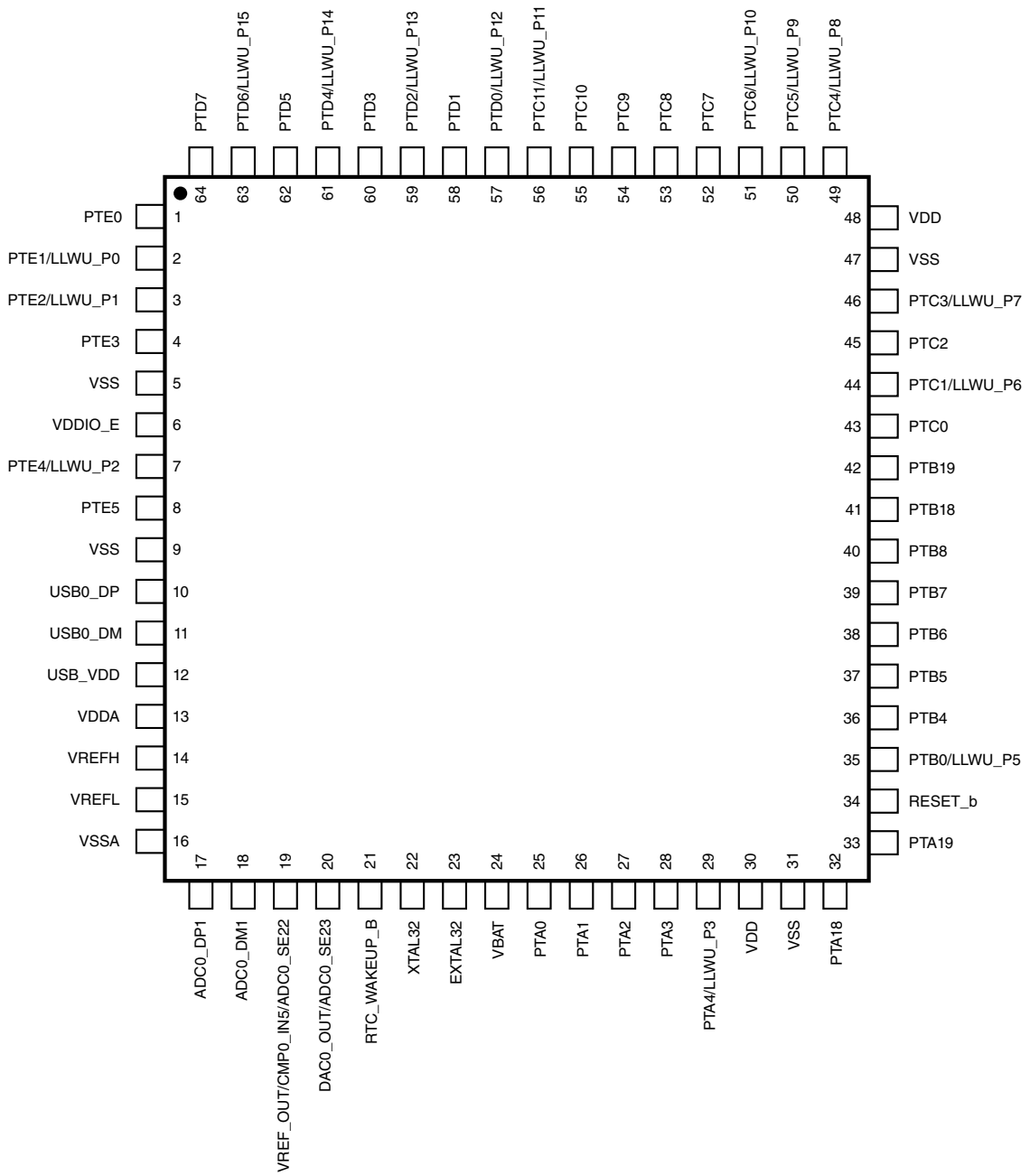


Figure 11-6. KL82 64-pin LQFP pinout diagram

NOTE

The 100-, 64-pin LQFP and 64-pin MAPBGA packages for this product are not yet available, however they are included in a Package Your Way program for KL MCUs. Please visit nxp.com/KPYW for more details.

11.4 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.

11.4.1 Core Modules

Table 11-3. SWD Signal Descriptions

Chip signal name	Module signal name	Description	I/O
SWD_DIO	SWD_DIO	Serial Wire Data	I/O
SWD_CLK	SWD_CLK	Serial Wire Clock	I

11.4.2 System modules

Table 11-4. System signal descriptions

Chip signal name	Module signal name	Description	I/O
NMI_b	—	Non-maskable interrupt **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
RESET_b	—	Reset bi-directional signal	I/O
VDD	—	MCU power	I
VDDIO_E	PTE	MCU power for IOs on PTE	I
VDDA	—	MCU analog power	I
VSS	—	MCU ground	I
VREFH	—	MCU analog voltage reference-high	I
VREFL	—	MCU analog voltage reference--low	I

Table 11-5. 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
EWM_OUT_ b	$\overline{\text{EWM\_out}}$	EWM reset out signal	O

Table 11-6. LLWU signal descriptions

Chip signal name	Module signal name	Description	I/O
LLWU_Pn	LLWU_Pn	Wakeup inputs	I

Table 11-7. EMVSIM0 signal descriptions

Chip signal name	Module signal name	Description	I/O
EMVSIM0_CLK	EMVSIM_SCLK	Card Clock. Clock to Smart Card.	O
EMVSIM0_IO	EMVSIM_IO	Card Data Line. Bi-directional data line.	I/O
EMVSIM0_PD	EMVSIM_PD	Card Presence Detect. Signal indicating presence or removal of card	I
EMVSIM0_RST	EMVSIM_SRST	Card Reset. Reset signal to Smart Card	O
EMVSIM0_VCCEN	EMVSIM_VCC_EN	Card Power Enable. This signal controls the power to Smart Card	O

Table 11-8. EMVSIM1 signal descriptions

Chip signal name	Module signal name	Description	I/O
EMVSIM1_CLK	EMVSIM_SCLK	Card Clock. Clock to Smart Card.	O
EMVSIM1_IO	EMVSIM_IO	Card Data Line. Bi-directional data line.	I/O
EMVSIM1_PD	EMVSIM_PD	Card Presence Detect. Signal indicating presence or removal of card	I
EMVSIM1_RST	EMVSIM_SRST	Card Reset. Reset signal to Smart Card	O
EMVSIM1_VCCEN	EMVSIM_VCC_EN	Card Power Enable. This signal controls the power to Smart Card	O

11.4.3 Clock Modules

Table 11-9. 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 11-10. RTC OSC signal descriptions

Chip signal name	Module signal name	Description	I/O
EXTAL32	EXTAL32	Analog input of the RTC oscillator	I
XTAL32	XTAL32	Analog output of the RTC oscillator module	O

11.4.4 Memories and memory interfaces

Table 11-11. QSPI signal description

Chip signal name	Module signal Name	Description	I/O
QSPI0A_SS0_B	PCSFA1	Peripheral Chip Select Flash A1. This signal is the chip select for the serial flash device A1. A1 represents the first device in a dual-die package flash A or the first of the two flash devices that share IOFA.	O
QSPI0A_SS1_B	PCSFA2	Peripheral Chip Select Flash A2. This signal is the chip select for the serial flash device A2. A2 represents the second device in a dual-die package flash A or the second of the two flash devices that share IOFA.	O
QSPI0B_SS0_B	PCSFB1	Peripheral Chip Select Flash B1. This signal is the chip select for the serial flash device B1. B1 represents the first device in a dual-die package flash B or the first of the two flash devices that share IOFB.	O
QSPI0A_SCLK	SCKFA	Serial Clock Flash A. This signal is the serial clock output to the serial flash device A.	O
QSPI0B_SCLK	SCKFB	Serial Clock Flash B. This signal is the serial clock output to the serial flash device B.	O
QSPI0B_DATA3 QSPI0B_DATA2 QSPI0B_DATA1 QSPI0B_DATA0 QSPI0A_DATA3 QSPI0A_DATA2 QSPI0A_DATA1 QSPI0A_DATA0	IOFA[7:0]	Serial I/O Flash A. These signals are the data I/O lines to/from the serial flash device A. Note that the signal pins of the serial flash device may change their function according to the SFM Command executed, leaving them as control inputs when Single and Dual Instructions are executed. The module supports driving these inputs to dedicated values.	I/O
QSPI0B_DATA3 QSPI0B_DATA2 QSPI0B_DATA1 QSPI0B_DATA0	IOFB[3:0]	Serial I/O Flash B. These signals are the data I/O lines to/from the serial flash device B. Note that the signal pins of the serial flash device may change their function according to the SFM Command executed, leaving them as control inputs when Single and Dual Instructions are executed. The module supports driving these inputs to dedicated values.	I/O

Table continues on the next page...

Table 11-11. QSPI signal description (continued)

Chip signal name	Module signal Name	Description	I/O
QSPI0A_DQS	DQSFA	Data Strobe signal Flash A. Data strobe signal for port A. Some flash vendors provide the DQS signal to which the read data is aligned in DDR mode.	I

11.4.5 Analog

Table 11-12. ADC0 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
ADC0_DP[1:0]	DADP1–DADP0	Differential analog channel inputs	I
ADC0_DM[1:0]	DADM1–DADM0	Differential Analog Channel Inputs	I
ADC0_SEn	ADn	Single-Ended Analog Channel Inputs[1]	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

1. See ADC channel assignment for the n.

Table 11-13. CMP0 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
CMP0_INn, n=[5,3:0]	INn, n=[5,3:0]	Analog voltage inputs, see CMP input connection for more details about the n.	I
CMP0_OUT	CMPO	Comparator output	O

NOTE

There is no CMP0_IN[4] coming from pad.

Table 11-14. DAC0 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
DAC0_OUT	—	DAC output	O

Table 11-15. VREF Signal Descriptions

Chip signal name	Module signal name	Description	I/O
VREF_OUT	VREF_OUT	Internally-generated Voltage Reference output	O

11.4.6 Timer Modules

Table 11-16. LPTMR0 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
LPTMR0_ALT[2:1]	LPTMR_ALTn	Pulse Counter Input	I

Table 11-17. LPTMR1 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
LPTMR1_ALT[2:1]	LPTMR_ALTn	Pulse Counter Input	I

Table 11-18. RTC Signal Descriptions

Chip signal name	Module signal name	Description	I/O
VBAT	—	Backup battery supply for RTC and VBAT register file	I
EXTAL32	EXTAL32	32.768 kHz oscillator input	I
XTAL32	XTAL32	32.768 kHz oscillator output	O
RTC_CLKOUT	RTC_CLKOUT	1 Hz square-wave output or OSCERCLK	O
RTC_WAKEUP_B	$\overline{\text{RTC\_WAKEUP}}$	Wakeup for external device	I/O

Table 11-19. TPM0 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
TPM_CLKIN[1:0]	TPM_EXTCLK	External clock. TPM external clock can be selected to increment the TPM counter on every rising edge synchronized to the counter clock.	I
TPM0_CH[5:0]	TPM_CHn	A TPM channel pin is configured as output when configured in an output compare or PWM mode and the TPM counter is enabled, otherwise the TPM channel pin is an input.	I/O

Table 11-20. TPM1 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
TPM_CLKIN[1:0]	TPM_EXTCLK	External clock. TPM external clock can be selected to increment the TPM counter on every rising edge synchronized to the counter clock.	I
TPM1_CH[1:0]	TPM_CHn	A TPM channel pin is configured as output when configured in an output compare or PWM mode and the TPM counter is enabled, otherwise the TPM channel pin is an input.	I/O

Table 11-21. TPM2 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
TPM_CLKIN[1:0]	TPM_EXTCLK	External clock. TPM external clock can be selected to increment the TPM counter on every rising edge synchronized to the counter clock.	I
TPM1_CH[1:0]	TPM_CHn	A TPM channel pin is configured as output when configured in an output compare or PWM mode and the TPM counter is enabled, otherwise the TPM channel pin is an input.	I/O

11.4.7 Communication interfaces

Table 11-22. 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
USB0_CLKIN	—	Alternate USB clock input	I
USB_VDD	—	USB domain power supply, 3.3 V.	I
USB0_SOF_OUT	—	USB start of frame signal. Can be used to make the USB start of frame available for external synchronization.	O

Table 11-23. SPI0 signal descriptions

Chip signal name	Module signal name	Description	I/O
SPI0_PCS0	PCS0/$\overline{SS}$	Peripheral Chip Select 0 (O) in the master mode and Slave Select (I) in the slave mode	I/O
SPI0_PCS[1:3]	PCS[1:3]	Peripheral Chip Selects 1–3 in the master mode	O
SPI0_PCS4	PCS4	Peripheral Chip Select 4 in the master mode	O
SPI0_PCS5	PCS5	Peripheral Chip Select 5 /Peripheral Chip Select Strobe in the master mode	O

Table continues on the next page...

Table 11-23. SPI0 signal descriptions (continued)

Chip signal name	Module signal name	Description	I/O
SPI0_SIN	SIN	Serial Data In	I
SPI0_SOUT	SOUT	Serial Data Out	O
SPI0_SCK	SCK	Serial Clock (O) in the master mode and Serial Clock (I) in the slave mode	I/O

Table 11-24. SPI1 signal descriptions

Chip signal name	Module signal name	Description	I/O
SPI1_PCS0	PCS0/$\overline{SS}$	Peripheral Chip Select 0 (O) in the master mode and Slave Select (I) in the slave mode	I/O
SPI1_PCS[1:3]	PCS[1:3]	Peripheral Chip Selects 1–3 in the master mode	O
SPI1_SIN	SIN	Serial Data In	I
SPI1_SOUT	SOUT	Serial Data Out	O
SPI1_SCK	SCK	Serial Clock (O) in the master mode and Serial Clock (I) in the slave mode	I/O

Table 11-25. I2C0 signal descriptions

Chip signal name	Module signal name	Description	I/O
I2C0_SCL	SCL	Bidirectional serial clock line of the I2C system.	I/O
I2C0_SDA	SDA	Bidirectional serial data line of the I2C system.	I/O

Table 11-26. I2C1 signal descriptions

Chip signal name	Module signal name	Description	I/O
I2C1_SCL	SCL	Bidirectional serial clock line of the I2C system.	I/O
I2C1_SDA	SDA	Bidirectional serial data line of the I2C system.	I/O

Table 11-27. LPUART0 signal descriptions

Chip signal name	Module signal name	Description	I/O
LPUART0_CTS_b	LPUART_CTS	Clear to Send	I
LPUART0_RTS_b	LPUART_RTS	Request to send	O
LPUART0_TX	LPUART_TX	Transmit data. This pin is normally an output, but is an input (tristated) in single wire mode whenever the transmitter is disabled or transmit direction is configured for receive data.	I/O
LPUART0_RX	LPUART_RX	Receive Data	I

Table 11-28. LPUART1 signal descriptions

Chip signal name	Module signal name	Description	I/O
LPUART1_CTS_b	LPUART_CTS	Clear to Send	I
LPUART1_RTS_b	LPUART_RTS	Request to send	O
LPUART1_TX	LPUART_TX	Transmit data. This pin is normally an output, but is an input (tristated) in single wire mode whenever the transmitter is disabled or transmit direction is configured for receive data.	I/O
LPUART1_RX	LPUART_RX	Receive Data	I

Table 11-29. LPUART2 signal descriptions

Chip signal name	Module signal name	Description	I/O
LPUART2_CTS_b	LPUART_CTS	Clear to Send	I
LPUART2_RTS_b	LPUART_RTS	Request to send	O
LPUART2_TX	LPUART_TX	Transmit data. This pin is normally an output, but is an input (tristated) in single wire mode whenever the transmitter is disabled or transmit direction is configured for receive data.	I/O
LPUART2_RX	LPUART_RX	Receive Data	I

Table 11-30. FlexIO signal descriptions

Chip signal name	Module signal name	Description	I/O
FXIO0_Dn(n=0-31)	FXIO_Dn (n=0...31)	Bidirectional FlexIO Shifter and Timer pin inputs/outputs	I/O

Table 11-31. EMVSIM0 signal descriptions

Chip signal name	Module signal name	Description	I/O
EMVSIM0_ CLK	EMVSIM_SCLK	Card Clock. Clock to Smart Card.	O
EMVSIM0_ IO	EMVSIM_IO	Card Data Line. Bi-directional data line.	I/O
EMVSIM0_ PD	EMVSIM_PD	Card Presence Detect. Signal indicating presence or removal of card	I
EMVSIM0_ RST	EMVSIM_SRST	Card Reset. Reset signal to Smart Card	O
EMVSIM0_ VCCEN	EMVSIM_VCC_EN	Card Power Enable. This signal controls the power to Smart Card	O

Table 11-32. EMVSIM1 signal descriptions

Chip signal name	Module signal name	Description	I/O
EMVSIM1_ CLK	EMVSIM_SCLK	Card Clock. Clock to Smart Card.	O
EMVSIM1_ IO	EMVSIM_IO	Card Data Line. Bi-directional data line.	I/O

Table continues on the next page...

Table 11-32. EMVSIM1 signal descriptions (continued)

Chip signal name	Module signal name	Description	I/O
EMVSIM1_ PD	EMVSIM_PD	Card Presence Detect. Signal indicating presence or removal of card	I
EMVSIM1_ RST	EMVSIM_SRST	Card Reset. Reset signal to Smart Card	O
EMVSIM1_ VCCEN	EMVSIM_VCC_EN	Card Power Enable. This signal controls the power to Smart Card	O

11.4.8 Human-machine interfaces (HMI)

Table 11-33. 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.

Table 11-34. TSI0 signal descriptions

Chip signal name	Module signal name	Description	I/O
TSI0_CH[15:0]	TSI[15:0]	TSI capacitive pins. Switches driver that connects directly to the electrode pins TSI[15:0] can operate as GPIO pins.	I/O

Chapter 12
Port Control and Interrupts (PORT)

12.1 Introduction

12.2 Overview

The Port Control and Interrupt (PORT) module provides support for port control, 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.

12.2.1 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 wake-up in low-power modes
  - Pin interrupt is functional in all digital pin muxing modes
  - Peripheral trigger output (active high, low) configured per pin
- Port control
  - Individual pull control fields with pullup, pulldown, and pull-disable support on selected pins
  - Individual slew rate field supporting fast and slow slew rates on selected pins

- Individual input passive filter field supporting enable and disable of the individual input passive filter on selected pins
- Individual open drain field supporting enable and disable of the individual open drain output on selected pins
- 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.

12.2.2 Modes of operation

12.2.2.1 Run mode

In Run mode, the PORT operates normally.

12.2.2.2 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.

12.2.2.3 Stop mode

In Stop mode, the PORT can be configured to exit the Low-Power mode via an asynchronous wake-up signal if an enabled interrupt is detected.

12.2.2.4 Debug mode

In Debug mode, PORT operates normally.

12.3 External signal description

The table found here describes the PORT external signal.

Table 12-1. 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.

12.4 Detailed signal description

The table found here contains the detailed signal description for the PORT interface.

Table 12-2. PORT interface—detailed signal description

Signal	I/O	Description	
PORTx[31:0]	I/O	External interrupt.	
		State meaning	Asserted—pin is logic 1. Negated—pin is logic 0.
		Timing	Assertion—may occur at any time and can assert asynchronously to the system clock. Negation—may occur at any time and can assert asynchronously to the system clock.

12.5 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	12.5.1/246
4004_9004	Pin Control Register n (PORTA_PCR1)	32	R/W	See section	12.5.1/246
4004_9008	Pin Control Register n (PORTA_PCR2)	32	R/W	See section	12.5.1/246
4004_900C	Pin Control Register n (PORTA_PCR3)	32	R/W	See section	12.5.1/246
4004_9010	Pin Control Register n (PORTA_PCR4)	32	R/W	See section	12.5.1/246
4004_9014	Pin Control Register n (PORTA_PCR5)	32	R/W	See section	12.5.1/246
4004_9018	Pin Control Register n (PORTA_PCR6)	32	R/W	See section	12.5.1/246
4004_901C	Pin Control Register n (PORTA_PCR7)	32	R/W	See section	12.5.1/246
4004_9020	Pin Control Register n (PORTA_PCR8)	32	R/W	See section	12.5.1/246
4004_9024	Pin Control Register n (PORTA_PCR9)	32	R/W	See section	12.5.1/246
4004_9028	Pin Control Register n (PORTA_PCR10)	32	R/W	See section	12.5.1/246
4004_902C	Pin Control Register n (PORTA_PCR11)	32	R/W	See section	12.5.1/246

Table continues on the next page...

PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_9030	Pin Control Register n (PORTA_PCR12)	32	R/W	See section	12.5.1/246
4004_9034	Pin Control Register n (PORTA_PCR13)	32	R/W	See section	12.5.1/246
4004_9038	Pin Control Register n (PORTA_PCR14)	32	R/W	See section	12.5.1/246
4004_903C	Pin Control Register n (PORTA_PCR15)	32	R/W	See section	12.5.1/246
4004_9040	Pin Control Register n (PORTA_PCR16)	32	R/W	See section	12.5.1/246
4004_9044	Pin Control Register n (PORTA_PCR17)	32	R/W	See section	12.5.1/246
4004_9048	Pin Control Register n (PORTA_PCR18)	32	R/W	See section	12.5.1/246
4004_904C	Pin Control Register n (PORTA_PCR19)	32	R/W	See section	12.5.1/246
4004_9050	Pin Control Register n (PORTA_PCR20)	32	R/W	See section	12.5.1/246
4004_9054	Pin Control Register n (PORTA_PCR21)	32	R/W	See section	12.5.1/246
4004_9058	Pin Control Register n (PORTA_PCR22)	32	R/W	See section	12.5.1/246
4004_905C	Pin Control Register n (PORTA_PCR23)	32	R/W	See section	12.5.1/246
4004_9060	Pin Control Register n (PORTA_PCR24)	32	R/W	See section	12.5.1/246
4004_9064	Pin Control Register n (PORTA_PCR25)	32	R/W	See section	12.5.1/246
4004_9068	Pin Control Register n (PORTA_PCR26)	32	R/W	See section	12.5.1/246
4004_906C	Pin Control Register n (PORTA_PCR27)	32	R/W	See section	12.5.1/246
4004_9070	Pin Control Register n (PORTA_PCR28)	32	R/W	See section	12.5.1/246
4004_9074	Pin Control Register n (PORTA_PCR29)	32	R/W	See section	12.5.1/246
4004_9078	Pin Control Register n (PORTA_PCR30)	32	R/W	See section	12.5.1/246
4004_907C	Pin Control Register n (PORTA_PCR31)	32	R/W	See section	12.5.1/246
4004_9080	Global Pin Control Low Register (PORTA_GPCLR)	32	W (always reads 0)	0000_0000h	12.5.2/249
4004_9084	Global Pin Control High Register (PORTA_GPCHR)	32	W (always reads 0)	0000_0000h	12.5.3/249
4004_9088	Global Interrupt Control Low Register (PORTA_GICLR)	32	W (always reads 0)	0000_0000h	12.5.4/250
4004_908C	Global Interrupt Control High Register (PORTA_GICHR)	32	W (always reads 0)	0000_0000h	12.5.5/250
4004_90A0	Interrupt Status Flag Register (PORTA_ISFR)	32	w1c	0000_0000h	12.5.6/251
4004_A000	Pin Control Register n (PORTB_PCR0)	32	R/W	See section	12.5.1/246
4004_A004	Pin Control Register n (PORTB_PCR1)	32	R/W	See section	12.5.1/246
4004_A008	Pin Control Register n (PORTB_PCR2)	32	R/W	See section	12.5.1/246
4004_A00C	Pin Control Register n (PORTB_PCR3)	32	R/W	See section	12.5.1/246
4004_A010	Pin Control Register n (PORTB_PCR4)	32	R/W	See section	12.5.1/246
4004_A014	Pin Control Register n (PORTB_PCR5)	32	R/W	See section	12.5.1/246
4004_A018	Pin Control Register n (PORTB_PCR6)	32	R/W	See section	12.5.1/246

Table continues on the next page...

PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_A01C	Pin Control Register n (PORTB_PCR7)	32	R/W	See section	12.5.1/246
4004_A020	Pin Control Register n (PORTB_PCR8)	32	R/W	See section	12.5.1/246
4004_A024	Pin Control Register n (PORTB_PCR9)	32	R/W	See section	12.5.1/246
4004_A028	Pin Control Register n (PORTB_PCR10)	32	R/W	See section	12.5.1/246
4004_A02C	Pin Control Register n (PORTB_PCR11)	32	R/W	See section	12.5.1/246
4004_A030	Pin Control Register n (PORTB_PCR12)	32	R/W	See section	12.5.1/246
4004_A034	Pin Control Register n (PORTB_PCR13)	32	R/W	See section	12.5.1/246
4004_A038	Pin Control Register n (PORTB_PCR14)	32	R/W	See section	12.5.1/246
4004_A03C	Pin Control Register n (PORTB_PCR15)	32	R/W	See section	12.5.1/246
4004_A040	Pin Control Register n (PORTB_PCR16)	32	R/W	See section	12.5.1/246
4004_A044	Pin Control Register n (PORTB_PCR17)	32	R/W	See section	12.5.1/246
4004_A048	Pin Control Register n (PORTB_PCR18)	32	R/W	See section	12.5.1/246
4004_A04C	Pin Control Register n (PORTB_PCR19)	32	R/W	See section	12.5.1/246
4004_A050	Pin Control Register n (PORTB_PCR20)	32	R/W	See section	12.5.1/246
4004_A054	Pin Control Register n (PORTB_PCR21)	32	R/W	See section	12.5.1/246
4004_A058	Pin Control Register n (PORTB_PCR22)	32	R/W	See section	12.5.1/246
4004_A05C	Pin Control Register n (PORTB_PCR23)	32	R/W	See section	12.5.1/246
4004_A060	Pin Control Register n (PORTB_PCR24)	32	R/W	See section	12.5.1/246
4004_A064	Pin Control Register n (PORTB_PCR25)	32	R/W	See section	12.5.1/246
4004_A068	Pin Control Register n (PORTB_PCR26)	32	R/W	See section	12.5.1/246
4004_A06C	Pin Control Register n (PORTB_PCR27)	32	R/W	See section	12.5.1/246
4004_A070	Pin Control Register n (PORTB_PCR28)	32	R/W	See section	12.5.1/246
4004_A074	Pin Control Register n (PORTB_PCR29)	32	R/W	See section	12.5.1/246
4004_A078	Pin Control Register n (PORTB_PCR30)	32	R/W	See section	12.5.1/246
4004_A07C	Pin Control Register n (PORTB_PCR31)	32	R/W	See section	12.5.1/246
4004_A080	Global Pin Control Low Register (PORTB_GPCLR)	32	W (always reads 0)	0000_0000h	12.5.2/249
4004_A084	Global Pin Control High Register (PORTB_GPCHR)	32	W (always reads 0)	0000_0000h	12.5.3/249
4004_A088	Global Interrupt Control Low Register (PORTB_GICLR)	32	W (always reads 0)	0000_0000h	12.5.4/250
4004_A08C	Global Interrupt Control High Register (PORTB_GICHR)	32	W (always reads 0)	0000_0000h	12.5.5/250
4004_A0A0	Interrupt Status Flag Register (PORTB_ISFR)	32	w1c	0000_0000h	12.5.6/251
4004_B000	Pin Control Register n (PORTC_PCR0)	32	R/W	See section	12.5.1/246
4004_B004	Pin Control Register n (PORTC_PCR1)	32	R/W	See section	12.5.1/246

Table continues on the next page...

PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_B008	Pin Control Register n (PORTC_PCR2)	32	R/W	See section	12.5.1/246
4004_B00C	Pin Control Register n (PORTC_PCR3)	32	R/W	See section	12.5.1/246
4004_B010	Pin Control Register n (PORTC_PCR4)	32	R/W	See section	12.5.1/246
4004_B014	Pin Control Register n (PORTC_PCR5)	32	R/W	See section	12.5.1/246
4004_B018	Pin Control Register n (PORTC_PCR6)	32	R/W	See section	12.5.1/246
4004_B01C	Pin Control Register n (PORTC_PCR7)	32	R/W	See section	12.5.1/246
4004_B020	Pin Control Register n (PORTC_PCR8)	32	R/W	See section	12.5.1/246
4004_B024	Pin Control Register n (PORTC_PCR9)	32	R/W	See section	12.5.1/246
4004_B028	Pin Control Register n (PORTC_PCR10)	32	R/W	See section	12.5.1/246
4004_B02C	Pin Control Register n (PORTC_PCR11)	32	R/W	See section	12.5.1/246
4004_B030	Pin Control Register n (PORTC_PCR12)	32	R/W	See section	12.5.1/246
4004_B034	Pin Control Register n (PORTC_PCR13)	32	R/W	See section	12.5.1/246
4004_B038	Pin Control Register n (PORTC_PCR14)	32	R/W	See section	12.5.1/246
4004_B03C	Pin Control Register n (PORTC_PCR15)	32	R/W	See section	12.5.1/246
4004_B040	Pin Control Register n (PORTC_PCR16)	32	R/W	See section	12.5.1/246
4004_B044	Pin Control Register n (PORTC_PCR17)	32	R/W	See section	12.5.1/246
4004_B048	Pin Control Register n (PORTC_PCR18)	32	R/W	See section	12.5.1/246
4004_B04C	Pin Control Register n (PORTC_PCR19)	32	R/W	See section	12.5.1/246
4004_B050	Pin Control Register n (PORTC_PCR20)	32	R/W	See section	12.5.1/246
4004_B054	Pin Control Register n (PORTC_PCR21)	32	R/W	See section	12.5.1/246
4004_B058	Pin Control Register n (PORTC_PCR22)	32	R/W	See section	12.5.1/246
4004_B05C	Pin Control Register n (PORTC_PCR23)	32	R/W	See section	12.5.1/246
4004_B060	Pin Control Register n (PORTC_PCR24)	32	R/W	See section	12.5.1/246
4004_B064	Pin Control Register n (PORTC_PCR25)	32	R/W	See section	12.5.1/246
4004_B068	Pin Control Register n (PORTC_PCR26)	32	R/W	See section	12.5.1/246
4004_B06C	Pin Control Register n (PORTC_PCR27)	32	R/W	See section	12.5.1/246
4004_B070	Pin Control Register n (PORTC_PCR28)	32	R/W	See section	12.5.1/246
4004_B074	Pin Control Register n (PORTC_PCR29)	32	R/W	See section	12.5.1/246
4004_B078	Pin Control Register n (PORTC_PCR30)	32	R/W	See section	12.5.1/246
4004_B07C	Pin Control Register n (PORTC_PCR31)	32	R/W	See section	12.5.1/246
4004_B080	Global Pin Control Low Register (PORTC_GPCLR)	32	W (always reads 0)	0000_0000h	12.5.2/249
4004_B084	Global Pin Control High Register (PORTC_GPCHR)	32	W (always reads 0)	0000_0000h	12.5.3/249
4004_B088	Global Interrupt Control Low Register (PORTC_GICLR)	32	W (always reads 0)	0000_0000h	12.5.4/250

Table continues on the next page...

PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_B08C	Global Interrupt Control High Register (PORTC_GICHR)	32	W (always reads 0)	0000_0000h	12.5.5/250
4004_B0A0	Interrupt Status Flag Register (PORTC_ISFR)	32	w1c	0000_0000h	12.5.6/251
4004_C000	Pin Control Register n (PORTD_PCR0)	32	R/W	See section	12.5.1/246
4004_C004	Pin Control Register n (PORTD_PCR1)	32	R/W	See section	12.5.1/246
4004_C008	Pin Control Register n (PORTD_PCR2)	32	R/W	See section	12.5.1/246
4004_C00C	Pin Control Register n (PORTD_PCR3)	32	R/W	See section	12.5.1/246
4004_C010	Pin Control Register n (PORTD_PCR4)	32	R/W	See section	12.5.1/246
4004_C014	Pin Control Register n (PORTD_PCR5)	32	R/W	See section	12.5.1/246
4004_C018	Pin Control Register n (PORTD_PCR6)	32	R/W	See section	12.5.1/246
4004_C01C	Pin Control Register n (PORTD_PCR7)	32	R/W	See section	12.5.1/246
4004_C020	Pin Control Register n (PORTD_PCR8)	32	R/W	See section	12.5.1/246
4004_C024	Pin Control Register n (PORTD_PCR9)	32	R/W	See section	12.5.1/246
4004_C028	Pin Control Register n (PORTD_PCR10)	32	R/W	See section	12.5.1/246
4004_C02C	Pin Control Register n (PORTD_PCR11)	32	R/W	See section	12.5.1/246
4004_C030	Pin Control Register n (PORTD_PCR12)	32	R/W	See section	12.5.1/246
4004_C034	Pin Control Register n (PORTD_PCR13)	32	R/W	See section	12.5.1/246
4004_C038	Pin Control Register n (PORTD_PCR14)	32	R/W	See section	12.5.1/246
4004_C03C	Pin Control Register n (PORTD_PCR15)	32	R/W	See section	12.5.1/246
4004_C040	Pin Control Register n (PORTD_PCR16)	32	R/W	See section	12.5.1/246
4004_C044	Pin Control Register n (PORTD_PCR17)	32	R/W	See section	12.5.1/246
4004_C048	Pin Control Register n (PORTD_PCR18)	32	R/W	See section	12.5.1/246
4004_C04C	Pin Control Register n (PORTD_PCR19)	32	R/W	See section	12.5.1/246
4004_C050	Pin Control Register n (PORTD_PCR20)	32	R/W	See section	12.5.1/246
4004_C054	Pin Control Register n (PORTD_PCR21)	32	R/W	See section	12.5.1/246
4004_C058	Pin Control Register n (PORTD_PCR22)	32	R/W	See section	12.5.1/246
4004_C05C	Pin Control Register n (PORTD_PCR23)	32	R/W	See section	12.5.1/246
4004_C060	Pin Control Register n (PORTD_PCR24)	32	R/W	See section	12.5.1/246
4004_C064	Pin Control Register n (PORTD_PCR25)	32	R/W	See section	12.5.1/246
4004_C068	Pin Control Register n (PORTD_PCR26)	32	R/W	See section	12.5.1/246
4004_C06C	Pin Control Register n (PORTD_PCR27)	32	R/W	See section	12.5.1/246
4004_C070	Pin Control Register n (PORTD_PCR28)	32	R/W	See section	12.5.1/246
4004_C074	Pin Control Register n (PORTD_PCR29)	32	R/W	See section	12.5.1/246
4004_C078	Pin Control Register n (PORTD_PCR30)	32	R/W	See section	12.5.1/246
4004_C07C	Pin Control Register n (PORTD_PCR31)	32	R/W	See section	12.5.1/246
4004_C080	Global Pin Control Low Register (PORTD_GPCLR)	32	W (always reads 0)	0000_0000h	12.5.2/249

Table continues on the next page...

PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_C084	Global Pin Control High Register (PORTD_GPCHR)	32	W (always reads 0)	0000_0000h	12.5.3/249
4004_C088	Global Interrupt Control Low Register (PORTD_GICLR)	32	W (always reads 0)	0000_0000h	12.5.4/250
4004_C08C	Global Interrupt Control High Register (PORTD_GICHR)	32	W (always reads 0)	0000_0000h	12.5.5/250
4004_C0A0	Interrupt Status Flag Register (PORTD_ISFR)	32	w1c	0000_0000h	12.5.6/251
4004_D000	Pin Control Register n (PORTE_PCR0)	32	R/W	See section	12.5.1/246
4004_D004	Pin Control Register n (PORTE_PCR1)	32	R/W	See section	12.5.1/246
4004_D008	Pin Control Register n (PORTE_PCR2)	32	R/W	See section	12.5.1/246
4004_D00C	Pin Control Register n (PORTE_PCR3)	32	R/W	See section	12.5.1/246
4004_D010	Pin Control Register n (PORTE_PCR4)	32	R/W	See section	12.5.1/246
4004_D014	Pin Control Register n (PORTE_PCR5)	32	R/W	See section	12.5.1/246
4004_D018	Pin Control Register n (PORTE_PCR6)	32	R/W	See section	12.5.1/246
4004_D01C	Pin Control Register n (PORTE_PCR7)	32	R/W	See section	12.5.1/246
4004_D020	Pin Control Register n (PORTE_PCR8)	32	R/W	See section	12.5.1/246
4004_D024	Pin Control Register n (PORTE_PCR9)	32	R/W	See section	12.5.1/246
4004_D028	Pin Control Register n (PORTE_PCR10)	32	R/W	See section	12.5.1/246
4004_D02C	Pin Control Register n (PORTE_PCR11)	32	R/W	See section	12.5.1/246
4004_D030	Pin Control Register n (PORTE_PCR12)	32	R/W	See section	12.5.1/246
4004_D034	Pin Control Register n (PORTE_PCR13)	32	R/W	See section	12.5.1/246
4004_D038	Pin Control Register n (PORTE_PCR14)	32	R/W	See section	12.5.1/246
4004_D03C	Pin Control Register n (PORTE_PCR15)	32	R/W	See section	12.5.1/246
4004_D040	Pin Control Register n (PORTE_PCR16)	32	R/W	See section	12.5.1/246
4004_D044	Pin Control Register n (PORTE_PCR17)	32	R/W	See section	12.5.1/246
4004_D048	Pin Control Register n (PORTE_PCR18)	32	R/W	See section	12.5.1/246
4004_D04C	Pin Control Register n (PORTE_PCR19)	32	R/W	See section	12.5.1/246
4004_D050	Pin Control Register n (PORTE_PCR20)	32	R/W	See section	12.5.1/246
4004_D054	Pin Control Register n (PORTE_PCR21)	32	R/W	See section	12.5.1/246
4004_D058	Pin Control Register n (PORTE_PCR22)	32	R/W	See section	12.5.1/246
4004_D05C	Pin Control Register n (PORTE_PCR23)	32	R/W	See section	12.5.1/246
4004_D060	Pin Control Register n (PORTE_PCR24)	32	R/W	See section	12.5.1/246
4004_D064	Pin Control Register n (PORTE_PCR25)	32	R/W	See section	12.5.1/246
4004_D068	Pin Control Register n (PORTE_PCR26)	32	R/W	See section	12.5.1/246
4004_D06C	Pin Control Register n (PORTE_PCR27)	32	R/W	See section	12.5.1/246
4004_D070	Pin Control Register n (PORTE_PCR28)	32	R/W	See section	12.5.1/246
4004_D074	Pin Control Register n (PORTE_PCR29)	32	R/W	See section	12.5.1/246

Table continues on the next page...

PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_D078	Pin Control Register n (PORTE_PCR30)	32	R/W	See section	12.5.1/246
4004_D07C	Pin Control Register n (PORTE_PCR31)	32	R/W	See section	12.5.1/246
4004_D080	Global Pin Control Low Register (PORTE_GPCLR)	32	W (always reads 0)	0000_0000h	12.5.2/249
4004_D084	Global Pin Control High Register (PORTE_GPCHR)	32	W (always reads 0)	0000_0000h	12.5.3/249
4004_D088	Global Interrupt Control Low Register (PORTE_GICLR)	32	W (always reads 0)	0000_0000h	12.5.4/250
4004_D08C	Global Interrupt Control High Register (PORTE_GICHR)	32	W (always reads 0)	0000_0000h	12.5.5/250
4004_D0A0	Interrupt Status Flag Register (PORTE_ISFR)	32	w1c	0000_0000h	12.5.6/251

12.5.1 Pin Control Register n (PORTx_PCR*n*)

NOTE

See 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 unbonded 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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0							ISF	0				IRQC			
W								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	LK	0				MUX			0	0	ODE	PFE	0	SRE	PE	PS
W																
Reset	0	0	0	0	0	*	*	*	0	0	0	*	0	*	*	*

* Notes:
- MUX 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 ISF	Interrupt Status Flag The pin interrupt configuration is valid in all digital pin muxing modes. 0 Configured interrupt is not detected. 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.

Table continues on the next page...

PORTx_PCR*n* field descriptions (continued)

Field	Description
	Otherwise, the flag remains set until a logic 1 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–16 IRQC	Interrupt Configuration The pin interrupt configuration is valid in all digital pin muxing modes. The corresponding pin is configured to generate interrupt/DMA request as follows: 0000 Interrupt Status Flag (ISF) is disabled. 0001 ISF flag and DMA request on rising edge. 0010 ISF flag and DMA request on falling edge. 0011 ISF flag and DMA request on either edge. 0100 Reserved. 0101 Flag sets on rising edge. 0110 Flag sets on falling edge. 0111 Flag sets on either edge. 1000 ISF flag and Interrupt when logic 0. 1001 ISF flag and Interrupt on rising-edge. 1010 ISF flag and Interrupt on falling-edge. 1011 ISF flag and Interrupt on either edge. 1100 ISF flag and Interrupt when logic 1. 1101 Enable active high trigger output, flag is disabled. [The trigger output goes to the trigger mux, which allows pins to trigger other peripherals (configurable polarity; 1 pin per port; if multiple pins are configured, then they are ORed together to create the trigger)] 1110 Enable active low trigger output, flag is disabled. 1111 Reserved.
15 LK	Lock Register 0 Pin Control Register fields [15:0] are not locked. 1 Pin Control Register fields [15:0] are locked and cannot be updated until the next system reset.
14–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–8 MUX	Pin Mux Control 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. The corresponding pin is configured in the following pin muxing slot as follows: 000 Pin disabled (Alternative 0) (analog). 001 Alternative 1 (GPIO). 010 Alternative 2 (chip-specific). 011 Alternative 3 (chip-specific). 100 Alternative 4 (chip-specific). 101 Alternative 5 (chip-specific). 110 Alternative 6 (chip-specific). 111 Alternative 7 (chip-specific).

Table continues on the next page...

PORTx_PCR*n* field descriptions (continued)

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 ODE	Open Drain Enable This field is read-only for pins that do not support a configurable open drain output. Open drain configuration is valid in all digital pin muxing modes. 0 Open drain output is disabled on the corresponding pin. 1 Open drain output is enabled on the corresponding pin, if the pin is configured as a digital output.
4 PFE	Passive Filter Enable This field is read-only for pins that do not support a configurable passive input filter. Passive filter configuration is valid in all digital pin muxing modes. 0 Passive input filter is disabled on the corresponding pin. 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 SRE	Slew Rate Enable This field is read-only for pins that do not support a configurable slew rate. Slew rate configuration is valid in all digital pin muxing modes. 0 Fast slew rate is configured on the corresponding pin, if the pin is configured as a digital output. 1 Slow slew rate is configured on the corresponding pin, if the pin is configured as a digital output.
1 PE	Pull Enable This field is read-only for pins that do not support a configurable pull resistor. Refer to the Chapter of Signal Multiplexing and Signal Descriptions for the pins that support a configurable pull resistor. Pull configuration is valid in all digital pin muxing modes. 0 Internal pullup or pulldown resistor is not enabled on the corresponding pin. 1 Internal pullup or pulldown resistor is enabled on the corresponding pin, if the pin is configured as a digital input.
0 PS	Pull Select This bit is read only for pins that do not support a configurable pull resistor direction. Pull configuration is valid in all digital pin muxing modes. 0 Internal pulldown resistor is enabled on the corresponding pin, if the corresponding PE field is set. 1 Internal pullup resistor is enabled on the corresponding pin, if the corresponding PE field is set.

12.5.2 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 GPWE	Global Pin Write Enable 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. 0 Corresponding Pin Control Register is not updated with the value in GPWD. 1 Corresponding Pin Control Register is updated with the value in GPWD.
GPWD	Global Pin Write Data Write value that is written to all Pin Control Registers bits [15:0] that are selected by GPWE.

12.5.3 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 GPWE	Global Pin Write Enable 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. 0 Corresponding Pin Control Register is not updated with the value in GPWD. 1 Corresponding Pin Control Register is updated with the value in GPWD.
GPWD	Global Pin Write Data Write value that is written to all Pin Control Registers bits [15:0] that are selected by GPWE.

12.5.4 Global Interrupt Control Low Register (PORTx_GICLR)

Only 32-bit writes are supported to this register.

Address: Base address + 88h 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	GIWD																GIWE															
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_GICLR field descriptions

Field	Description
31–16 GIWD	Global Interrupt Write Data Write value that is written to all Pin Control Registers bits [31:16] that are selected by GIWE.
GIWE	Global Interrupt Write Enable Selects which Pin Control Registers (15 through 0) bits [31:16] update with the value in GIWD. 0 Corresponding Pin Control Register is not updated with the value in GPWD. 1 Corresponding Pin Control Register is updated with the value in GPWD.

12.5.5 Global Interrupt Control High Register (PORTx_GICHR)

Only 32-bit writes are supported to this register.

Address: Base address + 8Ch 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	GIWD																GIWE															
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_GICHR field descriptions

Field	Description
31–16 GIWD	Global Interrupt Write Data Write value that is written to all Pin Control Registers bits [31:16] that are selected by GIWE.
GIWE	Global Interrupt Write Enable Selects which Pin Control Registers (31 through 16) bits [31:16] update with the value in GIWD. 0 Corresponding Pin Control Register is not updated with the value in GPWD. 1 Corresponding Pin Control Register is updated with the value in GPWD.

12.5.6 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
ISF	Interrupt Status Flag Each bit in the field indicates the detection of the configured interrupt of the same number as the field. 0 Configured interrupt is not detected. 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 1 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.

12.6 Functional description

12.6.1 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 LK bit (bit 15 of Pin Control Register PCR*n*) locks the lower 16 bits of each Pin Control register and blocks any writes to that register until the next system reset.

The lower half of the Pin Control register configures the following functions for each pin within the 32-bit port.

- Pullup or pulldown enable on selected pins
- Open drain enable on selected pins
- Passive input filter enable on selected pins
- 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, output buffer enable, 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 LK bit (bit 15) locks the lower 16 bits of each Pin Control register, and blocks any writes to that Pin Control register until the next system reset.

The configuration of each Pin Control register is retained when the PORT module is disabled.

Whenever a pin is configured in any digital pin muxing mode, the input buffer for that pin is enabled allowing the pin state to be read via the corresponding GPIO Port Data Input Register (GPIO_PDIR) or allowing a pin interrupt or DMA request to be generated. If a pin is ever floating when its input buffer is enabled, then this can cause an increase in power consumption and must be avoided. A pin can be floating due to an input pin that is not connected or an output pin that has tri-stated (output buffer is disabled).

Enabling the internal pull resistor (or implementing an external pull resistor) will ensure a pin does not float when its input buffer is enabled; note that the internal pull resistor is automatically disabled whenever the output buffer is enabled allowing the Pull Enable bit to remain set. Configuring the Pin Muxing mode to disabled or analog will disable the pin's input buffer and results in the lowest power consumption.

12.6.2 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 16 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 0.

12.6.3 Global interrupt control

The two global interrupt control registers allow a single register write to update the upper half of the pin control register on up to 16 pins, all with the same value.

The global interrupt control registers are designed to enable software to quickly configure multiple pins within the one port for the same interrupt configuration. However, the pin control functions cannot be configured using the global interrupt control registers.

The global interrupt control registers are write-only registers and always read as 0.

12.6.4 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
- Rising edge flag (for software polling)
- Falling edge flag (for software polling)
- Rising and falling edge flag (for software polling)
- Active high level peripheral trigger (status flag disabled)
- Active low level peripheral trigger (status flag disabled)
- 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 . 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 wake-up signal to exit the Low-Power mode.

The PORT module generates a single peripheral trigger output that asserts if any pin configured for active high trigger is logic one, or any pin triggered for active low trigger is logic zero. The peripheral trigger output asynchronously updates from the value on the configured pins.

Chapter 13
System Integration Module (SIM)

13.1 Introduction

The system integration module (SIM) provides system control and chip configuration registers.

13.1.1 Features

- Flash and System RAM size configuration
- USB regulator configuration

13.2 Memory map and register definition

The SIM module contains many bit fields for miscellaneous configuration of the device.

NOTE

The SIM registers can be written only in supervisor mode. In user mode, write accesses are blocked and will result in a bus error.

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	13.2.1/257
4004_8004	System Options Register 2 (SIM_SOPT2)	32	R/W	0000_0000h	13.2.2/258
4004_8010	System Options Register 5 (SIM_SOPT5)	32	R/W	0000_0000h	13.2.3/260
4004_8018	System Options Register 7 (SIM_SOPT7)	32	R/W	0000_0000h	13.2.4/261
4004_8020	System Options Register 9 (SIM_SOPT9)	32	R/W	0000_0000h	13.2.5/263

Table continues on the next page...

SIM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_8024	System Device Identification Register (SIM_SDID)	32	R	Undefined	13.2.6/264
4004_8034	System Clock Gating Control Register 4 (SIM_SCGC4)	32	R/W	F010_0030h	13.2.7/266
4004_8038	System Clock Gating Control Register 5 (SIM_SCGC5)	32	R/W	0004_0182h	13.2.8/268
4004_803C	System Clock Gating Control Register 6 (SIM_SCGC6)	32	R/W	0000_0001h	13.2.9/271
4004_8040	System Clock Gating Control Register 7 (SIM_SCGC7)	32	R/W	0000_0006h	13.2.10/273
4004_8044	System Clock Divider Register 1 (SIM_CLKDIV1)	32	R/W	0001_0000h	13.2.11/274
4004_8048	System Clock Divider Register 2 (SIM_CLKDIV2)	32	R/W	0000_0000h	13.2.12/277
4004_804C	Flash Configuration Register 1 (SIM_FCFG1)	32	R	See section	13.2.13/278
4004_8050	Flash Configuration Register 2 (SIM_FCFG2)	32	R	See section	13.2.14/279
4004_8054	Unique Identification Register High (SIM_UIDH)	32	R	See section	13.2.15/280
4004_8058	Unique Identification Register Mid-High (SIM_UIDMH)	32	R	See section	13.2.16/280
4004_805C	Unique Identification Register Mid Low (SIM_UIDML)	32	R	See section	13.2.17/281
4004_8060	Unique Identification Register Low (SIM_UIDL)	32	R	See section	13.2.18/281
4004_8064	System Clock Divider Register 3 (SIM_CLKDIV3)	32	R/W	0000_0000h	13.2.19/282
4004_806C	Misc Control Register (SIM_MISCCTRL)	32	R/W	0001_0000h	13.2.20/283
4004_8090	Secure Key Register 0 (SIM_SECKEY0)	32	R	Undefined	13.2.21/284
4004_8094	Secure Key Register 1 (SIM_SECKEY1)	32	R	Undefined	13.2.22/285
4004_8098	Secure Key Register 2 (SIM_SECKEY2)	32	R	Undefined	13.2.23/285
4004_809C	Secure Key Register 3 (SIM_SECKEY3)	32	R	Undefined	13.2.24/286

13.2.1 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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												OSC32KSEL		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	RAMSIZE				0											
W																
Reset	*	*	*	*	0	0	0	0	0	0	0	0	0	0	0	0

* Notes:
- RAMSIZE field: Device specific value.

SIM_SOPT1 field descriptions

Field	Description
31–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–18 OSC32KSEL	32K Oscillator Clock Select Selects the 32 kHz clock source (ERCLK32K) for RTC and LPTMR. This field is reset only on POR/LVD. 00 System oscillator (OSC32KCLK) 01 Reserved 10 RTC oscillator 11 LPO 1 kHz
17–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–12 RAMSIZE	System RAM Size Specifies the size of the System RAM 0001 8 KB 0011 16 KB 0100 24 KB 0101 32 KB 0110 48 KB 0111 64 KB 1000 96 KB

Table continues on the next page...

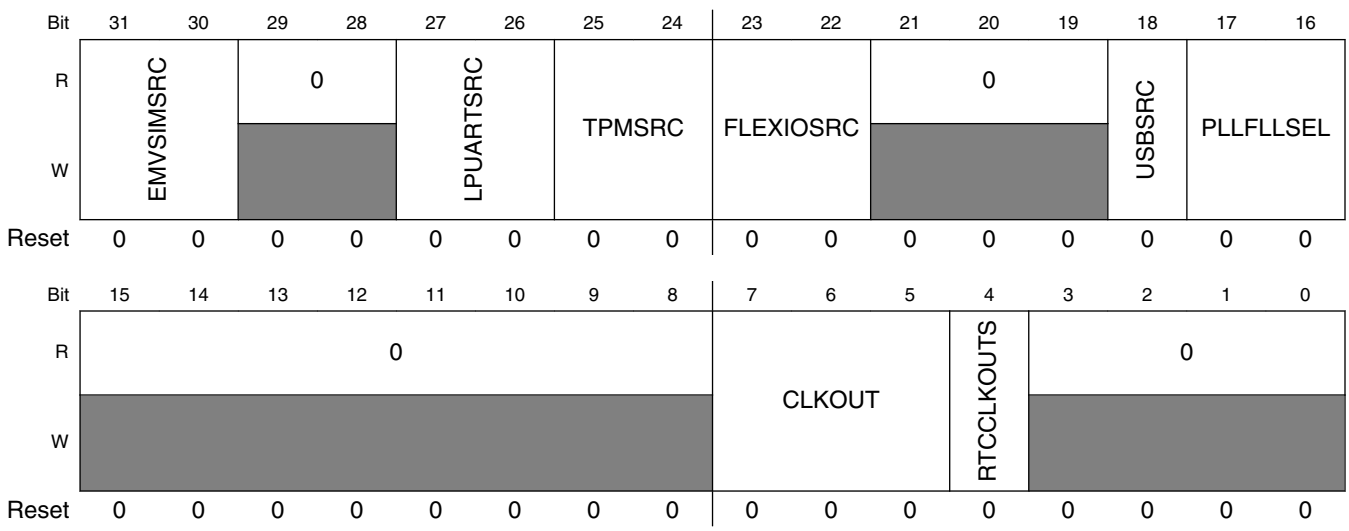
SIM_SOPT1 field descriptions (continued)

Field	Description
	1001 128 KB 1011 256 KB
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

13.2.2 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



SIM_SOPT2 field descriptions

Field	Description
31–30 EMVSIMSRC	EMVSIM Module Clock Source Select Selects the clock source for the EMVSIM transmit and receive clock. 00 Clock disabled 01 MCGFLLCLK ,MCGPLLCLK, or IRC48M clock as selected by SOPT2[PLLFLLSEL], and then divided by the PLLFLLCLK fractional divider as configured by SIM_CLKDIV3[PLLFLLFRAC] and SIM_CLKDIV3[PLLFLLDIV]. 10 OSCERCLK clock 11 MCGIRCLK clock
29–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIM_SOPT2 field descriptions (continued)

Field	Description
27–26 LPUARTSRC	LPUART clock source select Selects the clock source for the LPUART transmit and receive clock. 00 Clock disabled 01 MCGFLLCLK ,MCGPLLCLK, or IRC48M clock as selected by SOPT2[PLLFLLSEL], and then divided by the PLLFLLCLK fractional divider as configured by SIM_CLKDIV3[PLLFLLFRAC] and SIM_CLKDIV3[PLLFLLDIV]. 10 OSCERCLK clock 11 MCGIRCLK clock
25–24 TPMSRC	TPM clock source select Selects the clock source for the TPM counter clock 00 Clock disabled 01 MCGFLLCLK, or MCGPLLCLK, or IRC48M clock as selected by SOPT2[PLLFLLSEL]. 10 OSCERCLK clock 11 MCGIRCLK clock
23–22 FLEXIOSRC	FlexIO Module Clock Source Select Selects the clock source for the FlexIO transmit and receive clock. 00 System clock 01 MCGFLLCLK, or MCGPLLCLK, or IRC48M clock as selected by SOPT2[PLLFLLSEL]. 10 OSCERCLK clock 11 MCGIRCLK clock
21–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 USBSRC	USB clock source select Selects the clock source for the USB 48 MHz clock. 0 External bypass clock (USB_CLKIN). 1 MCGFLLCLK, or MCGPLLCLK, or IRC48M clock as selected by SOPT2[PLLFLLSEL], and then divided by the USB fractional divider as configured by SIM_CLKDIV2[USBFRAC, USBDIV].
17–16 PLLFLLSEL	PLL/FLL clock select Selects the high frequency clock for various peripheral clocking options. 00 MCGFLLCLK clock 01 MCGPLLCLK clock 10 Reserved 11 IRC48 MHz clock
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–5 CLKOUT	CLKOUT select Selects the clock to output on the CLKOUT pin. 000 Reserved 001 Reserved

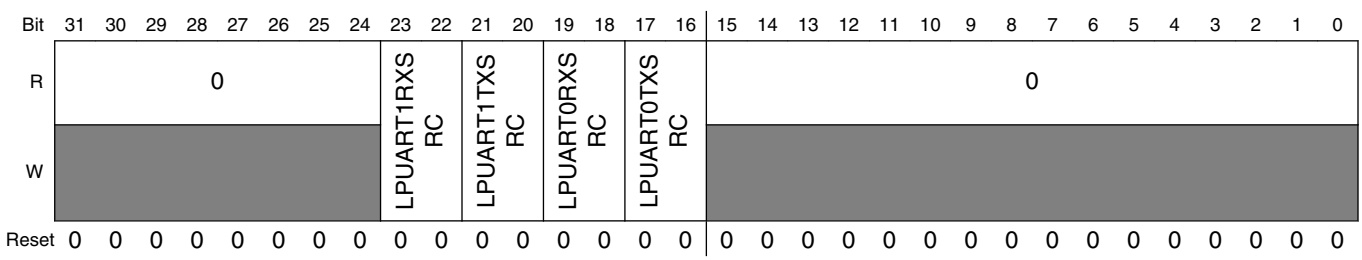
Table continues on the next page...

SIM_SOPT2 field descriptions (continued)

Field	Description
	010 Flash clock 011 LPO clock (1 kHz) 100 MCGIRCLK 101 RTC 32.768kHz clock 110 OSCERCLK0 111 IRC 48 MHz clock
4 RTCCLKOUTS	RTC clock out select Selects either the RTC 1 Hz clock or the 32.768kHz clock to be output on the RTC_CLKOUT pin. 0 RTC 1 Hz clock is output on the RTC_CLKOUT pin. 1 RTC 32.768kHz clock is output on the RTC_CLKOUT pin.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

13.2.3 System Options Register 5 (SIM_SOPT5)

Address: 4004_7000h base + 1010h offset = 4004_8010h



SIM_SOPT5 field descriptions

Field	Description
31–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–22 LPUART1RXSRC	LPUART1 receive data source select Selects the source for the LPUART1 receive data. 00 LPUART1_RX pin 01 CMP0 10 Reserved 11 Reserved
21–20 LPUART1TXSRC	LPUART1 transmit data source select Selects the source for the LPUART1 transmit data. 00 LPUART1_TX pin 01 LPUART1_TX pin modulated with TPM1 channel 0 output

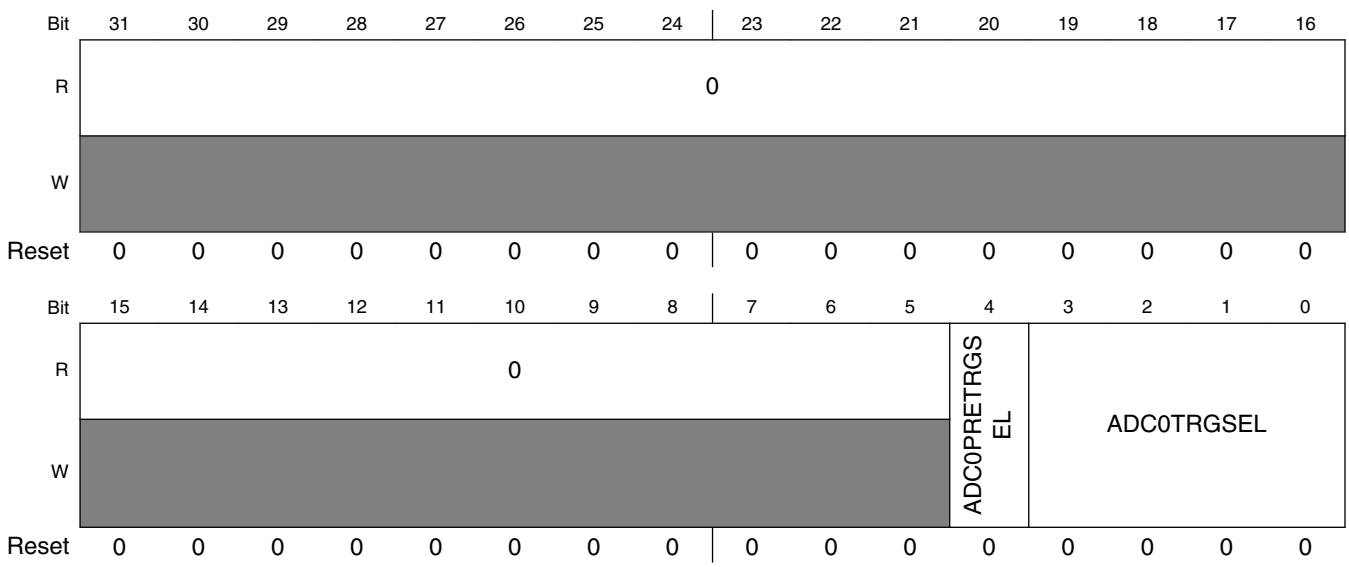
Table continues on the next page...

SIM_SOPT5 field descriptions (continued)

Field	Description
	10 LPUART1_TX pin modulated with TPM2 channel 0 output 11 Reserved
19–18 LPUART0RXSRC	LPUART 0 receive data source select Selects the source for the LPUART0 receive data. 00 LPUART0_RX pin 01 CMP0 10 Reserved 11 Reserved
17–16 LPUART0TXSRC	LPUART0 transmit data source select Selects the source for the LPUART0 transmit data. 00 LPUART0_TX pin 01 LPUART0_TX pin modulated with TPM1 channel 0 output 10 LPUART0_TX pin modulated with TPM2 channel 0 output 11 Reserved
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

13.2.4 System Options Register 7 (SIM_SOPT7)

Address: 4004_7000h base + 1018h offset = 4004_8018h

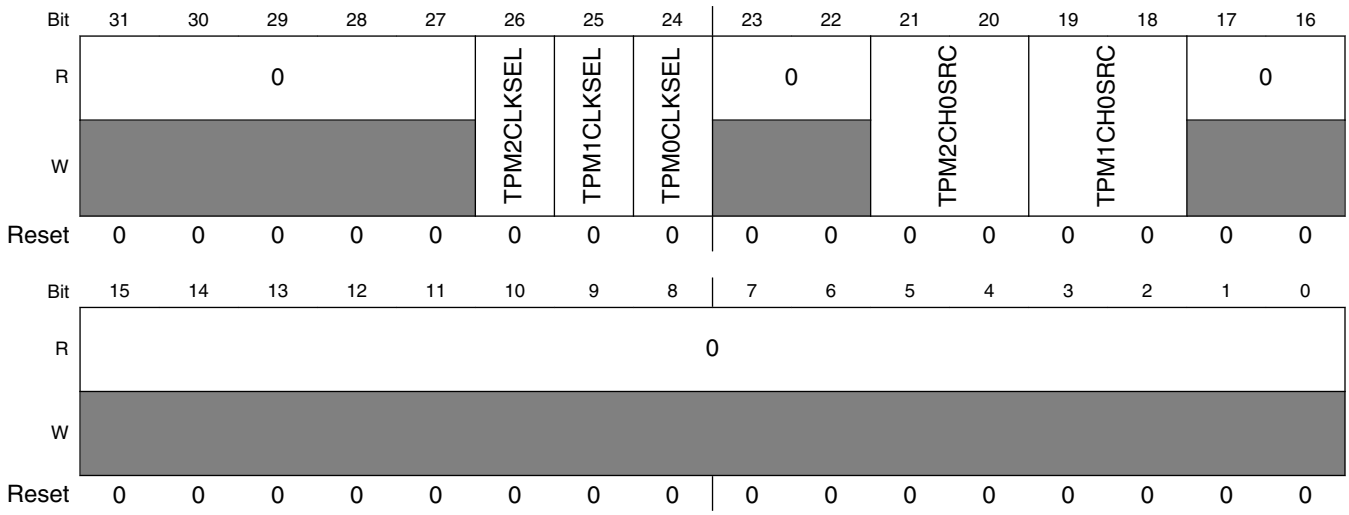


SIM_SOPT7 field descriptions

Field	Description
31–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 ADC0PRETRGSEL	ADC0 pretrigger select Selects the ADC0 pre-trigger source when alternative triggers are enabled through ADC0ALTTRGEN. This field is not used when the TPM trigger source is selected. 0 Pre-trigger A 1 Pre-trigger B
ADC0TRGSEL	ADC0 trigger select Selects the ADC0 trigger source when alternative triggers are functional in stop and VLPS modes. 0000 External trigger pin input (EXTRG) 0001 High speed comparator 0 output 0010 Reserved 0011 Reserved 0100 PIT trigger 0 0101 PIT trigger 1 0110 PIT trigger 2 0111 PIT trigger 3 1000 TPM0 trigger 1001 TPM1 trigger 1010 TPM2 trigger 1011 Low-power timer1 (LPTMR1) trigger 1100 RTC alarm 1101 RTC seconds 1110 Low-power timer (LPTMR) trigger 1111 TPM1 channel 0 (A pretrigger) and channel 1 (B pretrigger)

13.2.5 System Options Register 9 (SIM_SOPT9)

Address: 4004_7000h base + 1020h offset = 4004_8020h



SIM_SOPT9 field descriptions

Field	Description
31–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 TPM2CLKSEL	TPM2 External Clock Pin Select Selects the external pin used to drive the clock to the TPM2 module. **NOTE:** The selected pin must also be configured for the TPM external clock function through the appropriate pin control register in the port control module. 0 TPM_CLKIN0 pin 1 TPM_CLKIN1 pin
25 TPM1CLKSEL	TPM1 External Clock Pin Select Selects the external pin used to drive the clock to the TPM1 module. **NOTE:** The selected pin must also be configured for the TPM external clock function through the appropriate pin control register in the port control module. 0 TPM_CLKIN0 pin 1 TPM_CLKIN1 pin
24 TPM0CLKSEL	TPM0 External Clock Pin Select Selects the external pin used to drive the clock to the TPM0 module. **NOTE:** The selected pin must also be configured for the TPM external clock function through the appropriate pin control register in the port control module. 0 TPM_CLKIN0 pin 1 TPM_CLKIN1 pin

Table continues on the next page...

SIM_SOPT9 field descriptions (continued)

Field	Description
23–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–20 TPM2CH0SRC	TPM2 channel 0 input capture source select Selects the source for TPM2 channel 0 input capture. **NOTE:** When the TPM is not in input capture mode, clear this field. 00 TPM2_CH0 signal 01 CMP0 output 10 Reserved 11 Reserved
19–18 TPM1CH0SRC	TPM1 channel 0 input capture source select Selects the source for TPM1 channel 0 input capture. **NOTE:** When the TPM is not in input capture mode, clear this field. 00 TPM1_CH0 signal 01 CMP0 output 10 Reserved 11 Reserved
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

13.2.6 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	FAMILYID				SUBFAMID				0								REVID				DIEID					FAMID			PINID			
W																																
Reset	*	*	*	*	*	*	*	*	x*	x*	x*	x*	x*	x*	x*	x*	*	*	*	*	x*	x*	x*	x*	x*	*	*	*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.
- Device specific value.x = Undefined at reset.
- Device specific value.x = Undefined at reset.
- Device specific value.x = Undefined at reset.
- Device specific value.x = Undefined at reset.
- x = Undefined at reset.

SIM_SDID field descriptions

Field	Description
31–28 FAMILYID	Kinetis L family ID Specifies the Kinetis L family of the device.

Table continues on the next page...

SIM_SDID field descriptions (continued)

Field	Description
	0000 KL0x Family 0001 KL1x Family 0010 KL2x Family 0011 KL3x Family) 0100 KL4x Family) 0110 KL6x Family 0111 KL7x Family 1001 KL8x Family others Reserved
27–24 SUBFAMID	Kinetis Sub-Family ID Specifies the Kinetis sub-family of the device. 0000 KLx0 Subfamily 0001 KLx1 Subfamily 0010 KLx2 Subfamily 0011 KLx3 Subfamily 0100 KLx4 Subfamily 0101 KLx5 Subfamily 0110 KLx6 Subfamily 0111 KLx7 Subfamily 1000 KLx8 Subfamily 1001 KLx9 Subfamily others Reserved
23–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–12 REVID	Device Revision Number Specifies the silicon implementation number for the device.
11–7 DIEID	Device die number Specifies the silicon implementation number for the device.
6–4 FAMID	Kinetis family ID Specifies the Kinetis family of the device.This field is for compatibility only, but has been superceded by the SERIESID, FAMILYID and SUBFAMID fields in this register.
PINID	Pincount identification Specifies the pincount of the device. 0000 Reserved 0001 Reserved 0010 Reserved 0011 Reserved 0100 Reserved 0101 64-pin 0110 80-pin 0111 Reserved 1000 100-pin

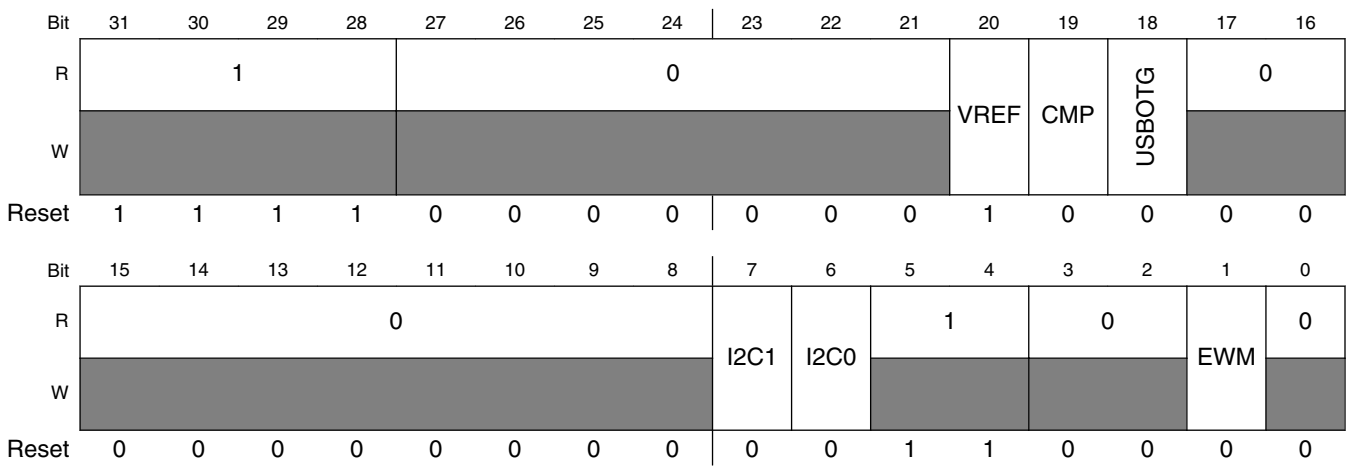
Table continues on the next page...

SIM_SDID field descriptions (continued)

Field	Description
	1001 121-pin 1010 Reserved 1011 Custom pinout (WLCSP) 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

13.2.7 System Clock Gating Control Register 4 (SIM_SCGC4)

Address: 4004_7000h base + 1034h offset = 4004_8034h



SIM_SCGC4 field descriptions

Field	Description
31–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
27–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 VREF	VREF Clock Gate Control Controls the clock gate to the EWM module. 0 Clock disabled 1 Clock enabled
19 CMP	CMP Clock Gate Control Controls the clock gate to the CMP module. 0 Clock disabled 1 Clock enabled

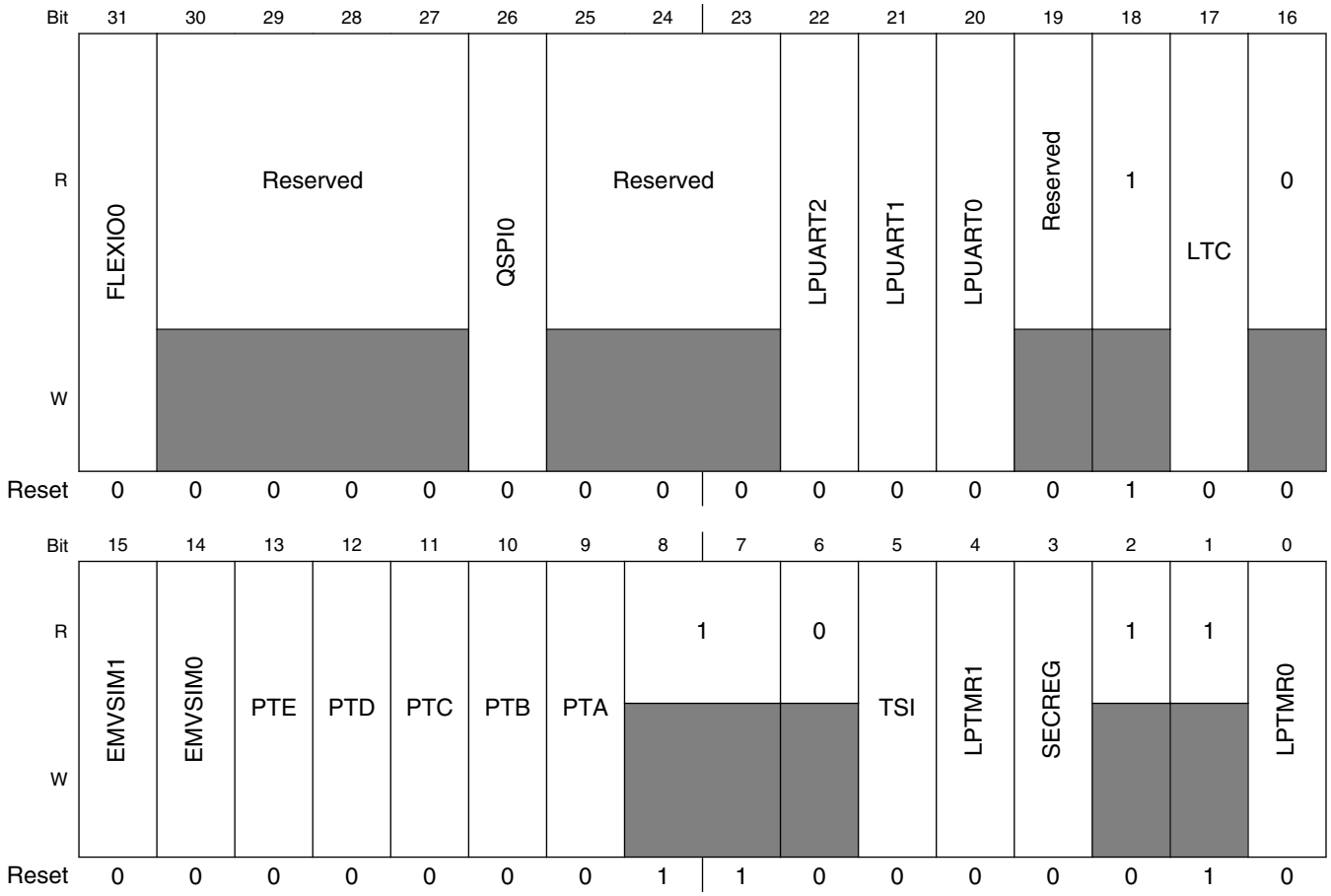
Table continues on the next page...

SIM_SCGC4 field descriptions (continued)

Field	Description
18 USBOTG	USB_OTG Clock Gate Control Controls the clock gate to the USB_OTG module. 0 Clock disabled 1 Clock enabled
17–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 I2C1	I2C1 Clock Gate Control This bit controls the clock gate to the I2C1 module. 0 Clock disabled 1 Clock enabled
6 I2C0	I2C0 Clock Gate Control This bit controls the clock gate to the I2C0 module. 0 Clock disabled 1 Clock enabled
5–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 EWM	EWM Clock Gate Control Controls the clock gate to the EWM module. 0 Clock disabled 1 Clock enabled
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

13.2.8 System Clock Gating Control Register 5 (SIM_SCGC5)

Address: 4004_7000h base + 1038h offset = 4004_8038h



SIM_SCGC5 field descriptions

Field	Description
31 FLEXIO0	FLEXIO0 Clock Gate Control Controls the clock gate to the FLEXIO0 module. 0 Clock disabled 1 Clock enabled
30–27 Reserved	This field is reserved.
26 QSPI0	QSPI0 Clock Gate Control Controls the clock gate to the QSPI0 module. 0 Clock disabled 1 Clock enabled

Table continues on the next page...

SIM_SCGC5 field descriptions (continued)

Field	Description
25–23 Reserved	This field is reserved.
22 LPUART2	LPUART2 Clock Gate Control This bit controls the clock gate to the LPUART2 module. 0 Clock disabled 1 Clock enabled
21 LPUART1	LPUART1 Clock Gate Control This bit controls the clock gate to the LPUART1 module. 0 Clock disabled 1 Clock enabled
20 LPUART0	LPUART0 Clock Gate Control This bit controls the clock gate to the LPUART0 module. 0 Clock disabled 1 Clock enabled
19 Reserved	This field is reserved.
18 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
17 LTC	LTC Clock Gate Control Controls the clock gate to the LTC module. 0 Clock disabled 1 Clock enabled
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 EMVSIM1	EMVSIM1 Clock Gate Control Controls the clock gate to the EMVSIM1 module. 0 Clock disabled 1 Clock enabled
14 EMVSIM0	EMVSIM0 Clock Gate Control Controls the clock gate to the EMVSIM0 module. 0 Clock disabled 1 Clock enabled
13 PTE	PTE Clock Gate Control Controls the clock gate to the PTE module. 0 Clock disabled 1 Clock enabled

Table continues on the next page...

SIM_SCGC5 field descriptions (continued)

Field	Description
12 PTD	PTD Clock Gate Control Controls the clock gate to the PTD module. 0 Clock disabled 1 Clock enabled
11 PTC	PTC Clock Gate Control Controls the clock gate to the PTC module. 0 Clock disabled 1 Clock enabled
10 PTB	PTB Clock Gate Control Controls the clock gate to the PTB module. 0 Clock disabled 1 Clock enabled
9 PTA	PTA Clock Gate Control Controls the clock gate to the PTA module. 0 Clock disabled 1 Clock enabled
8–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 TSI	TSI Access Control Controls software access to the TSI module. 0 Access disabled 1 Access enabled
4 LPTMR1	LPTMR1 Clock Gate Control Controls the clock gate to the LPTMR1 module. 0 Clock disabled 1 Clock enabled
3 SECREG	SECREG Clock Gate Control Controls the clock gate to the SECREG module. 0 Clock disabled 1 Clock enabled
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.

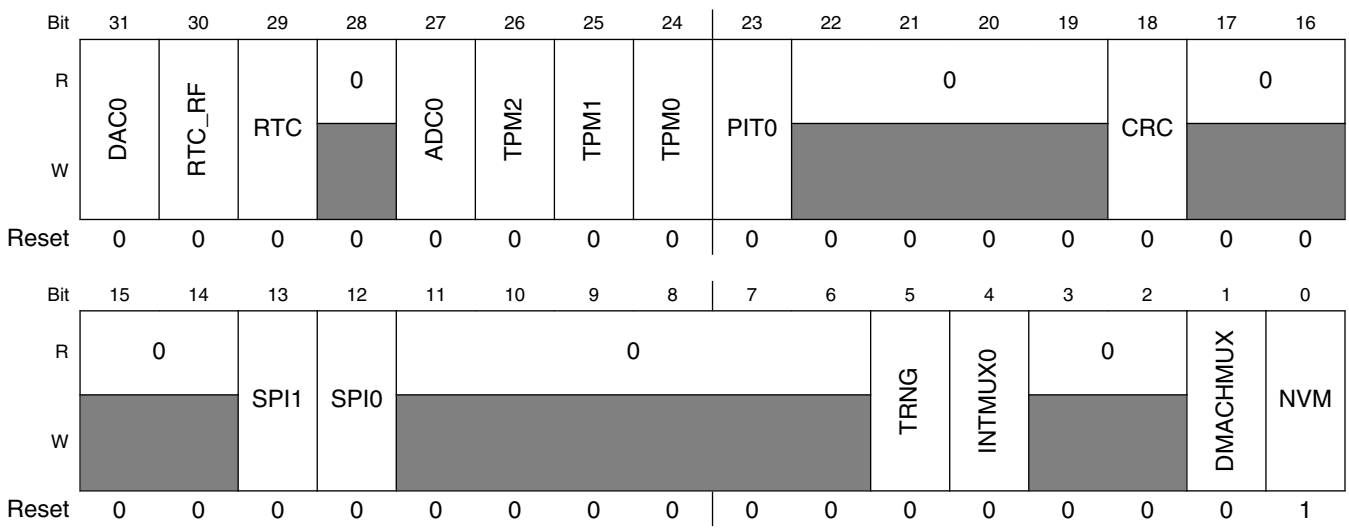
Table continues on the next page...

SIM_SCGC5 field descriptions (continued)

Field	Description
0 LPTMR0	LPTMR0 Clock Gate Control Controls the clock gate to the LPTMR0 module. 0 Clock disabled 1 Clock enabled

13.2.9 System Clock Gating Control Register 6 (SIM_SCGC6)

Address: 4004_7000h base + 103Ch offset = 4004_803Ch



SIM_SCGC6 field descriptions

Field	Description
31 DAC0	DAC0 Clock Gate Control Controls the clock gate to the DAC0 module. 0 Clock disabled 1 Clock enabled
30 RTC_RF	RTC_RF Clock Gate Control Controls the clock gate to the VBAT register file module. 0 Clock disabled 1 Clock enabled
29 RTC	RTC Clock Gate Control Controls the clock gate to the RTC module. 0 Clock disabled 1 Clock enabled

Table continues on the next page...

SIM_SCGC6 field descriptions (continued)

Field	Description
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 ADC0	ADC0 Clock Gate Control Controls the clock gate to the ADC0 module. 0 Clock disabled 1 Clock enabled
26 TPM2	TPM2 Clock Gate Control Controls the clock gate to the TPM2 module. 0 Clock disabled 1 Clock enabled
25 TPM1	TPM1 Clock Gate Control Controls the clock gate to the TPM1 module. 0 Clock disabled 1 Clock enabled
24 TPM0	TPM0 Clock Gate Control Controls the clock gate to the TPM0 module. 0 Clock disabled 1 Clock enabled
23 PIT0	PIT0 Clock Gate Control This bit controls the clock gate to the PIT0 module. 0 Clock disabled 1 Clock enabled
22–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 CRC	CRC Clock Gate Control Controls the clock gate to the CRC module. 0 Clock disabled 1 Clock enabled
17–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SPI1	SPI1 Clock Gate Control Controls the clock gate to the SPI1 module. 0 Clock disabled 1 Clock enabled
12 SPI0	SPI0 Clock Gate Control Controls the clock gate to the SPI0 module.

Table continues on the next page...

SIM_SCGC6 field descriptions (continued)

Field	Description
	0 Clock disabled 1 Clock enabled
11–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 TRNG	TRNG Clock Gate Control Controls the clock gate to the TRNG module. 0 Clock disabled 1 Clock enabled
4 INTMUX0	INTMUX0 Clock Gate Control Controls the clock gate to the INTMUX0 module. 0 Clock disabled 1 Clock enabled
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DMACHMUX	DMACHMUX Clock Gate Control Controls the clock gate to the DMACHMUX module. 0 Clock disabled 1 Clock enabled
0 NVM	NVM Clock Gate Control Controls the clock gate to the NVM module. 0 Clock disabled 1 Clock enabled

13.2.10 System Clock Gating Control Register 7 (SIM_SCGC7)

Address: 4004_7000h base + 1040h offset = 4004_8040h

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													MPU	DMA	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

SIM_SCGC7 field descriptions

Field	Description
31–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 MPU	MPU Clock Gate Control Controls the clock gate to the MPU module. 0 Clock disabled 1 Clock enabled
1 DMA	DMA Clock Gate Control Controls the clock gate to the DMA module. 0 Clock disabled 1 Clock enabled
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

13.2.11 System Clock Divider Register 1 (SIM_CLKDIV1)

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				0				OUTDIV4				OUTDIV5				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

SIM_CLKDIV1 field descriptions

Field	Description
31–28 OUTDIV1	Clock 1 output divider value 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 FTFA_FOPT[LPBOOT]. 0000 Divide-by-1. 0001 Divide-by-2. 0010 Divide-by-3. 0011 Divide-by-4. 0100 Divide-by-5. 0101 Divide-by-6. 0110 Divide-by-7. 0111 Divide-by-8.

Table continues on the next page...

SIM_CLKDIV1 field descriptions (continued)

Field	Description
	1000 Divide-by-9. 1001 Divide-by-10. 1010 Divide-by-11. 1011 Divide-by-12. 1100 Divide-by-13. 1101 Divide-by-14. 1110 Divide-by-15. 1111 Divide-by-16.
27–24 OUTDIV2	Clock 2 output divider value 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 FTFA_FOPT[LPBOOT]. The bus clock frequency must be an integer divide of the core/system clock frequency. 0000 Divide-by-1. 0001 Divide-by-2. 0010 Divide-by-3. 0011 Divide-by-4. 0100 Divide-by-5. 0101 Divide-by-6. 0110 Divide-by-7. 0111 Divide-by-8. 1000 Divide-by-9. 1001 Divide-by-10. 1010 Divide-by-11. 1011 Divide-by-12. 1100 Divide-by-13. 1101 Divide-by-14. 1110 Divide-by-15. 1111 Divide-by-16.
23–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–16 OUTDIV4	Clock 4 output divider value 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 FTFA_FOPT[LPBOOT]. The flash clock frequency must be an integer divide of the system clock frequency. 0000 Divide-by-1. 0001 Divide-by-2. 0010 Divide-by-3. 0011 Divide-by-4. 0100 Divide-by-5. 0101 Divide-by-6. 0110 Divide-by-7. 0111 Divide-by-8. 1000 Divide-by-9. 1001 Divide-by-10.

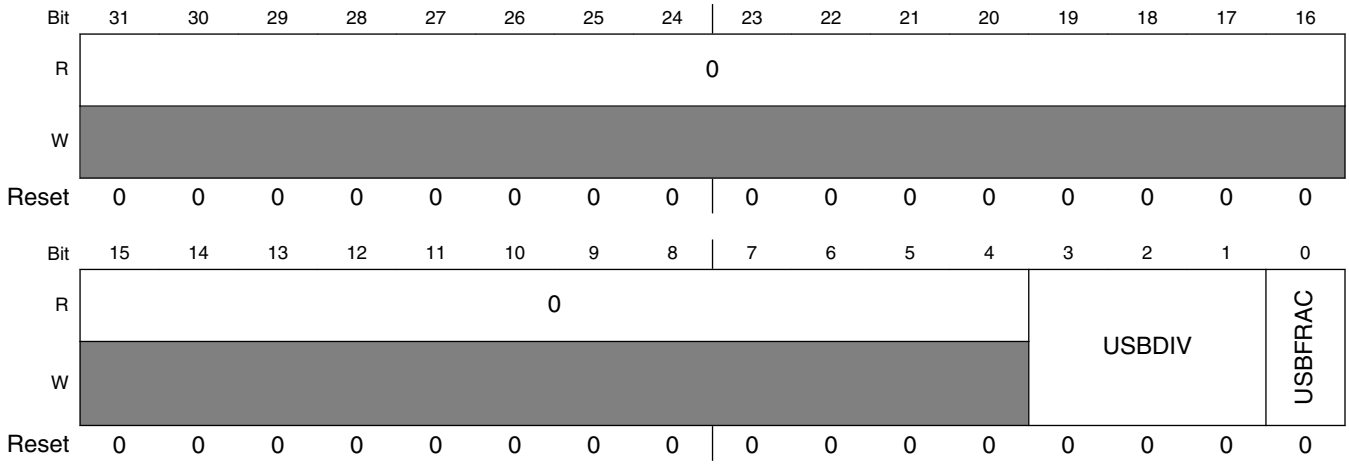
Table continues on the next page...

SIM_CLKDIV1 field descriptions (continued)

Field	Description
	1010 Divide-by-11. 1011 Divide-by-12. 1100 Divide-by-13. 1101 Divide-by-14. 1110 Divide-by-15. 1111 Divide-by-16.
15–12 OUTDIV5	Clock 5 output divider value This field sets the divide value for the fast bus clock from MCGOUTCLK. At the end of reset, it is loaded with either 0001 or 1111 depending on FTFA_FOPT[LPBOOT]. The flash clock frequency must be an integer divide of the system clock frequency. **NOTE:** Make sure (OUTDIV1 + 1) : (OUTDIV5 + 1) = 1 : 1 or 1 : 2, other configurations may result in unexpected failure. 0000 Divide-by-1. 0001 Divide-by-2. 0010 Divide-by-3. 0011 Divide-by-4. 0100 Divide-by-5. 0101 Divide-by-6. 0110 Divide-by-7. 0111 Divide-by-8. 1000 Divide-by-9. 1001 Divide-by-10. 1010 Divide-by-11. 1011 Divide-by-12. 1100 Divide-by-13. 1101 Divide-by-14. 1110 Divide-by-15. 1111 Divide-by-16.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

13.2.12 System Clock Divider Register 2 (SIM_CLKDIV2)

Address: 4004_7000h base + 1048h offset = 4004_8048h

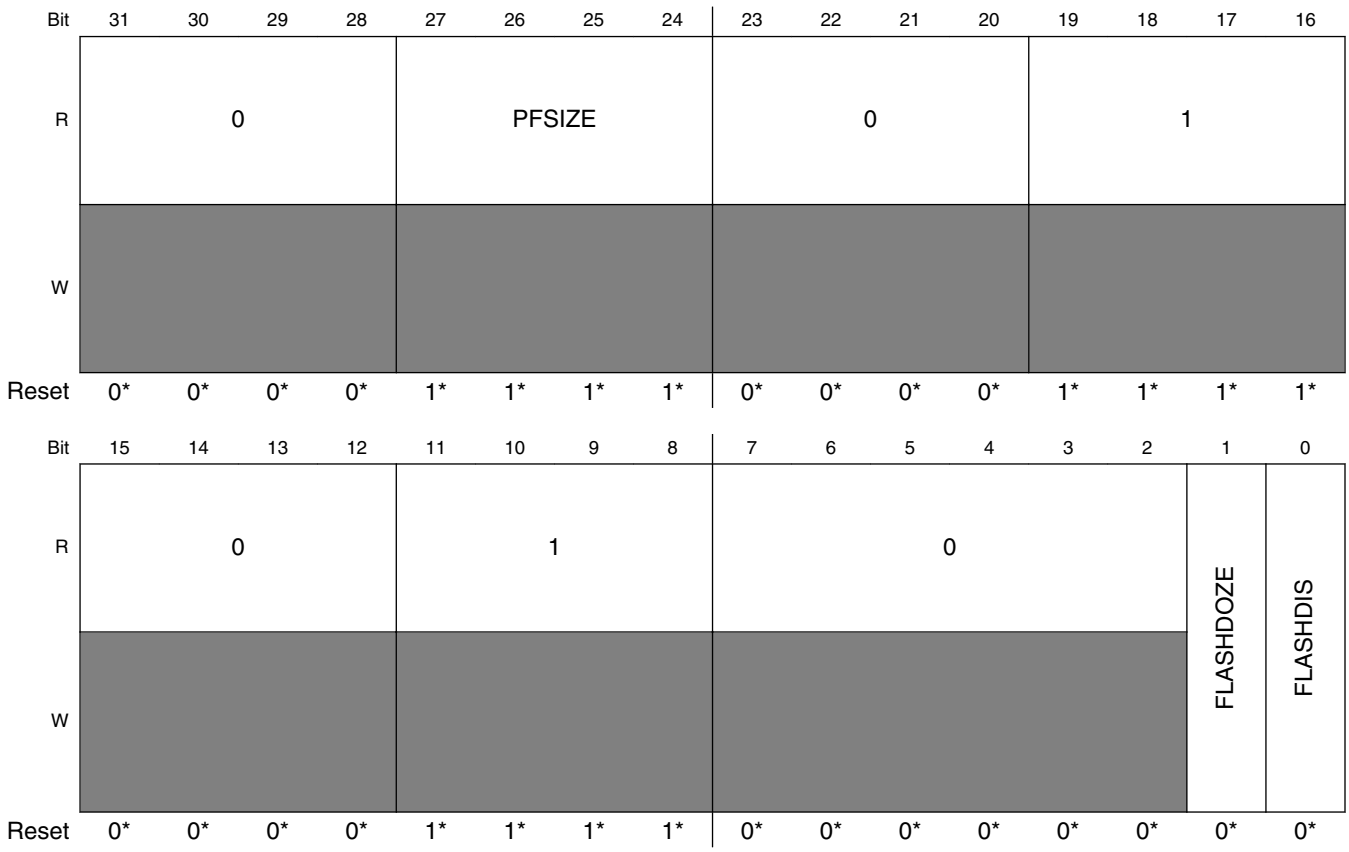


SIM_CLKDIV2 field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–1 USBDIV	USB clock divider divisor This field sets the divide value for the fractional clock divider when the MCGFLLCLK, or MCGPLLCLK, or IRC48M clock is the USB clock source (SOPT2[USBSRC] = 1). Divider output clock = Divider input clock × [(USBFRAC+1) / (USBDIV+1)]
0 USBFRAC	USB clock divider fraction This field sets the fraction multiply value for the fractional clock divider when the MCGFLLCLK, or MCGPLLCLK, or IRC48M clock is the USB clock source (SOPT2[USBSRC] = 1). Divider output clock = Divider input clock × [(USBFRAC+1) / (USBDIV+1)]

13.2.13 Flash Configuration Register 1 (SIM_FCFG1)

Address: 4004_7000h base + 104Ch offset = 4004_804Ch



SIM_FCFG1 field descriptions

Field	Description
31–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–24 PFSIZE	Program flash size This field specifies the amount of program flash memory available on the device . Undefined values are reserved. 0011 32 KB of program flash memory 0101 64 KB of program flash memory 0111 128 KB of program flash memory 1001 256 KB of program flash memory 1011 512 KB of program flash memory 1101 1024 KB of program flash memory 1111 128 KB of program flash memory

Table continues on the next page...

SIM_FCFG1 field descriptions (continued)

Field	Description
23–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
7–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 FLASHDOZE	Flash Doze 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 vectors do not need to be relocated out of Flash memory. The wakeup time from Wait mode is extended when this bit is set. 0 Flash remains enabled during Wait mode 1 Flash is disabled for the duration of Wait mode
0 FLASHDIS	Flash Disable 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. 0 Flash is enabled 1 Flash is disabled

13.2.14 Flash Configuration Register 2 (SIM_FCFG2)

Address: 4004_7000h base + 1050h offset = 4004_8050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MAXADDR0							1	0						
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	1*	0*	0*	0*	0*	0*	0*	0*

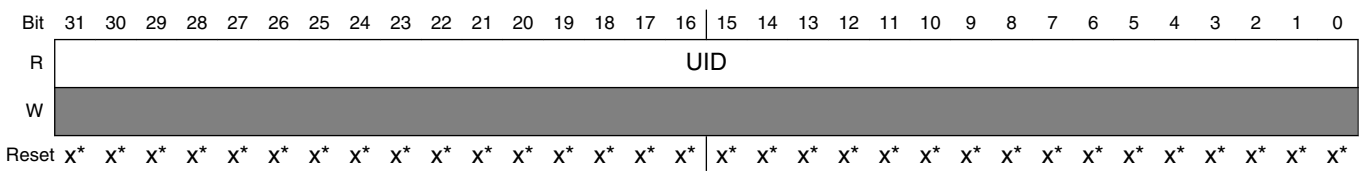
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

SIM_FCFG2 field descriptions

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–24 MAXADDR0	Max address block 0 This field concatenated with 13 trailing zeros indicates the first invalid address of each program flash block. 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

13.2.15 Unique Identification Register High (SIM_UIDH)

Address: 4004_7000h base + 1054h offset = 4004_8054h

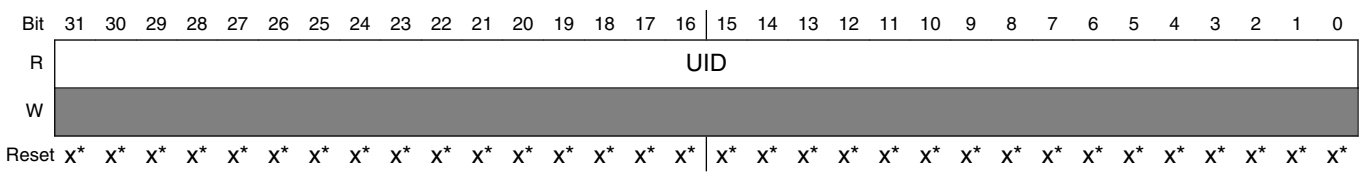


SIM_UIDH field descriptions

Field	Description
UID	Unique Identification Unique identification [127:96] for the device.

13.2.16 Unique Identification Register Mid-High (SIM_UIDMH)

Address: 4004_7000h base + 1058h offset = 4004_8058h



SIM_UIDMH field descriptions

Field	Description
UID	Unique Identification Unique identification [95:64] for the device.

13.2.17 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	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*

SIM_UIDML field descriptions

Field	Description
UID	Unique Identification Unique identification [63:32] for the device.

13.2.18 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	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*

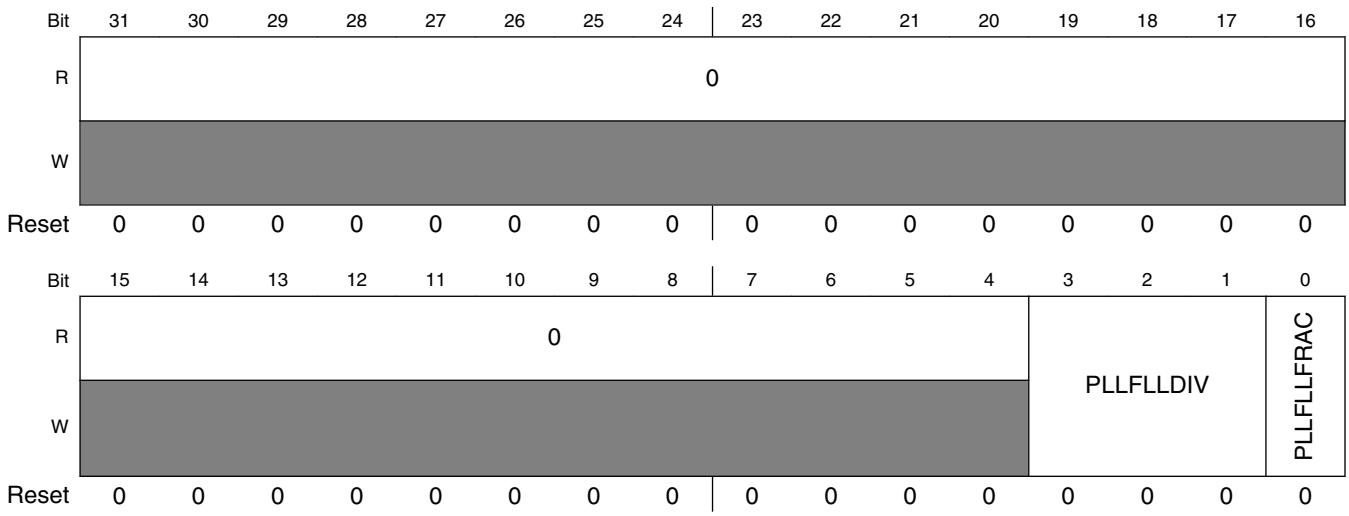
SIM_UIDL field descriptions

Field	Description
UID	Unique Identification Unique identification [31:0] for the device.

13.2.19 System Clock Divider Register 3 (SIM_CLKDIV3)

This register should only be written when the LPUART, EMVSIM,FlexIO and TPM modules are disabled.

Address: 4004_7000h base + 1064h offset = 4004_8064h

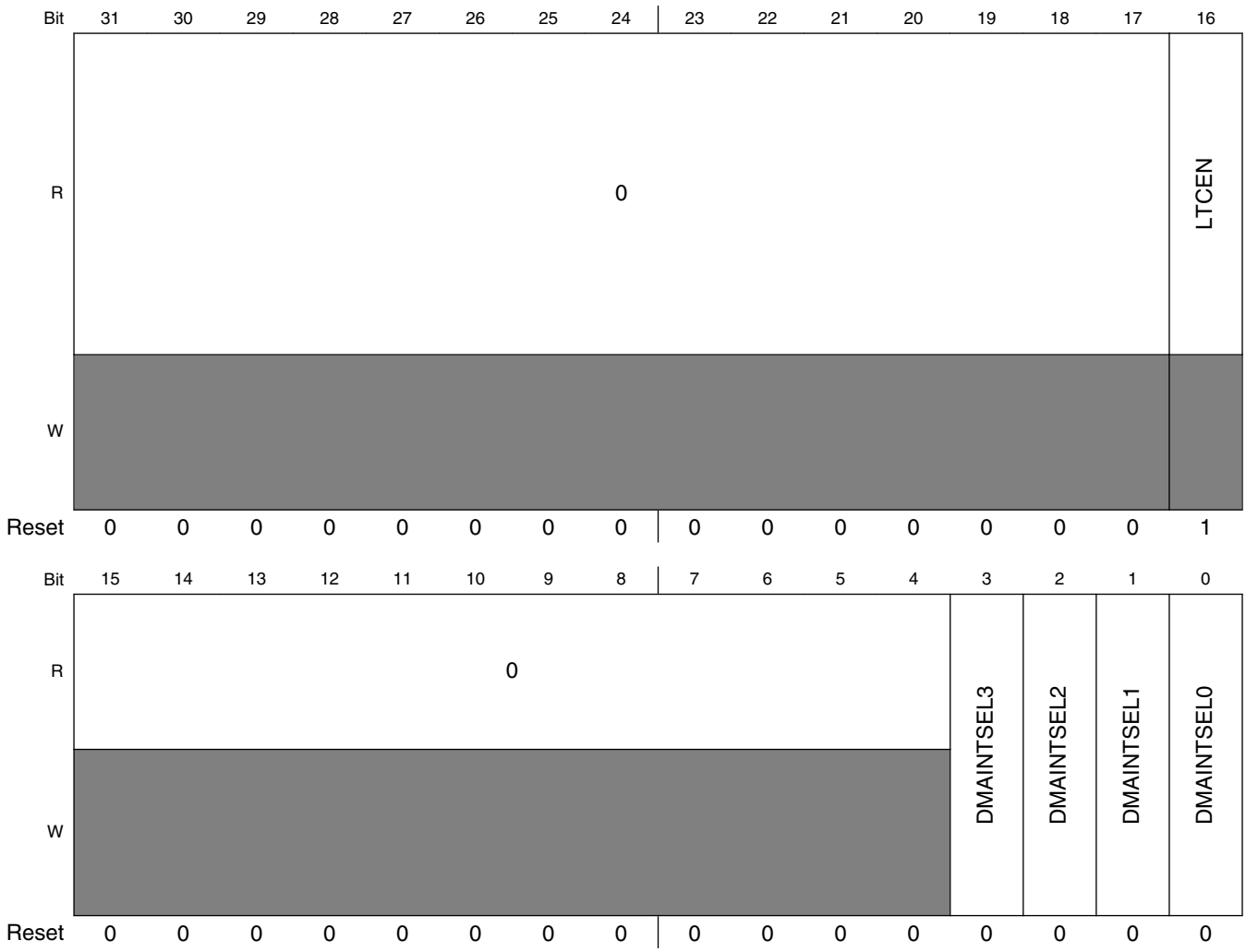


SIM_CLKDIV3 field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–1 PLLFLLDIV	PLLFLL clock divider divisor This field sets the divide value for the fractional clock divider used as a source for various peripheral clocks. The source clock for the fractional clock divider is set by the SOPT2 PLLFLLSEL register bit. Divider output clock = Divider input clock * ((PLLFLLFRAC+1)/(PLLFLLDIV+1))
0 PLLFLLFRAC	PLLFLL clock divider fraction This field sets the divide value for the fractional clock divider used as a source for various peripherals. The source clock for the fractional clock divider is set by the SOPT2 PLLFLLSEL register bit. Divider output clock = Divider input clock*((PLLFLLFRAC+1)/(PLLFLLDIV+1))

13.2.20 Misc Control Register (SIM_MISCCTRL)

Address: 4004_7000h base + 106Ch offset = 4004_806Ch



SIM_MISCCTRL field descriptions

Field	Description
31–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 LTCEN	LTC Status Indicate the status of the LTC. 0 LTC is not available. 1 LTC is available.
15–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIM_MISCCTRL field descriptions (continued)

Field	Description
3 DMAINTSEL3	DMA Channel Interrupts Select 3 Enable DMA channel 7 in vector 19. 0 DMA0 channel 7 is not available in vector 19. 1 DMA0 channel 7 is available in vector 19.
2 DMAINTSEL2	DMA Channel Interrupts Select 2 Enable DMA channel 6 in vector 18. 0 DMA0 channel 6 is not available in vector 18. 1 DMA0 channel 6 is available in vector 18.
1 DMAINTSEL1	DMA Channel Interrupts Select 1 Enable DMA channel 5 in vector 17. 0 DMA0 channel 5 is not available in vector 17. 1 DMA0 channel 5 is available in vector 17.
0 DMAINTSEL0	DMA Channel Interrupts Select 0 Enable DMA channel 4 in vector 16. 0 DMA0 channel 4 is not available in vector 16. 1 DMA0 channel 4 is available in vector 16.

13.2.21 Secure Key Register 0 (SIM_SECKEY0)

Address: 4004_7000h base + 1090h offset = 4004_8090h

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	SECKEY																															
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*

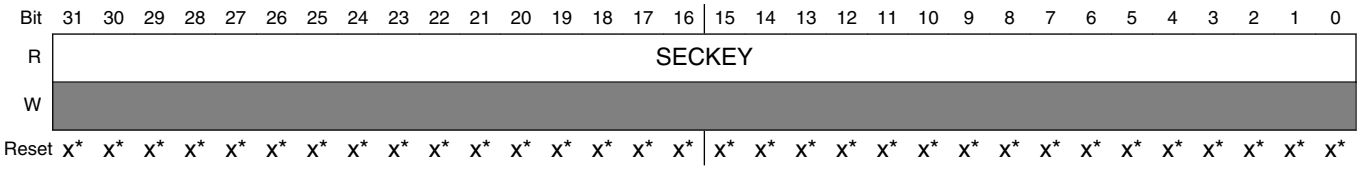
* Notes:
- x = Undefined at reset.

SIM_SECKEY0 field descriptions

Field	Description
SECKEY	Secure Key 31:0 During reset, these bit fieleds are loaded by the data stored inside the Program Once Field with indexes 0x20. If the flash is in secure state, this register is cleared once read. If the flash is in unsecure state, this register is always read as 0.

13.2.22 Secure Key Register 1 (SIM_SECKEY1)

Address: 4004_7000h base + 1094h offset = 4004_8094h



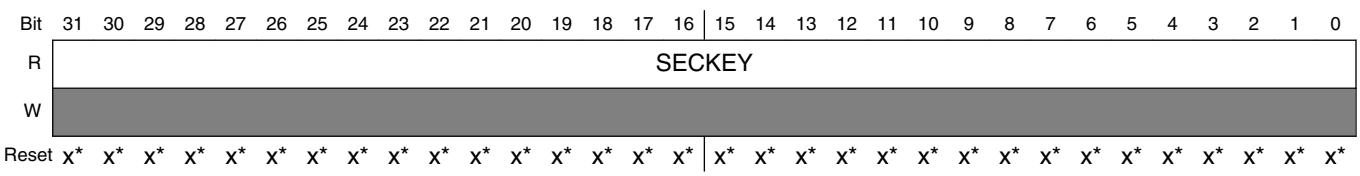
* Notes:
- x = Undefined at reset.

SIM_SECKEY1 field descriptions

Field	Description
SECKEY	Secure Key 31:0 During reset, these bit fieleds are loaded by the data stored inside the Program Once Field with indexes 0x21. If the flash is in secure state, this register is cleared once read. If the flash is in unsecure state, this register is always read as 0.

13.2.23 Secure Key Register 2 (SIM_SECKEY2)

Address: 4004_7000h base + 1098h offset = 4004_8098h



* Notes:
- x = Undefined at reset.

SIM_SECKEY2 field descriptions

Field	Description
SECKEY	Secure Key 31:0 During reset, these bit fieleds are loaded by the data stored inside the Program Once Field with indexes 0x22. If the flash is in secure state, this register is cleared once read. If the flash is in unsecure state, this register is always read as 0.

13.2.24 Secure Key Register 3 (SIM_SECKEY3)

Address: 4004_7000h base + 109Ch offset = 4004_809Ch

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	SECKEY																															
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.

SIM_SECKEY3 field descriptions

Field	Description
SECKEY	Secure Key 31:0 During reset, these bit fieleds are loaded by the data stored inside the Program Once Field with indexes 0x23. If the flash is in secure state, this register is cleared once read. If the flash is in unsecure state, this register is always read as 0.

Chapter 14
EMV SIM

14.1 Chip-specific EMVSIM information

14.1.1 Overview

The EMV_SIM (Euro/Mastercard/Visa/SIM Serial Interface Module) is a module that supports ISO-7816 protocol.

14.1.2 EMV_SIM instantiations

This device contains 2 EMV_SIM modules. This section describes how each module is configured on this device.

Table 14-1. EMV_SIM instantiations

Feature	Parameter settings
EMV_SIMs instantiated	2
EMV_SIM0 FIFO (RX)	16 bytes(Standard is 16 bytes)
EMV_SIM0 FIFO (TX)	16 bytes(Standard is 16 bytes)
EMV_SIM1 FIFO (RX)	16 bytes(Standard is 16 bytes)
EMV_SIM1 FIFO (TX)	16 bytes(Standard is 16 bytes)

14.2 Introduction

The Euro, MasterCard, Visa Subscriber Identification Module (EMV SIM) is designed to facilitate communication to Smart Cards compatible to the EMV ver4.3 standard (Book 1) and Smart Cards compatible with ISO/IEC 7816-3 Standard.

14.2.1 Features

The EMV SIM module supports the following features:

- Supports Smart Cards based on the EMV Standard v4.3 and ISO 7816-3 standard
- Independent clock for SIM logic (transmitter + receiver) and independent clock for register read-write interface
- 16 byte deep FIFO for transmitter and receiver
- Automatic NACK generation on parity error and receiver FIFO overflow error
- Support for both Inverse and Direct conventions
- Re-transmission of byte upon Smart Card NACK request with programmable threshold of re-transmissions
- Auto detection of Initial Character in receiver and setting of data format (inverse or direct)
- NACK detection in receiver
- Independent timers to measure character wait time, block wait time and block guard time
- Two general purpose counters available for use by software application with programmable clock selection for the counters
- DMA support available to transfer data to/from FIFOs. Programmable option available to select interrupt or DMA feature
- Programmable Prescaler to generate the desired frequency for Card Clock and Baud Rate Divisor to generate the internal ETU clocks for transmitter and receiver for any F/D ratio
- Deep sleep wake-up via Smart Card presence detect interrupt
- Manual control of all Smart Card interface signals
- Automatic power down of port logic on Smart Card presence detect
- Support for 8-bit LRC and 16-bit CRC generation for bytes sent out from transmitter and checking incoming message checksum for receiver

14.3 Block Diagram

Figure below shows the block diagram for the EMV SIM module.

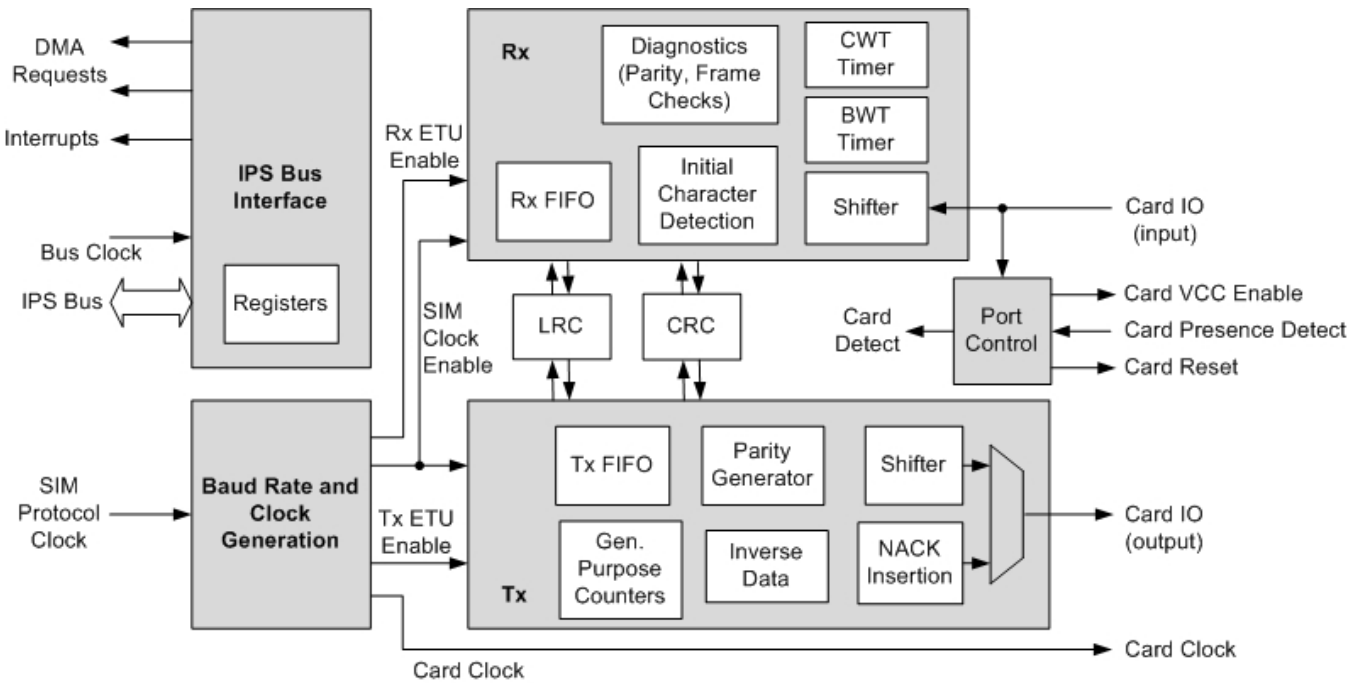


Figure 14-1. EMV SIM Block Diagram

14.4 Design Overview

The EMV SIM is designed to be compliant to the EMV ICC Specifications ver. 4.3 (dated Nov 2011) and the ISO-7816-3 standard.

The EMV SIM module is a block that is configured via the device's peripheral bus interface. The module has 32-bit aligned memory mapped registers that are used to configure and control the operations of the module. The software can write the transmit bytes or read the received bytes using either interrupt operation or DMA operation. The module can generate necessary DMA requests for the operation.

The module interfaces to the external card (Smart Card) via card clock, card reset, card V_{CC} enable and IO pins. The transmitter output from the module and the receiver input to the module are connected to the same IO pin and used in the open drain configuration. A pull-up resistor must be connected to the IO pin on board to allow proper functioning.

The key functional blocks in the module are clocking, transmitter and receiver which control the interactions with the Smart Card. These blocks function on a clock that is different and asynchronous to the register read/write clock. In order for the transmitter and receiver to function properly, the respective configuration must be done before enabling the transmitter or receiver.

The clocking module generates the card clock (output to Smart Card), and the internal clocks for the transmitter and receiver. The internal clocks control the ETU period and the sampling and driving of data on the IO line. The receive operation uses an internal over-sampled clock that oversamples the received input at 16x times the ETU rate. The prescaler generates the card clock in the range of 1 to 5 MHz with near 50% duty cycle and the baud divider generates the internal ETU clocks for transmitter and receiver.

The transmitter comprises of a 16-byte deep FIFO to store the bytes to be sent to the Smart Card. These bytes are sent to the Smart Card serially by the transmitter. The data format (i.e. inverse or direct conversion) is supported and the byte is converted into the right data format before being shifted out serially. The transmitter also supports retransmission of the current byte on the reception of a NACK. The transmitter automatically inserts NACK when the receiver indicates a parity or FIFO overflow error, if configured to do so by software. The transmitter is not required to be enabled for this. The transmitter also includes timer to insert the programmable guard time between the transmitter bytes and general purpose timers to allow software performs certain measurements. The status of the transmitter operation is intimated to software via maskable interrupts.

The receiver comprises of a 16-byte deep FIFO to store the correctly received bytes from the Smart Card. Software must ensure that it reads out the data from FIFO to prevent any overflow when more bytes are sent from the card. The receiver can be configured to detect the initial character. When configured to do so, the data format (direct or inverse convention) is also determined by the receiver and appropriate bit is set in the register map. Internal timers in the receiver measure the various wait times (Character, Block and Guard). On reception of a byte, the receiver checks the byte for Parity Error (in T=0 mode) and Framing Error. It can then direct the transmitter to insert NACK on parity error detection. For a block of bytes received, the receiver checks for LRC or CRC errors (in T=1 mode). Other errors are stored as status for software to be aware of their occurrence. The status of the receiver operation is intimated to software via maskable interrupts.

In addition to the above blocks, the module also contains the 8-bit LRC and the 16-bit CRC blocks that are common to the transmitter and receiver. Both transmitter and receiver can use the LRC and CRC blocks and can insert the LRC or CRC into the byte stream or check the validity of the input stream using the LRC and CRC values received. Since transmit and receive operations are mutually exclusive, common LRC and CRC blocks for the transmitter and receiver can be used.

The Smart Card interface signals, except IO, which is controlled by the transmitter and receiver; are generated by the port control block which controls the assertion and de-assertion of the clock, reset and VCC signals to Smart Card. It can do so manually (software needs to explicitly write the necessary bits) or using auto power down wherein

the module can itself power down the Smart Card when configured to do so by software. The port control block can also detect the Smart Card presence and generate necessary interrupts to indicate the same.

14.5 Signal Description

The following table shows the user-accessible signals available on EMV SIM. See the chip-level specification to find out which signals are actually connected to the external pins.

Table 14-2. EMV SIM Signal Description

Signal	Description	I/O
EMVSIM_SCLK	Card Clock. Clock to Smart Card.	O
EMVSIM_SRST	Card Reset. Reset signal to Smart Card	O
EMVSIM_VCC_EN	Card Power Enable. This signal controls the power to Smart Card	O
EMVSIM_IO	Card Data Line. Bi-directional data line.	I/O
EMVSIM_PD	Card Presence Detect. Signal indicating presence or removal of card	I

14.6 Memory Map and Registers

The memory map comprises of 32-bit aligned registers which can be accessed via 8-bit, 16-bit, or 32-bit accesses. Write access to reserved locations will generate transfer error. Read access to reserved locations will also generate transfer error and the read data bus will show all 0s.

The module will not check for correctness of programmed values in the registers and software must ensure that correct values are being written.

EMVSIM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_E000	Version ID Register (EMVSIM0_VER_ID)	32	R	0000_0000h	14.6.1/293
4004_E004	Parameter Register (EMVSIM0_PARAM)	32	R	0000_1010h	14.6.2/293
4004_E008	Clock Configuration Register (EMVSIM0_CLKCFG)	32	R/W	0000_0000h	14.6.3/294
4004_E00C	Baud Rate Divisor Register (EMVSIM0_DIVISOR)	32	R/W	0000_0174h	14.6.4/295
4004_E010	Control Register (EMVSIM0_CTRL)	32	R/W	0100_0006h	14.6.5/296

Table continues on the next page...

EMVSIM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_E014	Interrupt Mask Register (EMVSIM0_INT_MASK)	32	R/W	0000_FFFFh	14.6.6/300
4004_E018	Receiver Threshold Register (EMVSIM0_RX_THD)	32	R/W	0000_0001h	14.6.7/302
4004_E01C	Transmitter Threshold Register (EMVSIM0_TX_THD)	32	R/W	0000_000Fh	14.6.8/303
4004_E020	Receive Status Register (EMVSIM0_RX_STATUS)	32	w1c	0000_0000h	14.6.9/304
4004_E024	Transmitter Status Register (EMVSIM0_TX_STATUS)	32	w1c	0000_00B8h	14.6.10/307
4004_E028	Port Control and Status Register (EMVSIM0_PCSR)	32	R/W	0100_0000h	14.6.11/310
4004_E02C	Receive Data Read Buffer (EMVSIM0_RX_BUF)	32	R	0000_0000h	14.6.12/312
4004_E030	Transmit Data Buffer (EMVSIM0_TX_BUF)	32	W (always reads 0)	0000_0000h	14.6.13/313
4004_E034	Transmitter Guard ETU Value Register (EMVSIM0_TX_GETU)	32	R/W	0000_0000h	14.6.14/313
4004_E038	Character Wait Time Value Register (EMVSIM0_CWT_VAL)	32	R/W	0000_FFFFh	14.6.15/314
4004_E03C	Block Wait Time Value Register (EMVSIM0_BWT_VAL)	32	R/W	FFFF_FFFFh	14.6.16/314
4004_E040	Block Guard Time Value Register (EMVSIM0_BGT_VAL)	32	R/W	0000_0000h	14.6.17/314
4004_E044	General Purpose Counter 0 Timeout Value Register (EMVSIM0_GPCNT0_VAL)	32	R/W	0000_FFFFh	14.6.18/315
4004_E048	General Purpose Counter 1 Timeout Value (EMVSIM0_GPCNT1_VAL)	32	R/W	0000_FFFFh	14.6.19/315
4004_F000	Version ID Register (EMVSIM1_VER_ID)	32	R	0000_0000h	14.6.1/293
4004_F004	Parameter Register (EMVSIM1_PARAM)	32	R	0000_1010h	14.6.2/293
4004_F008	Clock Configuration Register (EMVSIM1_CLKCFG)	32	R/W	0000_0000h	14.6.3/294
4004_F00C	Baud Rate Divisor Register (EMVSIM1_DIVISOR)	32	R/W	0000_0174h	14.6.4/295
4004_F010	Control Register (EMVSIM1_CTRL)	32	R/W	0100_0006h	14.6.5/296
4004_F014	Interrupt Mask Register (EMVSIM1_INT_MASK)	32	R/W	0000_FFFFh	14.6.6/300
4004_F018	Receiver Threshold Register (EMVSIM1_RX_THD)	32	R/W	0000_0001h	14.6.7/302
4004_F01C	Transmitter Threshold Register (EMVSIM1_TX_THD)	32	R/W	0000_000Fh	14.6.8/303
4004_F020	Receive Status Register (EMVSIM1_RX_STATUS)	32	w1c	0000_0000h	14.6.9/304
4004_F024	Transmitter Status Register (EMVSIM1_TX_STATUS)	32	w1c	0000_00B8h	14.6.10/307
4004_F028	Port Control and Status Register (EMVSIM1_PCSR)	32	R/W	0100_0000h	14.6.11/310
4004_F02C	Receive Data Read Buffer (EMVSIM1_RX_BUF)	32	R	0000_0000h	14.6.12/312
4004_F030	Transmit Data Buffer (EMVSIM1_TX_BUF)	32	W (always reads 0)	0000_0000h	14.6.13/313
4004_F034	Transmitter Guard ETU Value Register (EMVSIM1_TX_GETU)	32	R/W	0000_0000h	14.6.14/313
4004_F038	Character Wait Time Value Register (EMVSIM1_CWT_VAL)	32	R/W	0000_FFFFh	14.6.15/314
4004_F03C	Block Wait Time Value Register (EMVSIM1_BWT_VAL)	32	R/W	FFFF_FFFFh	14.6.16/314
4004_F040	Block Guard Time Value Register (EMVSIM1_BGT_VAL)	32	R/W	0000_0000h	14.6.17/314

Table continues on the next page...

EMVSIM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_F044	General Purpose Counter 0 Timeout Value Register (EMVSIM1_GPCNT0_VAL)	32	R/W	0000_FFFFh	14.6.18/315
4004_F048	General Purpose Counter 1 Timeout Value (EMVSIM1_GPCNT1_VAL)	32	R/W	0000_FFFFh	14.6.19/315

14.6.1 Version ID Register (EMVSIMx_VER_ID)

Version ID for the IP. It corresponds to the version of IP being used.

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	VER																															
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

EMVSIMx_VER_ID field descriptions

Field	Description
VER	Version ID of the module EMV SIM Version Number used on the chip 31:16 - Major Version (example: 01.00) 15:0 - Minor Version (example: 01.00)

14.6.2 Parameter Register (EMVSIMx_PARAM)

This register provides details on the parameter settings that were used while including this module in the chip.

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																TX_FIFO_DEPTH								RX_FIFO_DEPTH							
W																																
Reset	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	1	0	0	0	0

EMVSIMx_PARAM field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–8 TX_FIFO_ DEPTH	Transmit FIFO Depth Value of parameter for Transmit FIFO Depth (in Bytes)
RX_FIFO_ DEPTH	Receive FIFO Depth Value of parameter for Receive FIFO Depth (in Bytes)

14.6.3 Clock Configuration Register (EMVSIMx_CLKCFG)

This register provides configuration details to enable the right clock frequency of clocks used by EMV SIM module.

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				GPCNT0_ CLK_SEL		GPCNT1_ CLK_SEL		CLK_PRSC							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EMVSIMx_CLKCFG field descriptions

Field	Description
31–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–10 GPCNT0_CLK_ SEL	General Purpose Counter 0 Clock Select Selects which clock source is used by EMV SIM Module general purpose counter 0. The only way to reset the counter is to set these bits to zero. The counter will begin counting as soon as the clock input is selected and the clocks are active. These input clocks are enabled through other register bits of the EMV SIM module (KILL_CLOCK, RCV_EN, and XMT_EN). Counter is active while RCV_EN or XMT_EN are set. 00 Disabled / Reset (default) 01 Card Clock 10 Receive Clock 11 ETU Clock (transmit clock)
9–8 GPCNT1_CLK_ SEL	General Purpose Counter 1 Clock Select Selects which clock source is used by EMV SIM Module general purpose counter 1. The only way to reset the counter is to set these bits to zero. The counter will begin counting as soon as the clock input is selected and the clocks are active. These input clocks are enabled through other register bits of the EMV

Table continues on the next page...

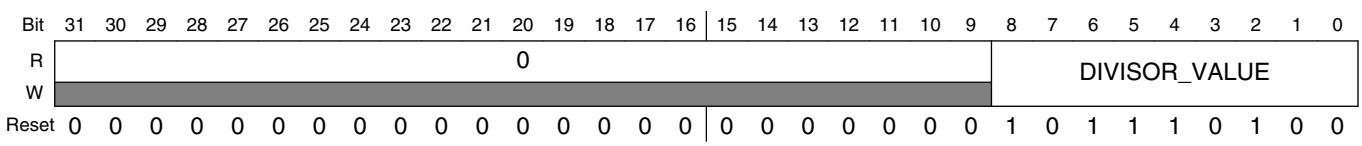
EMVSIMx_CLKCFG field descriptions (continued)

Field	Description
	SIM module (KILL_CLOCK, RCV_EN, and XMT_EN). Counter is active while RCV_EN or XMT_EN are set. 00 Disabled / Reset (default) 01 Card Clock 10 Receive Clock 11 ETU Clock (transmit clock)
CLK_PRSC	Clock Prescaler Value The value written to this register will determine the desired card clock frequency. The Card Clock frequency is the EMV SIM Protocol (or logic) clock divided by this prescaler value. The prescaler value should be updated only when card clock is disabled and transmitter and receiver are not operational. 0, 1 Max Divide value used (default 0 is used) 2 Divide by 2 >2 Non-zero value will divide clock accordingly

14.6.4 Baud Rate Divisor Register (EMVSIMx_DIVISOR)

This regiter configures the divisor value to generate the baud clock which will drive the card clock and also generate the transmit and receive clocks and respective ETUs.

Address: Base address + Ch offset

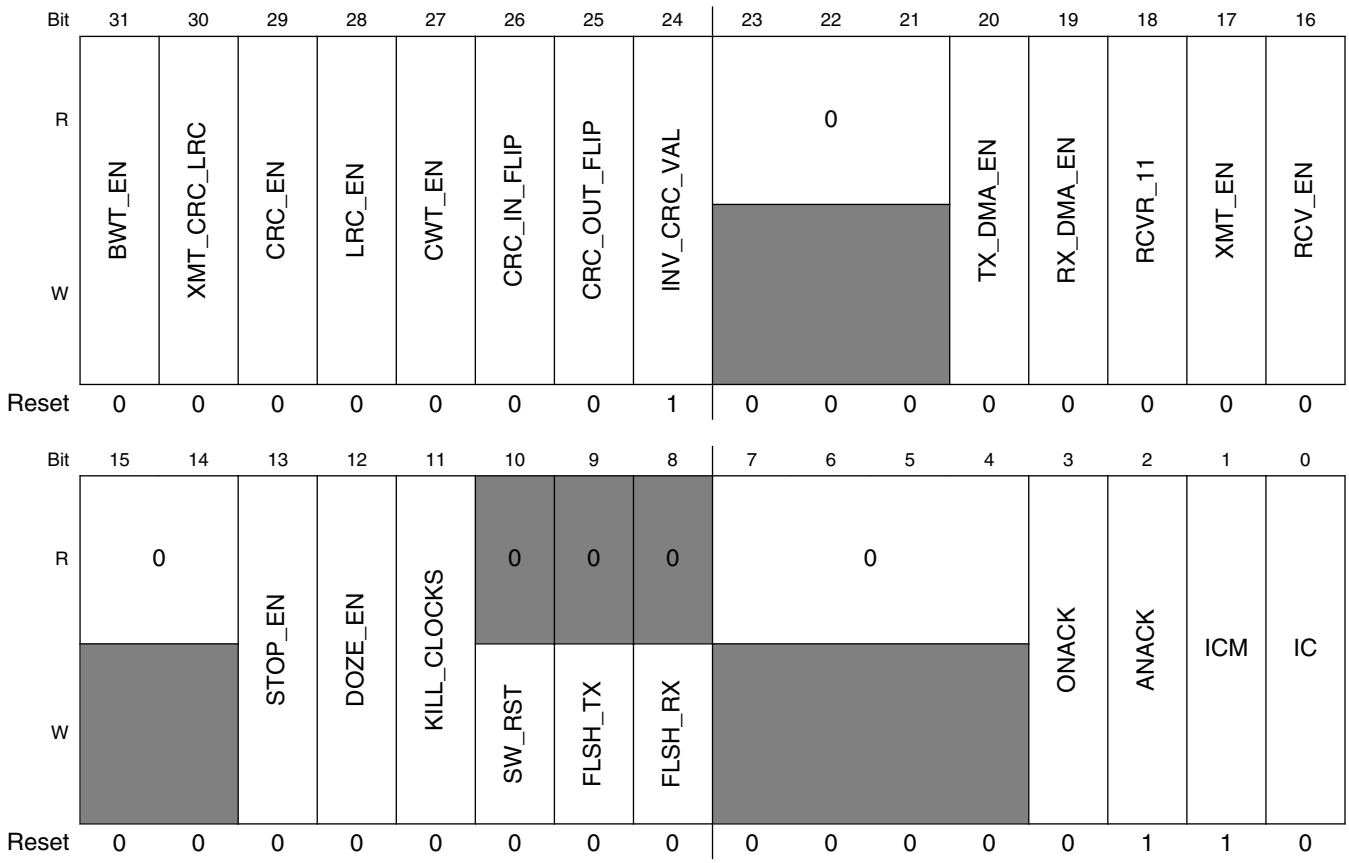


EMVSIMx_DIVISOR field descriptions

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
DIVISOR_ VALUE	Divisor (F/D) Value The value written to this register will be used to generate the ETU bit period that will be used by the transmitter and receiver. The divisor value is the integer result of F/D value; where F and D are the clock rate conversion integer and baud rate adjustment integer, respectively, that required for the desired operation. The value in this register can be changed when no transaction is active with the smart card. When programming this field, the fractional part of F/D should be rounded off the nearest integer. 0 - 4 Invalid. As per ISO 7816 specification, minimum value of F/D is 5 372 Divisor value for F = 372 and D = 1 (default) others Integer value of result of F/D

14.6.5 Control Register (EMVSIMx_CTRL)

Address: Base address + 10h offset



EMVSIMx_CTRL field descriptions

Field	Description
31 BWT_EN	Block Wait Time Counter Enable Writing a '1' to this bit will enable the BWT and BGT functions. The BWT and BGT functions can then be individually selected using the interrupt mask. 0 Disable BWT, BGT Counters (default) 1 Enable BWT, BGT Counters
30 XMT_CRC_LRC	Transmit CRC or LRC Enable This bit specifies whether or not to transmit the redundancy checking data (LRC or CRC) at the end of a transmission (that is, when the FIFO becomes empty). 0 No CRC or LRC value is transmitted (default) 1 Transmit LRC or CRC info when FIFO empties (whichever is enabled)
29 CRC_EN	CRC Enable This bit enables the calculation of the 16-bit CRC value for both receiver and transmitter. The result of the calculation is continuously compared to the expected remainder and reflected in the CRC_OK bit in the RX_STATUS register. Clearing this bit resets the current CRC residual value in the EMV SIM hardware.

Table continues on the next page...

EMVSIMx_CTRL field descriptions (continued)

Field	Description
	0 16-bit Cyclic Redundancy Checking disabled (default) 1 16-bit Cyclic Redundancy Checking enabled
28 LRC_EN	LRC Enable This bit enables the calculation of the 8-bit LRC value for both receiver and transmitter. The result of the calculation is continuously compared to zero and reflected in the LRC_OK bit in the RX_STATUS register. Clearing this bit resets the current LRC value in the EMV SIM hardware. 0 8-bit Linear Redundancy Checking disabled (default) 1 8-bit Linear Redundancy Checking enabled
27 CWT_EN	Character Wait Time Counter Enable Enables the character wait time counter. Clearing this bit resets the counter to zero. 0 Character Wait time Counter is disabled (default) 1 Character Wait time counter is enabled
26 CRC_IN_FLIP	CRC Input Byte's Bit Reversal or Flip Control This bit control whether the bits in the CRC input byte will be reversed before CRC calculation or not. When set to '1', the bits within the input byte for CRC will change from 7:0 to 0:7. For CCITT CRC calculation, this bit should be set to '1'. For any other CRC using same polynomial, this bit can be set accordingly. 0 Bits in the input byte will not be reversed (i.e. 7:0 will remain 7:0) before the CRC calculation (default) 1 Bits in the input byte will be reversed (i.e. 7:0 will become 0:7) before CRC calculation
25 CRC_OUT_FLIP	CRC Output Value Bit Reversal or Flip This bit control whether the bits in the CRC output bytes will be reversed or not. When set to '1', the bits within the two bytes for CRC Output will change from 15:0 to {8:15,0:7}. For CCITT CRC calculation, this bit should be set to '1'. For any other CRC using same polynomial, this bit can be set accordingly. 0 Bits within the CRC output bytes will not be reversed i.e. 15:0 will remain 15:0 (default) 1 Bits within the CRC output bytes will be reversed i.e. 15:0 will become {8:15,0:7}
24 INV_CRC_VAL	Invert bits in the CRC Output Value This bit control whether the bits within the CRC Output value will be inverted (1's complement) or not. For CCITT CRC calculation, this bit should be set to '1'. For any other CRC using same polynomial, this bit can be set accordingly. 0 Bits in CRC Output value will not be inverted. 1 Bits in CRC Output value will be inverted. (default)
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 TX_DMA_EN	Transmit DMA Enable Enables assertion of DMA write request when Transmit FIFO is empty. Request is held asserted till Transmit FIFO reaches the programmed data threshold value. Transmit Data Threshold Interrupt will not be generated when this bit is asserted. 0 No DMA Write Request asserted for Transmitter (default) 1 DMA Write Request asserted for Transmitter

Table continues on the next page...

EMVSIMx_CTRL field descriptions (continued)

Field	Description
19 RX_DMA_EN	Receive DMA Enable Enables assertion of DMA read request when Receive FIFO reaches the programmed data threshold value. Request is held asserted till all the programmed threshold bytes are read out. Receiver Data Interrupt will not be generated when this bit is asserted. 0 No DMA Read Request asserted for Receiver (default) 1 DMA Read Request asserted for Receiver
18 RCVR_11	Receiver 11 ETU Mode Enable Used to configure the EMV SIM module receiver for 11 ETU operation (that is, 1 Stop bit). This bit is provided for support of T=1 cards. 0 Receiver configured for 12 ETU operation mode (default) 1 Receiver configured for 11 ETU operation mode
17 XMT_EN	Transmitter Enable Used to enable/disable the EMV SIM transmitter block. It can be set to 0 during the auto power down sequence. **Note:** Setting this bit (transition from 0 to 1) will reset the CRC and LRC values. 0 EMV SIM Transmitter disabled (default) 1 EMV SIM Transmitter enabled
16 RCV_EN	Receiver Enable Used to enable/disable the EMV SIM receiver block. Once the transmitter has completed its operation, the software must enable the receiver using this bit. It can be set to 0 during the auto power down sequence. 0 EMV SIM Receiver disabled (default) 1 EMV SIM Receiver enabled
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 STOP_EN	STOP Enable Used to configure the operation of the EMV SIM module when a processor STOP instruction is executed. This bit is added to provide support for Smart Cards that do not allow the Smart Card clock to be stopped while power is applied. **Note:** To enable Card Clock in STOP mode, both STOP_EN and DOZE_EN should be set to 1. 0 STOP instruction shuts down all EMV SIM clocks (default) 1 STOP instruction shuts down all clocks except for the Smart Card Clock (SCK) (clock provided to Smart Card)
12 DOZE_EN	Doze Enable Used to configure the operation of the EMV SIM module when a processor DOZE instruction is executed. 0 DOZE instruction will gate all internal EMV SIM clocks as well as the Smart Card clock when the transmit FIFO is empty (default) 1 DOZE instruction has no effect on EMV SIM module
11 KILL_CLOCKS	Kill all internal clocks Used to enable/disable the clock input to the EMV SIM module. This bit will gate all clocks including the Smart Card clock regardless of the state of the STOP bit described above.

Table continues on the next page...

EMVSIMx_CTRL field descriptions (continued)

Field	Description
	Note: This bit will have no effect on the register read write clock. Only EMV SIM logic clock is gated. 0 EMV SIM input clock enabled (default) 1 EMV SIM input clock is disabled
10 SW_RST	Software Reset Bit Used to reset the entire EMV SIM module. This acts the same as a hardware reset for the EMV SIM module. This bit is self-clearing and always reads 0. **Note:** Software should allow a minimum of 4 Protocol clock cycles before attempting to access the EMV SIM module after a software reset. 0 EMV SIM Normal operation (default) 1 EMV SIM held in Reset
9 FLSH_TX	Flush Transmitter Bit This bit operates as an EMV SIM transmitter reset. The receive portion of the EMV SIM module is not affected. This bit clears automatically and always reads 0. 0 EMV SIM Transmitter normal operation (default) 1 EMV SIM Transmitter held in Reset
8 FLSH_RX	Flush Receiver Bit This bit operates as an EMV SIM receiver reset. The transmit portion of the EMV SIM module is not affected. This bits clears automatically and always reads 0. 0 EMV SIM Receiver normal operation (default) 1 EMV SIM Receiver held in Reset
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 ONACK	Overrun NACK Enable Enables NACK generation when Rx FIFO is full and another message is ready for writing into the FIFO. 0 NACK generation on overrun is disabled (default) 1 NACK generation on overrun is enabled
2 ANACK	Auto NACK Enable Enables NACK generation for parity errors in receieved messages or when invalid initial characters are received in ICM mode. 0 NACK generation on errors disabled 1 NACK generation on errors enabled (default)
1 ICM	Initial Character Mode Enables initial character mode. Will be automatically cleared by hardware once a valid initial character is received. 0 Initial Character Mode disabled 1 Initial Character Mode enabled (default)
0 IC	Inverse Convention

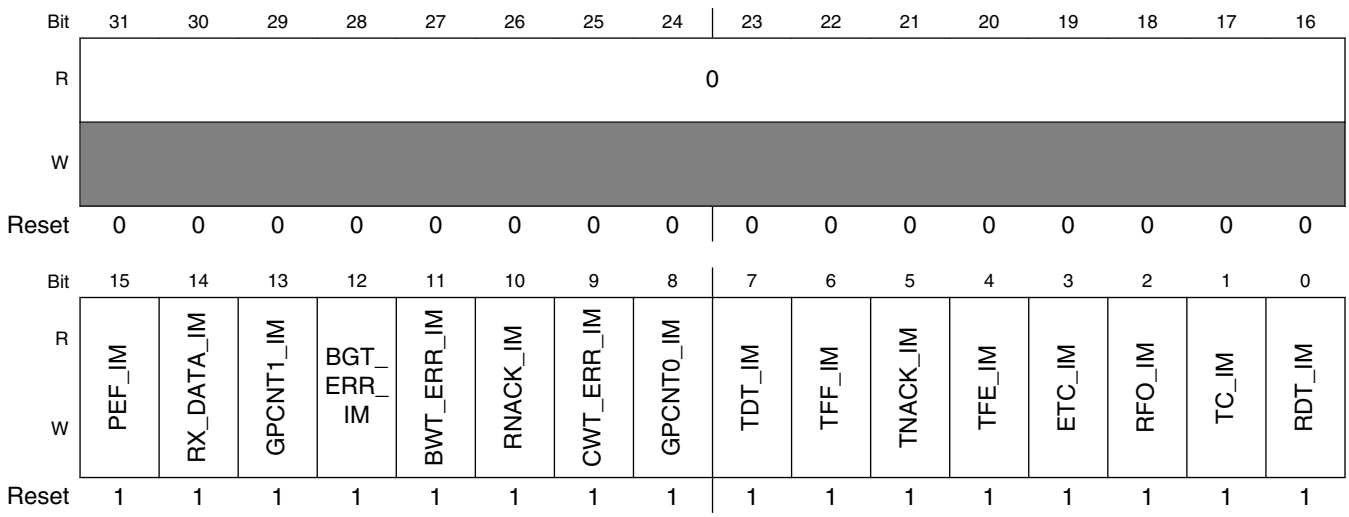
Table continues on the next page...

EMVSIMx_CTRL field descriptions (continued)

Field	Description
	Used to configure the EMV SIM to use either inverse convention or direct convention for its data format. The IC bit can be controlled by software, but it is normally set by hardware as a result of the interpretation of the initial character when in ICM mode. 0 Direction convention transfers enabled (default) 1 Inverse convention transfers enabled

14.6.6 Interrupt Mask Register (EMVSIMx_INT_MASK)

Address: Base address + 14h offset



EMVSIMx_INT_MASK field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 PEF_IM	Parity Error Interrupt Mask Used to enable/disable the ability of the PEF flag in the RX_STATUS register to generate EMV SIM interrupts. 0 PEF interrupt enabled 1 PEF interrupt masked (default)
14 RX_DATA_IM	Receive Data Interrupt Mask Used to enable/disable the ability of the RX_DATA flag in the RX_STATUS register to generate EMV SIM interrupts. 0 RX_DATA interrupt enabled 1 RX_DATA interrupt masked (default)
13 GPCNT1_IM	General Purpose Counter 1 Timeout Interrupt Mask

Table continues on the next page...

EMVSIMx_INT_MASK field descriptions (continued)

Field	Description
	Used to enable/disable the ability of the GPCNT1_TO flag in the TX_STATUS register to generate EMV SIM interrupts. 0 GPCNT1_TO interrupt enabled 1 GPCNT1_TO interrupt masked (default)
12 BGT_ERR_IM	Block Guard Time Error Interrupt Used to enable/disable the ability of the BGT_ERR flag in the RX_STATUS register to generate EMV SIM interrupts. 0 BGT_ERR interrupt enabled 1 BGT_ERR interrupt masked (default)
11 BWT_ERR_IM	Block Wait Time Error Interrupt Mask Used to enable/disable the ability of the BWT_ERR flag in the RX_STATUS register to generate EMV SIM interrupts. 0 BWT_ERR interrupt enabled 1 BWT_ERR interrupt masked (default)
10 RNACK_IM	Receiver NACK Threshold Interrupt Mask Used to enable/disable the ability of the RTE flag in the RX_STATUS register to generate EMV SIM interrupts. 0 RTE interrupt enabled 1 RTE interrupt masked (default)
9 CWT_ERR_IM	Character Wait Time Error Interrupt Mask Used to enable/disable the ability of the CWT_ERR flag in the RX_STATUS register to generate EMV SIM interrupts. 0 CWT_ERR interrupt enabled 1 CWT_ERR interrupt masked (default)
8 GPCNT0_IM	General Purpose Timer 0 Timeout Interrupt Mask Used to enable/disable the ability of the GPCNT0_TO flag in the TX_STATUS register to generate EMV SIM interrupts. 0 GPCNT0_TO interrupt enabled 1 GPCNT0_TO interrupt masked (default)
7 TDT_IM	Transmit Data Threshold Interrupt Mask Used to enable/disable the ability of the TDTF flag in the TX_STATUS register to generate EMV SIM interrupts. 0 TDTF interrupt enabled 1 TDTF interrupt masked (default)
6 TFF_IM	Transmit FIFO Full Interrupt Mask Used to enable/disable the ability of the TFF flag in the TX_STATUS register to generate EMV SIM interrupts. 0 TFF interrupt enabled 1 TFF interrupt masked (default)

Table continues on the next page...

EMVSIMx_INT_MASK field descriptions (continued)

Field	Description
5 TNACK_IM	Transmit NACK Threshold Interrupt Mask Used to enable/disable the ability of the TNTE flag in the TX_STATUS register to generate EMV SIM interrupts. 0 TNTE interrupt enabled 1 TNTE interrupt masked (default)
4 TFE_IM	Transmit FIFO Empty Interrupt Mask Used to enable/disable the ability of the TFE flag in the TX_STATUS register to generate EMV SIM interrupts. 0 TFE interrupt enabled 1 TFE interrupt masked (default)
3 ETC_IM	Early Transmit Complete Interrupt Mask Used to enable/disable the ability of the ETC flag in the TX_STATUS register to generate EMV SIM interrupts. 0 ETC interrupt enabled 1 ETC interrupt masked (default)
2 RFO_IM	Receive FIFO Overflow Interrupt Mask Used to enable/disable the ability of the RFO flag in the RX_STATUS register to generate EMV SIM interrupts. 0 RFO interrupt enabled 1 RFO interrupt masked (default)
1 TC_IM	Transmit Complete Interrupt Mask Used to enable/disable the ability of the TCF flag in the TX_STATUS register to generate EMV SIM interrupts. 0 TCF interrupt enabled 1 TCF interrupt masked (default)
0 RDT_IM	Receive Data Threshold Interrupt Mask Used to enable/disable the ability of the RDTF flag in the RX_STATUS register to generate EMV SIM interrupts. 0 RDTF interrupt enabled 1 RDTF interrupt masked (default)

14.6.7 Receiver Threshold Register (EMVSIMx_RX_THD)

Address: Base address + 18h 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																				RNCK_THD				Reserved				RDT				
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	1

EMVSIMx_RX_THD field descriptions

Field	Description
31–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–8 RNCK_THD	Receiver NACK Threshold Value Used to specify the number of consecutive NACK's transmitted by the EMV SIM module, for a given character, before the receive threshold error (RTE) flag is triggered. A value of 0 indicates that RTE is never set. When a valid character is received by the EMV SIM, the internal counter keeping track of the NACK count resets to zero for the subsequent byte being received. If the ANACK bit is clear in the CNTL register, RNCK_THD has no effect. 0 Zero Threshold. RTE will not be set >0 RTE will set after programmed value of NACKs are received
7–4 Reserved	This field is reserved.
RDT	Receiver Data Threshold Value Determines the number of bytes that must exist in the Receive FIFO to trigger the receive data threshold interrupt flag (RDTF). Value must not exceed the total bytes that can be stored in the Receive FIFO.

14.6.8 Transmitter Threshold Register (EMVSIMx_TX_THD)

Address: Base address + 1Ch 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																				TNCK_THD				0				TDT			
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	1	1	1

EMVSIMx_TX_THD field descriptions

Field	Description
31–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–8 TNCK_THD	Transmitter NACK Threshold Value Used to set the NACK threshold for the transmitter. Once the threshold number set by TNCK_THD has been reached for the current byte being transmitted, the error flag TNTE in the TX_STATUS register will be set. Setting of TNTE causes the remaining transmissions queued in the transmit FIFO to be aborted and no more transmissions to occur until software clears TNTE. To trigger TNTE, a given byte being transmitted must reach the TNCK_THD threshold itself. Transmit NACKs accumulated on one byte are not carried over to the next. 0 TNTE will never be set; retransmission after NACK reception is disabled. 1 TNTE will be set after 1 nack is received; 0 retransmissions occurs. 2 TNTE will be set after 2 nacks are received; at most 1 retransmission occurs. 3 TNTE will be set after 3 nacks are received; at most 2 retransmissions occurs. . . . . . . N TNTE will be set after N nacks are received; at most (N - 1) retransmissions occurs.

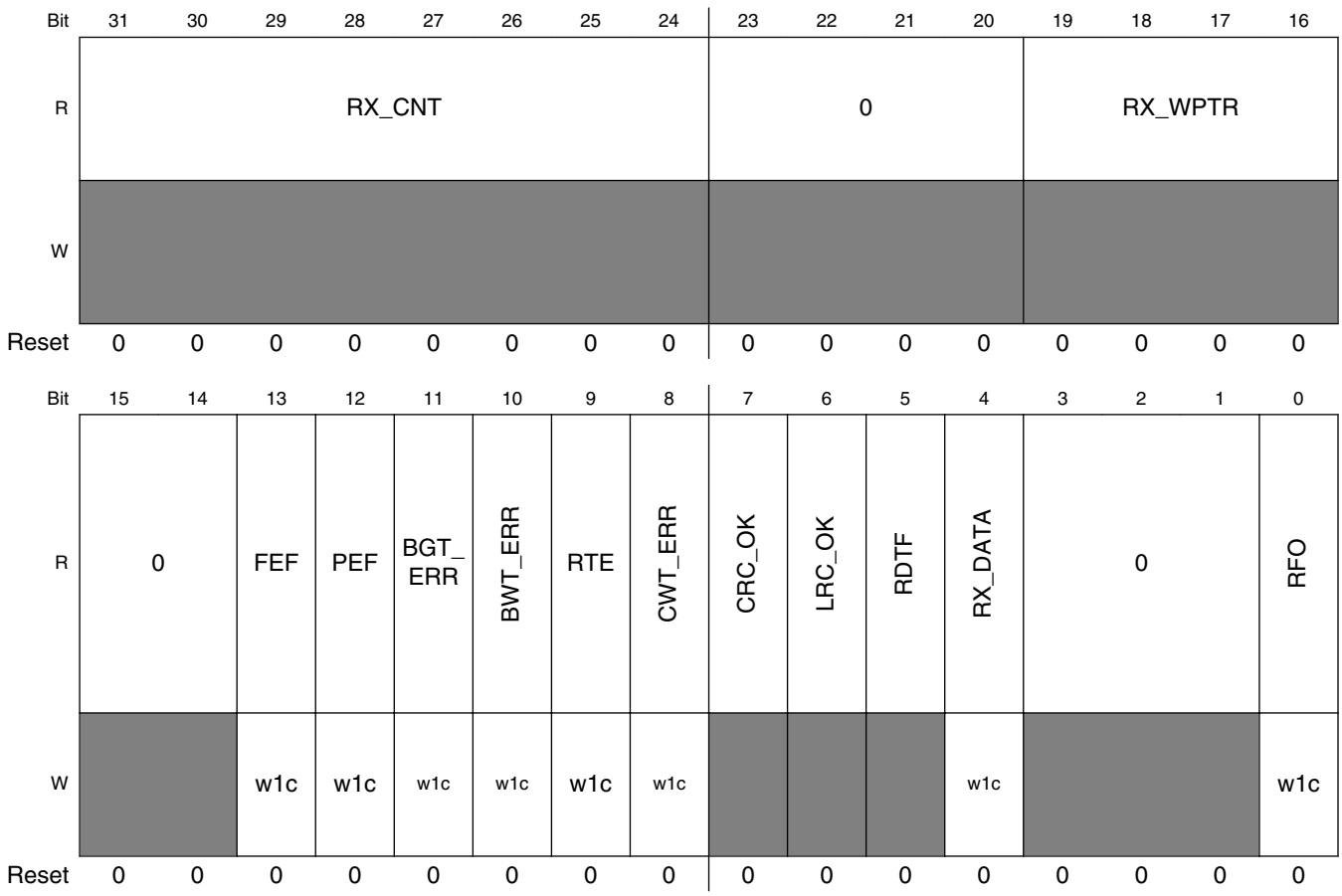
Table continues on the next page...

EMVSIMx_TX_THD field descriptions (continued)

Field	Description
	. . . . 15 TNTE will be set after 15 nacks are received; at most 14 retransmissions occurs.
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TDT	Transmitter Data Threshold Value Used to set the threshold value for the Transmit FIFO at which the TDTF bit in the TX_STATUS register will be set. When the number of bytes in the Transmit FIFO is less than or equal to TDT[3:0], TDTF will be set.

14.6.9 Receive Status Register (EMVSIMx_RX_STATUS)

Address: Base address + 20h offset



EMVSIMx_RX_STATUS field descriptions

Field	Description
31–24 RX_CNT	Receive FIFO Byte Count These bits indicate the number of bytes stored in the receive FIFO.

Table continues on the next page...

EMVSIMx_RX_STATUS field descriptions (continued)

Field	Description
	0 FIFO is emtpy >0 Total bytes available in FIFO
23–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–16 RX_WPTR	Receive FIFO Write Pointer Value Value of write pointer of Receive FIFO
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 FEF	Frame Error Flag Used to indicate if the current received byte did not have proper STOP bits. This will not cause a NACK transmission. This bit will not cause an interrupt assertion. It is simply an indication that a framing error has occured. 0 No frame error detected 1 Frame error detected
12 PEF	Parity Error Flag Used to indicate if a received byte had a parity error or not. A parity error will cause transmission of NACK if ANACK bit is set in CTRL register and PEF bit will not be asserted. When ANACK is *not* set, PEF will be asserted. This bit will also not assert when ICM bit is set; however parity is checked in ICM = 1 mode. It is simply an indication that a parity error has occured. 0 No parity error detected 1 Parity error detected
11 BGT_ERR	Block Guard Time Error Flag Used to indicate if the block guard time was too small. The threshold is set by the block guard time register. 0 Block guard time was sufficient 1 Block guard time was too small
10 BWT_ERR	Block Wait Time Error Flag Used to indicate if the block wait time has been exceeded. The threshold is set by the block wait time registers. 0 Block wait time not exceeded 1 Block wait time was exceeded
9 RTE	Received NACK Threshold Error Flag Used to indicate whether the number of consecutive NACK's generated by the EMV SIM module in response to receive parity errors, for the byte being received, equals the value programmed in the RTH[3:0] in the RX_THRESHOLD register. This bit will never set unless the ANACK bit is set in the CNTL register. The SAPDx bit in the CNTL resister must be set to enable the threshold error to trigger the auto power down sequence. RTE is a write once to clear bit. Clearing this bit also resets the internal counter for consecutive NACK's being transmitted for a given byte.

Table continues on the next page...

EMVSIMx_RX_STATUS field descriptions (continued)

Field	Description
	0 Number of NACKs generated by the receiver is less than the value programmed in RTH[3:0] 1 Number of NACKs generated by the receiver is equal to the value programmed in RTH[3:0]
8 CWT_ERR	Character Wait Time Error Flag Used to indicate when the time between received characters is equal to or greater than the value programmed in the CHAR_WAIT register. 0 No CWT violation has occurred (default). 1 Time between two consecutive characters has exceeded the value in CHAR_WAIT.
7 CRC_OK	CRC Check OK Flag Used to indicate when the calculated 16-bit CRC value matches the expected value for the current input data stream. The value is calculated across all received characters from the point the CRC_EN bit is set in the CTRL register. The current CRC residual can be reset by three mechanisms: • Clear CRC_EN bit in CTRL register • Set XMT_EN bit in ENABLE register • Automatically by hardware when ETC flag is set at the end of a transmission. 0 Current CRC value does not match remainder. 1 Current calculated CRC value matches the expected result.
6 LRC_OK	LRC Check OK Flag Used to indicate when the calculated 8-bit LRC value is correct for the current input data stream. The value is calculated across all received characters from the point the LRC_EN bit is set in the CTRL register. The current LRC residual can be reset by three mechanisms: • Clear LRC_EN bit in CTRL register • Set XMT_EN bit in ENABLE register • Automatically by hardware when ETC flag is set at the end of a transmission. 0 Current LRC value does not match remainder. 1 Current calculated LRC value matches the expected result (i.e. zero).
5 RDTF	Receive Data Threshold Interrupt Flag Interrupt flag asserted when total bytes in the Receive FIFO equal or is greater than the programmed receive threshold RDT[3:0]. The RDTF flag will be set any time the number of unread bytes in the receive FIFO is equal to or greater than the value set by RDT[3:0]. The flag can be cleared by reading enough bytes out of the receive FIFO so as to bring the number of bytes left in the FIFO below the RDT[3:0] level. Another way to clear the flag is to set the RDT[3:0] level higher than the number of unread bytes currently in the FIFO. The RDTF flag will create an interrupt if the RDT_IM bit in the INT_MASK register is cleared. 0 Number of unread bytes in receive FIFO less than the value set by RDT[3:0] (default). 1 Number of unread bytes in receive FIFO greater or than equal to value set by RDT[3:0].
4 RX_DATA	Receive Data Interrupt Flag Interrupt asserted when a new data byte is received and entered into the Receive FIFO. 0 No new byte is received 1 New byte is received ans stored in Receive FIFO
3–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

EMVSIMx_RX_STATUS field descriptions (continued)

Field	Description
0 RFO	Receive FIFO Overflow Flag Used to indicate that the EMV SIM was unable to store received data due to already unread bytes in the FIFO (FIFO full or almost full). It does not necessarily indicate that data has been lost. If the ONACK control bit in the CNTL register is set, there will be a NACK pulse generated on bytes that would otherwise cause a loss of data due to a full FIFO. These bytes should be retransmitted by the Smart Card which implies that no data has actually been lost. In this case, the RFO flag is just an indicator that this situation has occurred which may be helpful in system debug. For the case where ONACK is not set, a set RFO flag does indicate a loss of data since all bytes received with the OEF flag set will indeed be lost (including the byte that caused the bit to be set). The RFO flag will cause an interrupt if the RFO_IM bit in the INT_MASK register. The RFO flag is a write-one-to-clear bit. 0 No overrun error has occurred (default) 1 A byte was received when the received FIFO was already full

14.6.10 Transmitter Status Register (EMVSIMx_TX_STATUS)

Address: Base address + 24h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TX_CNT								0				TX_RPTR			
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						GPCNT1_TO	GPCNT0_TO	TDTF	TFF	TCF	ETCF	TFE	0		TNTE
W							w1c	w1c		w1c	w1c	w1c	w1c			w1c
Reset	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0

EMVSIMx_TX_STATUS field descriptions

Field	Description
31–24 TX_CNT	Transmit FIFO Byte Count These bits indicate the number of bytes stored in the transmit FIFO. 0 FIFO is emtpy >0 Total bytes in FIFO
23–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–16 TX_RPTR	Transmit FIFO Read Pointer Value of the read pointer of transmit FIFO.
15–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 GPCNT1_TO	General Purpose Counter 1 Timeout Flag Used to indicate when the General Purpose Counter 1 has reached the GPCNT1_VAL value in the GPCNT_VAL register. 0 GPCNT1_VAL time not reached, or bit has been cleared. (default) 1 General Purpose counter has reached the GPCNT1_VAL value
8 GPCNT0_TO	General Purpose Counter 0 Timeout Flag Used to indicate when the General Purpose Counter 0 has reached the GPCNT0_VAL value in the GPCNT_VAL register. 0 GPCNT0_VAL time not reached, or bit has been cleared. (default) 1 General Purpose counter has reached the GPCNT0_VAL value
7 TDTF	Transmit Data Threshold Flag Interrupt flag asserted when total bytes in the Transmit FIFO is less than or equal to the programmed transmit data threshold TDT[3:0]. The TDTF flag will be set any time the number of bytes in the transmit FIFO is less than or equal to the value set by TDT[3:0]. The flag can be cleared by writing enough bytes into the transmit FIFO so as to take the number of bytes in the FIFO above the TDT[3:0] level. Another way to clear the flag is to set the TDT[3:0] level lower than the number of bytes currently in the FIFO. The TDTF flag will create an interrupt if the TDT_IM bit in the INT_MASK register is cleared. 0 Number of bytes in FIFO is greater than TDT[3:0], or bit has been cleared 1 Number of bytes in FIFO is less than or equal to TDT[3:0] (default)
6 TFF	Transmit FIFO Full Flag Used to indicate when the Transmit FIFO has been written with maximum number of bytes that it can store. Subsequent writes may get ignored. The TFF bit can only be cleared by writing 1 to this bit or setting the FLUSH_XMT or SOFT_RESET bit in the RESET_CNTL register. 0 Transmit FIFO Full condition has *not* occurred (default) 1 A Transmit FIFO Full condition has occurred
5 TCF	Transmit Complete Flag Used to indicate whether the EMV SIM transmitter is ready for a new transmission. The TC flag becomes set when the guard time has expired after the last byte in the transmit FIFO (if XMT_CRC_LRC = 0) or the LRC or CRC byte (if XMT_CRC_LRC bit = 1) has been transmitted. The TC flag will create an interrupt if TC_IM in the INT_MASK register is low. The TC bit is a write-one-to-clear bit.

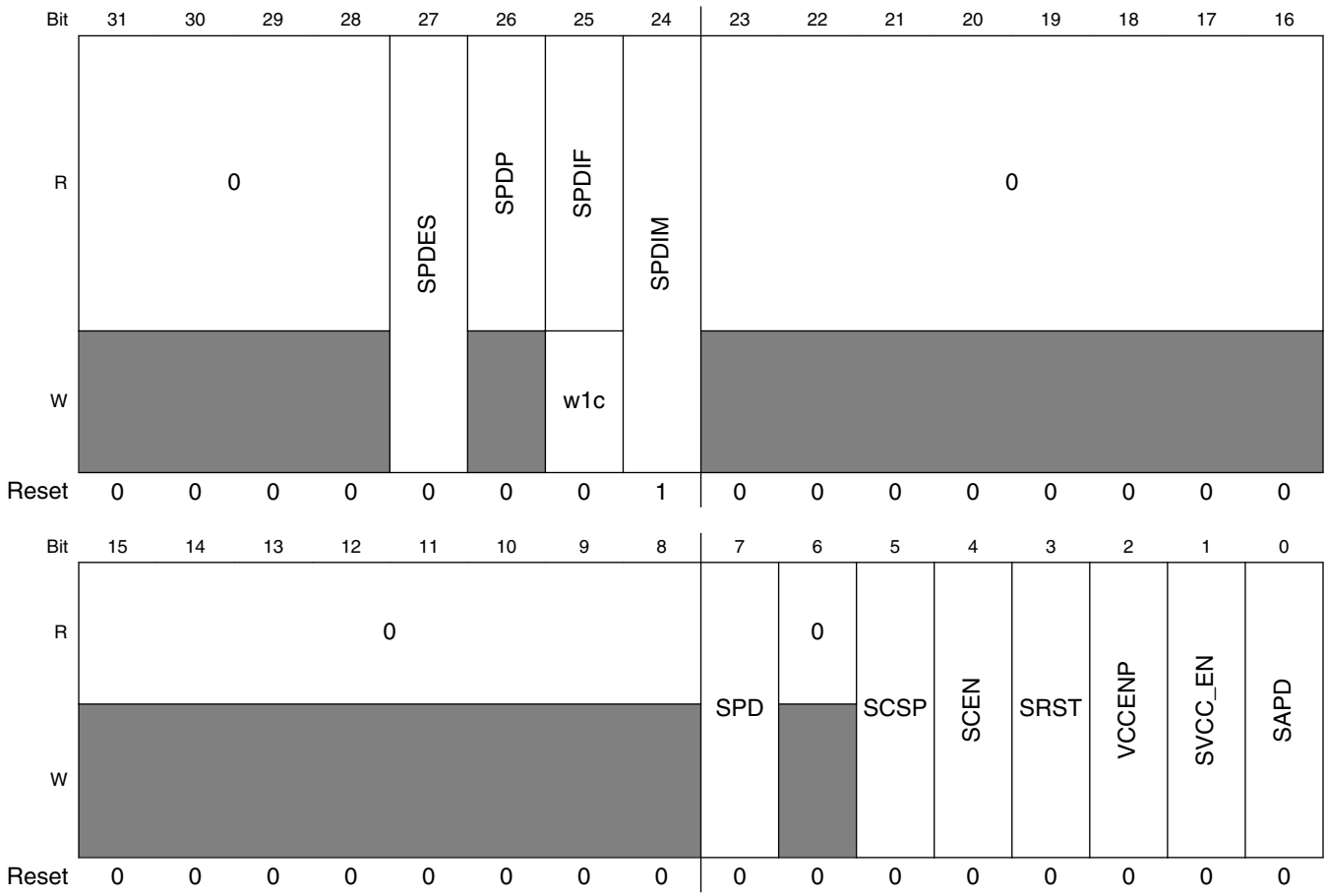
Table continues on the next page...

EMVSIMx_TX_STATUS field descriptions (continued)

Field	Description
	0 Transmit pending or in progress 1 Transmit complete (default)
4 ETCF	Early Transmit Complete Flag Used to indicate that the EMV SIM transmitter has finished sending the last byte in transmit FIFO (if XMT_CRC_LRC = 0) or finished sending the LRC or CRC byte (if XMT_CRC_LRC = 1). This bit differs from the TC bit in that it is set before the guard time of the last byte has elapsed. The ETC flag will create an interrupt if ETC_IM in the INT_MASK register is low. The ETC bit is a write-one-to-clear bit. 0 Transmit pending or in progress 1 Transmit complete (default)
3 TFE	Transmit FIFO Empty Flag Used to indicate that the EMV SIM transmit FIFO has been emptied. This bit will be set when the last byte in the transmit FIFO has been transferred out of the EMV SIM transmitter to the Smart Card successfully. TFE will also be set when the TNTE flag is set. The TFE flag will create an interrupt if TFE_IM in the INT_MASK register is low. The TFE bit is a write-one-to-clear bit. 0 Transmit FIFO is not empty 1 Transmit FIFO is empty (default)
2–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 TNTE	Transmit NACK Threshold Error Flag Used to indicate the transmit NACK threshold has been reached. When TNTE is high, no further transmissions will be done until the TNTE flag is cleared. Any data transmissions still pending in the transmit FIFO will be aborted, and the TC, ETC, and TFE flags will be set. The TNTE flag will create an interrupt if TNACK_IM in the INT_MASK register is low. The TNTE bit is a write-one-to-clear bit. 0 Transmit NACK threshold has not been reached (default) 1 Transmit NACK threshold reached; transmitter frozen

14.6.11 Port Control and Status Register (EMVSIMx_PCSR)

Address: Base address + 28h offset



EMVSIMx_PCSR field descriptions

Field	Description
31–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 SPDES	SIM Presence Detect Edge Select Controls which edge of the Smart Card Presence Detect pin is used to detect the presence of the Smart Card. 0 Falling edge on the pin (default) 1 Rising edge on the pin
26 SPDP	Smart Card Presence Detect Pin Status This bit reflects the state of the Smart Card Presence Detect pin. It is not a latched register bit, but instead a synchronized version of the state of the pin itself. 0 SIM Presence Detect pin is logic low 1 SIM Presence Detectpin is logic high

Table continues on the next page...

EMVSIMx_PCSR field descriptions (continued)

Field	Description
25 SPDIF	Smart Card Presence Detect Interrupt Flag Status flag to indicate that an insertion or removal of a Smart Card has been detected on port. Can create an interrupt to the MCU if SPDIM is low and/or initiate the auto power down sequence if SAPD bit is '1'. Write a '1' to this bit to clear. 0 No insertion or removal of Smart Card detected on Port (default) 1 Insertion or removal of Smart Card detected on Port
24 SPDIM	Smart Card Presence Detect Interrupt Mask Interrupt mask for the presence detect interrupt flag (SPDIF). 0 SIM presence detect interrupt is enabled 1 SIM presence detect interrupt is masked (default)
23–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SPD	Auto Power Down Control Writing a '1' to this location will start the autopower down sequence provided the auto power down feature is enabled by writing '1' to SAPD bit of this register. The SPD bit will autoclear when power down is complete. 0 No effect (default) 1 Start Auto Powerdown or Power Down is in progress
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SCSP	Smart Card Clock Stop Polarity Used to control the polarity of the idle EMV SIM clock when the clock is disabled by SCEN. 0 Clock is logic 0 when stopped by SCEN 1 Clock is logic 1 when stopped by SCEN
4 SCEN	Clock Enable for Smart Card Used to enable/disable the clock to the Smart Card. It can be forced low by hardware during the auto power down sequence. 0 Smart Card Clock Disabled 1 Smart Card Clock Enabled
3 SRST	Reset to Smart Card Used to control state of reset line to the Smart Card. It can be forced low by hardware during the auto power down sequence. Smart Card reset signals are active low. 0 Smart Card Reset is asserted (default) 1 Smart Card Reset is de-asserted
2 VCCENP	VCC Enable Polarity Control Used to control the polarity of the SVEN output pad via the SVCC_EN bit. When set to '1', this bit will invert the value of SVCC_EN bit of this register and output to SVEN output pad of device.

Table continues on the next page...

EMVSIMx_PCSR field descriptions (continued)

Field	Description
	0 VCC_EN is active high. Polarity of SVCC_EN is unchanged. 1 VCC_EN is active low. Polarity of SVCC_EN is inverted.
1 SVCC_EN	Vcc Enable for Smart Card Used to control the state of the SVEN pin on Smart Card port. The SVEN pin controls the Smart Card Vcc enable in the power management chip. It can be forced low by hardware during the auto power down sequence. This bit is XORed with the VCCENP bit to control the polarity of the SVEN output pad. 0 Smart Card Voltage disabled (default) 1 Smart Card Voltage enabled
0 SAPD	Auto Power Down Enable Used to enable/disable the auto power down feature. This bit must be set prior to the auto power conditions are met. This bit will be return to 0 at the end of the auto power down sequence. This bit controls the auto power down function which can be initiated by: • Setting the SPD bit in this register by software • Assertion of the RTE bit in the RX_STATUS register • Assertion of the SPDIF bit when the interrupt is used for detecting card removal Manual Power Down via software by writing to the SCEN, SRST, and VCC_EN can still be carried out irrespective of the value of this bit. 0 Auto power down disabled (default) 1 Auto power down enabled

14.6.12 Receive Data Read Buffer (EMVSIMx_RX_BUF)

Address: Base address + 2Ch offset

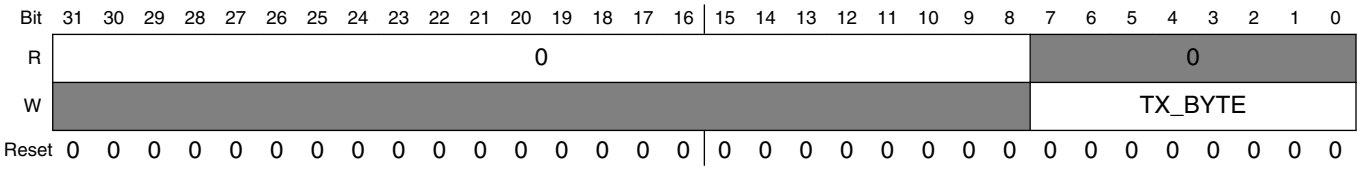
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																								RX_BYTE							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EMVSIMx_RX_BUF field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
RX_BYTE	Receive Data Byte Read Provides the byte value from the top of the receive FIFO. Each read access to this register will increment the read pointer of the Receive FIFO.

14.6.13 Transmit Data Buffer (EMVSIMx_TX_BUF)

Address: Base address + 30h offset

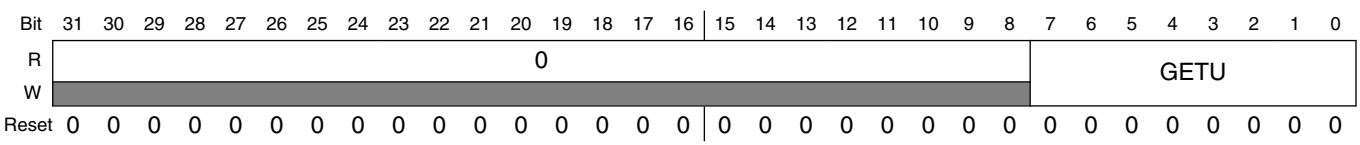


EMVSIMx_TX_BUF field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TX_BYTE	Transmit Data Byte Write to this register to fill the transmit FIFO with the bytes to be transmitted. The Tx FIFO can be written to only when the transmitter is enabled. Writing to this register while transmitter is disabled will lead to ignoring of all write accesses to this register. Reads to this register will return zeros. **NOTE:** Writing more data to the transmit FIFO than it can hold, will cause a Transmit FIFO Full (TFF) error.

14.6.14 Transmitter Guard ETU Value Register (EMVSIMx_TX_GETU)

Address: Base address + 34h offset

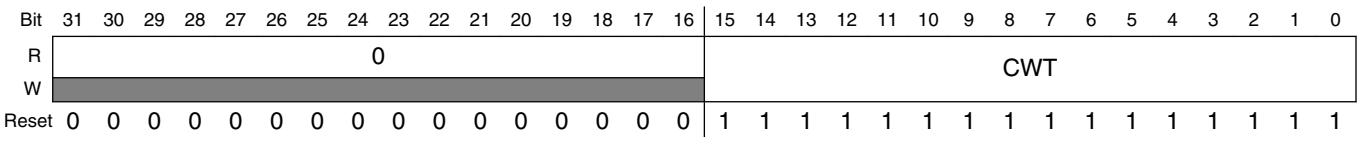


EMVSIMx_TX_GETU field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
GETU	Transmitter Guard Time Value in ETU Used to control the number of additional Elementary Time Units (ETUs) inserted between bytes transmitted by the EMV SIM transmitter. An ETU is equivalent to one bit time at the given baud rate (for example, the length of a START bit). The guard time has no effect on the EMV SIM receiver. A value of 0x00 inserts no additional ETUs, while a value of 0xFE inserts 254 additional ETUs. A value of 0xFF subtracts one ETU by reducing the number of STOP bits from two to one. 0 no additional ETUs inserted (default) 1 1 additional ETU inserted ... 254 254 additional ETUs inserted 255 Subtracts one ETU by reducing the number of STOP bits from two to one

14.6.15 Character Wait Time Value Register (EMVSIMx_CWT_VAL)

Address: Base address + 38h offset

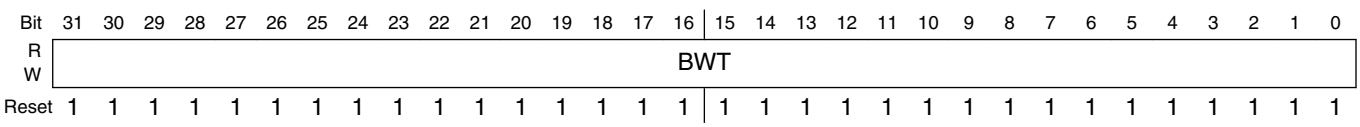


EMVSIMx_CWT_VAL field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CWT	Character Wait Time Value The value written to this register will specify the number of ETU times allowed between characters. Default is 0xFFFF

14.6.16 Block Wait Time Value Register (EMVSIMx_BWT_VAL)

Address: Base address + 3Ch offset

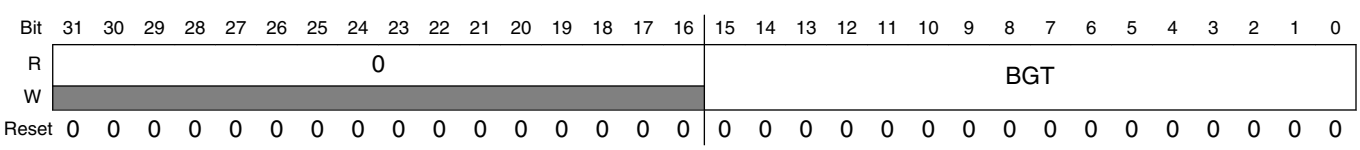


EMVSIMx_BWT_VAL field descriptions

Field	Description
BWT	Block Wait Time Value The time from START bit of last byte sent from the EMV SIM module to the START bit of the first byte sent from the Smart Card must be less than the value in this register. If it is not, then the BWT_TO flag is set.

14.6.17 Block Guard Time Value Register (EMVSIMx_BGT_VAL)

Address: Base address + 40h offset



EMVSIMx_BGT_VAL field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
BGT	Block Guard Time Value The value in this register is the block guard time. Time from START bit of last byte sent from the EMV SIM module to the START bit of the first byte sent from the SmartCard must be greater than this value. If it is not, then the BGT flag will be set.

14.6.18 General Purpose Counter 0 Timeout Value Register (EMVSIMx_GPCNT0_VAL)

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																GPCNT0																
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

EMVSIMx_GPCNT0_VAL field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
GPCNT0	General Purpose Counter 0 Timeout Value The value written to this register will be used to compare to the general purpose counter 0 in the EMV SIM module. Once the General purpose counter reaches this value, the GPCNT0 flag in the TX_STATUS register will be set. This counter is intended to be used for any events that must be monitored for duration based on the card clock, receiver sample rate, or ETU rate (transmit clock). Example: ATR arrival time and ATR duration.

14.6.19 General Purpose Counter 1 Timeout Value (EMVSIMx_GPCNT1_VAL)

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																GPCNT1																
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

EMVSIMx_GPCNT1_VAL field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
GPCNT1	General Purpose Counter 1 Timeout Value The value written to this register will be used to compare to the general purpose counter in the EMV SIM module. Once the General purpose counter reaches this value, the GPCNT1 flag in the XMT_STATUS register will be set. This counter is intended to be used for any events that must be monitored for duration based on the card clock, receiver sample rate, or ETU rate (transmit clock). Example: ATR arrival time and ATR duration.

14.7 Functional Description

14.7.1 Initialization

14.7.1.1 Configuring EMV SIM

The first operation that must be done is to configure the EMV SIM interface signals to allow the EMV SIM Transmit or Receive operation to commence. The following steps describe a typical way to enable the Smart Card:

- Configure prescaler by programming the desired value in PRSC register to generate the card clock. The card clock frequency is calculated as:
  - f_{SIM} / (PRSC[7:0]); where f_{SIM} is the EMV SIM Clock Frequency
  - Prescaler value must be set such that the maximum card clock frequency specification set by EMV & ISO-7816 specifications are not violated
- Configure the DIVISOR_VALUE in the Baud Rate Divisor Register. This value determines the ETU period to be used by the transmitter or receiver
- Enable power to the Smart Card by setting the VCC_EN bit in PCSR register
- Enable the Smart Card clock by setting the SCEN bit in PCSR register
- Release Smart Card reset using the SRST bit in PCSR register to bring the Smart Card out of reset
- Program the power down functionality i.e. if Auto Power Down of Smart Card is required or not
- Select the Data Format type for the EMV SIM
  - Either set the IC bit to the desired data format (inverse or direct) OR
  - Set the ICM bit to allow automatic detection of Initial Character. IC bit will be updated to desired data format upon valid Initial Character detection

- Configure if the EMV SIM module is enabled to run in STOP and DOZE modes by setting the STOP_EN and DOZE_EN bits as desired
- Enable DMA request generation for Transmitter or Receiver as desired

Once the above configuration is complete, the receiver or transmitter can be configured depending on the operation to be carried out on the Smart Card.

NOTE

XMT_EN and RCV_EN bits should not be changed in the same write cycle. Software should attempt separate write accesses to CTRL register for updating these bits, one by one. For example, if XMT_EN = 1 and RCV_EN = 0 and in order to enable the receiver, first write to CTRL register should make XMT_EN = 0 and second write to CTRL register should make RCV_EN = 1.

14.7.1.2 Configuring Receiver

The following is a list of configurations that need to be performed (after the "Configuring EMV SIM" step) in order to configure the EMV SIM receiver for operation:

- Enable NACK generation capability
  - To generate NACK on parity errors and invalid initial characters, set the ANACK bit in CTRL register
  - To generate NACK on receiver FIFO overflow error, set the ONACK bit in CTRL register
- Configure the desired threshold for NACK generation and data threshold in the RX_THD register. These are the thresholds at which the RTE and RDTF flags will be set, respectively.
  - If auto power down is desired on RTE flag assertion, then write to SAPD bit PCSR to initiate the Smart Card power down
- Configure the timeout values for Character Wait Time Counter, the Block Wait Timer Counter and the Block Guard Time Counter
- Enable necessary interrupts by clearing respective bits in the INT_MASK register
- Configure the General Purpose Counter as needed and select desired clock source (GPCNT0/1_CLKSEL in the CLKCFG register)
- Enable Receiver by setting RCV_EN bit to 1 in CTRL register
- Receiver will enter Initial Character Detection mode if ICM bit is set or enter normal receive mode if ICM bit is not set
- Enable LRC or CRC as desired

14.7.1.3 Configuring Transmitter

The following is a list of configurations that need to be performed (after the "Configuring EMV SIM" step) in order to configure the EMV SIM transmitter for operation:

- Select desired re-transmission threshold for NACKed characters in TX_THD register
  - This is the threshold at which the TNTE flag will be set by using the TNACK_THD[3:0] bits
- Configure the desired Guard Time between the characters transmitted by writing to the GETU[7:0] bits in TX_GETU register
- Configure the desired data threshold in TX_THD register.
- Enable necessary interrupts by clearing respective bits in the INT_MASK register
- Configure the General Purpose Counter as needed and select desired clock source (GPCNT0/1_CLKSEL in the CLKCFG register)
- Enable Transmitter by setting the XMT_EN in the CTRL register
- Enable LRC & CRC as desired
- Program bytes to be transmitted into the Tx FIFO. If DMA is enabled, DMA request will get asserted automatically.

NOTE

Tx FIFO can be written to after the transmitter is enabled (XMT_EN = 1). Writing to Tx FIFO while transmitter is disabled will not be successful and write access will be ignored.

NOTE

The Transmit and Receive operations are mutually exclusive and should not be enabled together.

14.7.2 Smart Card Interface and Control

The EMV SIM module controls the all the interface signals for the Smart Card. The Smart Card clock is generated using the prescaler (PRSC register) and the ETU periods are generated using the baud rate divisor (DIVISOR register). The Smart Card Reset and Voltage Enable are controlled by software through the PCSR register. The message bytes are transmitted and received on the Card IO device pin which is used in open drain configuration.

14.7.2.1 Smart Card Presence Detect

The Smart Card Presence Detect pin (SIMPD) allows for detection of the insertion or removal of a Smart Card. The SPDES bit in the PCSR register allows the software to configure which edge of the SIMPD pin will cause a presence detect interrupt flag to be asserted.

An internal pull-down is required on SIMPD pin to allow a high to low transition on this pin when the Smart Card is removed.

14.7.2.2 Powering Up

The first step to communicating with a Smart Card is to provide power and a clock signal to the card. Once the card is detected as present (using the presence detect features, or some other method), the Smart Card should be powered up according to the power up sequence specified in the ISO 7816-3 specification.

1. Apply voltage to the Smart Card by setting the VCC_EN bit in the PCSR register.
2. Select the appropriate clock frequency for the Smart Card by programming the CLKCFG and DIVISOR registers.
3. Enable the clock to the Smart Card by setting the SCEN bit in the PCSR register.
4. Remove the card from reset by setting the SRST bit in the PCSR register.

14.7.2.3 Automatic Power Down

Smart Cards require a proper sequence of operations to be followed to allow them to be powered down. The power down sequence as per ISO 7816 specifications is:

- Smart Card Reset (RST) pin transitions from high to low
- Smart Card Clock (CLK) is turned off to low
- Smart Card IO (IO) is transitioned from high impedance to low
- Smart Card Voltage Enable (VCC_EN) to turned off (pin driven low)

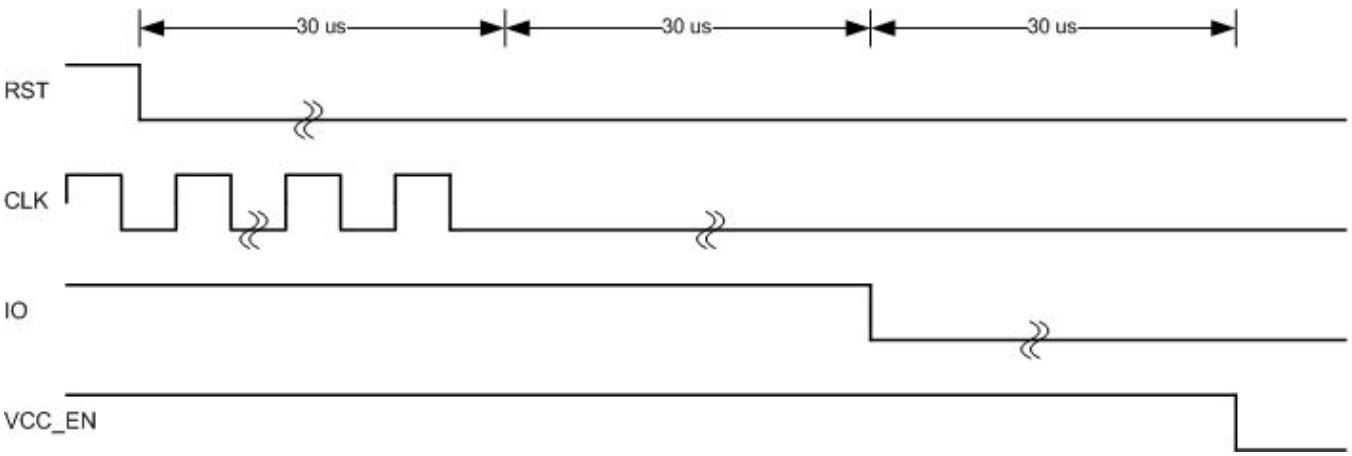


Figure 14-2. Auto Power Down Sequence

These operations can be done manually by software by writing to appropriate bits (SCEN, SRST and VCC_EN) in the PCSR register or automatically by hardware under the following conditions provided the auto power down feature is enabled in the PCSR register.

- Setting the SPD bit in PSCR register by software
- Assertion of RTE bit in RX_STATUS register
- Assertion of SPDIF bit in PCSR register when configured to detect card removal

NOTE

The SPDES bit in PCSR register determines which edge transition of the Smart Card Presence Detect pin is used for Smart Card presence detection. Presence detection can be used to determine if the card has been inserted or removed. The occurrence of the edge specified by SPDES bit will cause the following: SPDIF to be set; if the SPDIM mask is clear, an interrupt to the CPU; and if SAPD in the PCSR register is set, an auto power down sequence to begin. If Smart card insertion is expected, SAPD can be set low to avoid the auto power down sequence from falsely triggering. The bit SPDP can be used to determine the current state of the Smart Card Presence Detect pin.

NOTE

The auto power down operation requires a low frequency clock (usually 32 kHz clock source) to be active on the device.

The auto power down sequence will clear the following register bits upon power down sequence completion:

- SPD bit in PCSR register
- SCEN bit in PCSR register

- SRST bit in PCSR register
- VCC_EN bit in PCSR register
- XMT_EN and RCV_EN bits in CTRL register

14.7.3 EMV SIM Receiver

The EMV SIM Receiver is designed to receive all message bytes and store valid bytes into the receive FIFO. It performs checks on the incoming messages and indicates the transmitter to enable NACK insertion for each message. In addition, the receiver automatically detects NACKs received when the transmitter is sending out message bytes.

14.7.3.1 Data Sampling

The incoming data is internally synchronized and sampled using a 16x oversampled clock, called the EMV SIM logic clock. The frequency of this clock is 16 times the ETU Clock (1/ETU Period). The whole receiver works on this EMV SIM logic clock.

14.7.3.2 Initial Character Detection

The EMV SIM module supports the detection of the Initial Character and determine the data format when it is placed in the initial character mode by setting the ICM bit in CTRL register. When placed in this mode, the EMV SIM module samples in the input line and waits for one of the initial characters to be detected. Once detected, the ICM bit is cleared and the data format bit, IC bit, is set to appropriate value depending on the data format detected using the initial character.

The two possible data formats are inverse convention and direct convention. Figures below illustrate the differences between the two formats. Essentially, inverse convention differs from direct convention in that the order of the bits is flipped (i.e. MSB for LSB) and the data bits and parity bit are logically inverted. When receiving inverse convention data, the transformation of the data back to direct convention format is done in hardware, including the inversion of the data and parity bits.

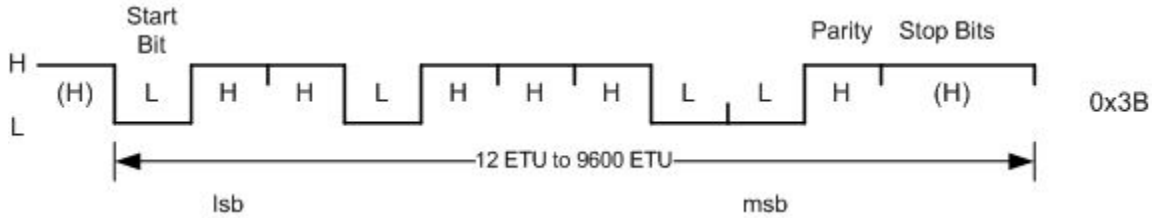


Figure 14-3. Initial Character in Direct Conversion

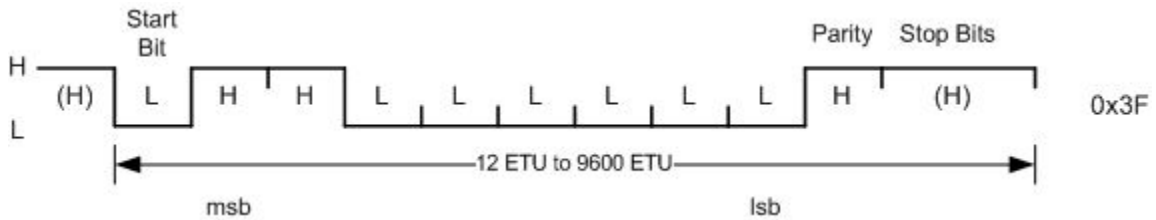


Figure 14-4. Initial Character in Inverse Conversion

To place the EMV SIM into initial character mode, set the ICM bit in the CTRL register. Once a valid initial character is received, the IC bit in the CTRL register will be set accordingly by the hardware, and the ICM bit will be cleared. Software can read the state of the IC bit to determine which mode the EMV SIM is currently using. The 0x3B (as decoded by direct convention) with parity bit high will cause direct convention to be used (IC set to logic 0), whereas a 0x3F (as decoded by inverse convention) with parity bit high will cause inverse convention to be used (IC set to logic 1).

The receiver remains in initial character mode until disabled by software or a valid initial character is detected. Upon detection of valid character, the receiver enters data receive mode. It is recommended to use one of the general purpose counter as a timeout during initial character mode.

When the receiver is in initial character mode, all received bytes will continue to be placed into the receive FIFO whether they be valid initial characters or not. If a valid initial character is received that causes the data format being used to change, all subsequent bytes will be decoded with that format before being placed into the FIFO, including the initial character byte itself. That is, if the IC bit is low, and the correct initial character for setting inverse convention is received, that character and all subsequently received characters will be stored in the FIFO after having been decoded using inverse convention (for example, the initial character will be stored as 0x3F).

NOTE

There is a condition where a parity error in the initial byte for direct convention (0x3B) could be decoded as what appears to be a valid initial character for inverse convention (that is, 0x3F). The EMV SIM module will not recognize this as a valid

initial character for inverse convention and will set the PEF flag.

14.7.3.3 Receiver Diagnostics

The EMV SIM module performs diagnostic checks on the incoming messages. These checks include parity check and message frame checks.

Parity Error Detection: The receiver is responsible for detecting parity errors in the received data. Data is always transmitted with even parity, except when in inverse convention mode. In inverse convention mode, all data bits and the parity bit are complemented making the data appear to be odd parity. The parity bit is defined as the 10th bit of the received data. The parity of the 2nd through 10th received bits is calculated by the receiver parity logic. This logic determines if the parity of the 9 received bits is correct.

When a parity error is detected on a given byte, the parity error flag (PEF bit) is asserted in the RX_STATUS register if NACK generation on errors is disabled (ANACK = 0 in CTRL register). The parity error flag does not generate an interrupt to the system, however, the module can cause the transmitter to transmit a NACK (if ANACK = 1 in CTRL register) without software needed to enable transmitter for this and request re-transmission of the current byte.

Framing Error Detection: The receiver is responsible for detecting framing errors in the received data. Data is always transmitted with 2 STOP bits, except when in 11 ETU mode, where there is one STOP bit. The STOP bits are defined as high state at the 11th and 12th bit positions in the received data. In either mode, the receiver expects to see a high at the 11th bit position in the received data. If a low is detected at the 11th bit position, the receiver indicates a framing error for that byte.

When a framing error is detected on a given byte, the FEF bit is set in the RX_STATUS register. A framing error cannot cause an interrupt, nor can it create a NACK pulse to the Smart Card asking for a retransmission of the corrupted data.

LRC & CRC: The receiver also checks the LRC or CRC on the incoming block of bytes (if LRC or CRC is enabled). The result of the CRC or LRC check is updated in the RX_STATUS register once the complete message block is received.

14.7.3.4 NACK Detection

During transmission of message bytes, the receiver automatically checks for NACKs sent by the Smart Card. The software does not need to enable receiver for this. The NACK is detected by sampling the IO line at 11 ETUs after the falling edge of the START bit. Once a NACK is detected, the transmitter is signaled and the transmitter will retransmit the current byte after two ETU time (as per ISO 7816 specification).

14.7.3.5 Using EMV SIM Receiver with "T=1" Smart Cards

The EMV SIM module provides hardware support for “T=1” type Smart Cards. These type of cards present several requirements above and beyond the standard “T=0” cards. The features provided to meet the requirements that pertain to the EMV SIM receiver are as follows:

- 11 ETU Characters
  - “T=1” cards can transmit with character lengths of 11 ETUs (that is, one STOP bit). The EMV SIM module provides the RCVR11 bit in the CTRL register in order to configure the receiver state machine to accept 11 ETU characters.
- Character Wait Time Counter
  - The character waiting time (CWT) is defined as the time between the start bits of two consecutive characters received from the Smart Card. The value of CWT can range from 12 ETU to 32779 ETU. The EMV SIM module provides a 16-bit counter with programmable comparator clocked at the ETU bit rate to identify when the CWT has been exceeded by the Smart Card.
- Block Wait Time
  - The block waiting time (BWT) is defined as the maximum time between the start bits of the last character of a transmitted block and the first character of the next received block. The block wait time must not exceed a value that is programmable. The EMV SIM module provides a 32-bit Block Wait Timer that can be used to identify when the BWT has been violated.
- Block Guard Time
  - The block guard time (BGT) is defined as the minimum delay between the start bits of the last character of a transmitted block and the first character of the next received block. The block guard must be greater than a value that is programmable. The EMV SIM module provides a 16-bit Block Guard Timer that can be used to identify when the BGT has been violated.
- Error Detection Code
  - “T=1” cards can specify LRC or CRC error detection codes to be used. The EMV SIM module provides hardware support for both the LRC and CRC operations.

14.7.4 EMV SIM Transmitter

The transmitter is responsible for reading the message bytes from the Tx FIFO, adding the LRC or CRC at the end of the message block and re-transmit any byte on reception of a NACK. It assists the receive operation by inserting NACKs when directed by the receiver.

14.7.4.1 Message Transmission

The transmitter operation does not start until the Tx FIFO is empty. Software can configure the EMV SIM module and enable the transmitter but unless a byte is written into the Tx FIFO, no transmission will start. Clearing the XMT_EN bit while the transmitter is in operation, will halt any transmission in progress, flush the transmit FIFO. Refer Message Handling section for more details on using the Tx FIFO.

The transmitter continuously transmits the messages from the Tx FIFO. Upon NACK detection, the transmitter resends the same byte until the Transmitter NACK Threshold limit is reached. When TNTE is set, the EMV SIM transmitter is halted, and all pending transfers are aborted, and the TC, ETC, AND TFE flags are set. All bytes remaining in the transmit FIFO are lost. There is no way to restart the transmission on the next byte in the FIFO. The transmitter remains frozen until TNTE is cleared by software. The only way to clear TNTE is to write a 1 to the TNTE bit in the TX_STATUS register. It is possible to disable the detection of NACKs from the Smart Card by setting TNCK_THD[3:0] to 0x0. By setting TNCK_THD[3:0] to 0x1, it is possible to disable all retransmissions while still setting TNTE on the first NACK received. In general, TNTE is set on the NACK that causes the threshold set by TNCK_THD[3:0] to be reached. This final NACK will not cause a retransmission, whereas all previous NACKs will cause a retransmission.

If all bytes are successfully transmitted and the Tx FIFO is empty, the transmitter send the check byte (i.e. LRC or CRC) if the XMT_CRC_LRC and LRC_EN or CRC_EN bits are set. The transmitter sends each byte to the LRC or CRC block to allow the check byte generation.

At the end of each byte, the transmitter inserts the programmed Guard ETUs. Refer to Protocol Timers section for more details on Guard Time Counter.

14.7.4.2 NACK Insertion

The transmitter is responsible for driving the output low during the STOP bit time to signify an error was detected in the received data from the Smart Card. This logic responds to a NACK request generated by the receiver block.

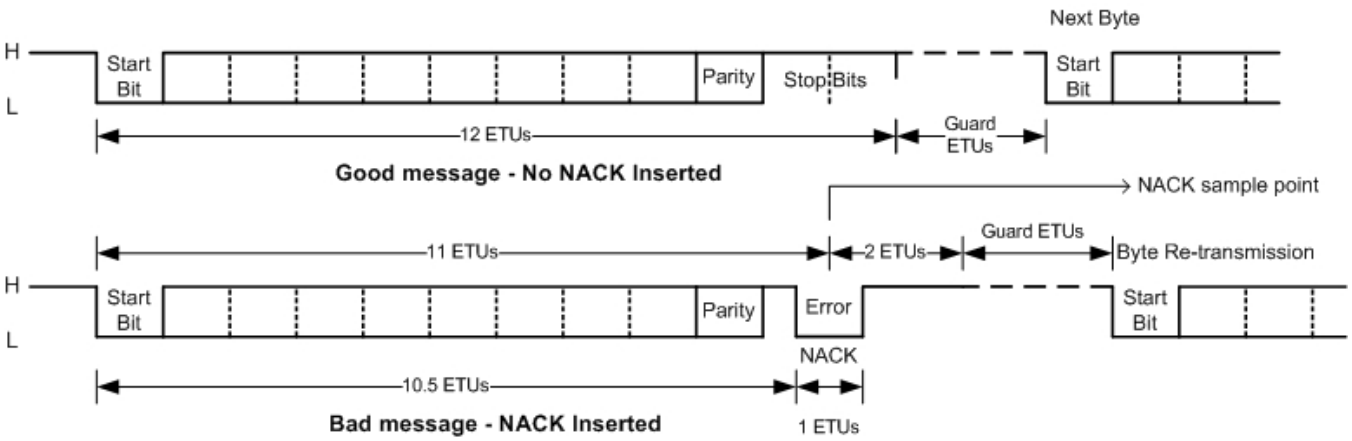


Figure 14-5. NACK Insertion

The NACK generation logic is also responsible for keeping track of the number of NACKs received during a transmit operation. The receiver detects NACKs generated by the Smart Card, and reports them to the transmit NACK logic. Once the number of detected NACKs has reached the programmed threshold (TNCK_THD[3:0]), an interrupt flag (TNTE) is generated, the transmit FIFO is flushed, and the transmitter is disabled.

14.7.4.3 Using EMV SIM Transmitter with "T=1" Smart Cards

The EMV SIM module provides hardware support for “T=1” type Smart Cards. These type of cards present several requirements above and beyond the standard “T=0” cards. The features provided to meet the requirements that pertain to the EMV SIM transmitter are as follows:

- 11 ETU Characters
  - The EMV SIM module transmitter has a programmable guard time register that allows the programmer to specify the number of ETUs between character transmissions. Programming a value of 255 (0xFF) in the GETU[7:0] bits in the GUARD_CNTL register will set the number of ETUs per character transmitted to 11.
- Character Waiting Time
  - The character waiting time (CWT) is defined as the time between the start bits of two consecutive characters. The value of CWT can range from 12 ETU to 32779 ETU. The time between transmitted characters is controlled by the

programmable guard time in the GETU register. However, the time between the last byte in the transmit FIFO, and the next transmitted byte can be largely affected by software response time to the transmit interrupts. The EMV SIM transmitter provides a Transmit FIFO threshold (TDTF) interrupt to signal the system when the expected number of characters have been transmitted from the transmit FIFO. The minimum CWT is achieved only if the software can respond to the TDTF interrupt and write new data to the transmit FIFO before the last character in the Transmit FIFO has been sent.

- Block Waiting Time
  - The block waiting time (BWT) is defined as the maximum time between the start bits of the last character of a transmitted block and the first character of the next received block. The value of BWT is always greater than 1800 ETU. The EMV SIM transmitter provides two General Purpose Counter(s) either of which can be used to track the BWT. The BWT is purely determined by software response time to the transmit interrupts.
- Block Guard Time
  - The block guard time (BGT) is defined as the minimum delay between the start bits of the last character of a transmitted block and the first character of the next received block. The value of BGT is 22 ETU. The EMV SIM module supports the BGT by providing the ability to generate an interrupt when the last byte is received, and transmitting within 2 ETU after the XMT_EN bit is set. The BGT will be determined by the speed at which the software can react to an interrupt and enable the transmitter.
- Error Detection Code
  - “T=1” cards can specify LRC or CRC error detection codes to be used. The EMV SIM module provides hardware support for both the LRC and CRC operation.

14.7.5 LRC and CRC

14.7.5.1 LRC Block

The EMV SIM module provides an 8-bit Linear Redundancy Check (LRC) generator / checker. The block is provided for use with T=1 Smart Cards that support LRC. This block can be enabled through the LRC_EN bit in the CNTL register. This block performs an 8-bit exclusive-OR on all received or transmitted characters. At the end of the reception of a block of characters, the result is expected to be 00. If so, the LRC_OK bit is set in the RX_STATUS register. During transmission, the LRC block Exclusive-ORs

each character that is transmitted with the current value of the LRC. If the XMT_CRC_LRC bit in the CTRL register is set, the LRC value will automatically be sent by the transmitter as the final character when the transmit FIFO empties.

The LRC value can be reset in multiple ways. Clearing the LCR_EN bit in the CTRL register will reset the LRC value. At the end of a transmission (either after the LRC byte is transmitted, or after the last character in the transmit FIFO is sent when XMT_CRC_LRC is clear), the LRC value is automatically reset by the EMV SIM hardware. Finally, when setting the XMT_EN bit, the EMV SIM hardware resets the LRC value.

Summary of configuration needed to use LRC block:

- Use the LRC_EN bit in CTRL register to enable the LRC block
- Use the XMT_CRC_LRC bit to enable the transmission of the LRC Character after the last character in the Transmit FIFO is sent.

14.7.5.2 CRC Block

The EMV SIM module provides an 16-bit Cyclic Redundancy Check (CRC) generator / checker. The block is provided for use with T=1 Smart Cards that support CRC. This block can be enabled through the CRC_EN bit in the CTRL register. This block performs a polynomial based check on all received or transmitted characters. The polynomial used for calculating the CRC is $G(x) = x^{16} + x^{12} + x^5 + 1$. The CRC register is initialized to all "1" before the data is shifted in. The CRC block can be configured to compute using the ISO 7816 CRC method or the 16-bit CCITT CRC method. The software can select the following bits: CRC_IN_FLIP, CRC_OUT_FLIP, and INV_CRC_VAL of the CTRL register.

- To select CCITT CRC method, set
  - CRC_IN_FLIP = 1
  - CRC_OUT_FLIP = 1
  - INV_CRC_VAL = 1
- To select normal CRC method (i.e. ISO 7816 CRC method), set
  - CRC_IN_VAL = 0
  - CRC_OUT_VAL = 0
  - INV_CRC_VAL = 1

During transmission, the CRC block updates the current value of the CRC using each character. If the XMT_CRC_LRC bit in the CTRL register is set, the CRC value will automatically be formatted as per the selected CRC method (CCITT or normal) and sent by the EMV SIM transmitter as the final two characters when the transmit FIFO empties. At the end of the reception of a block of characters, the residual from the CRC calculation is compared and the CRC_OK bit is set in the RX_STATUS register.

NOTE

Setting any other value for CRC_IN_FLIP, CRC_OUT_FLIP and INV_CRC_VAL will cause CRC value to be formatted accordingly; however the CRC will not be compliant to the standard CRC used in Smart Cards.

The CRC value can be reset/initialized in multiple ways. Clearing the CRC_EN bit in the CTRL register will reset the CRC value. At the end of a transmission (either after the CRC characters are transmitted, or after the last character in the transmit FIFO is sent when XMT_CRC_LRC is clear), the CRC value is automatically reset by the EMV SIM hardware. Finally, when setting the XMT_EN bit, the EMV SIM hardware resets the CRC value.

Summary of configuration needed to use CRC block:

- Use CRC_IN_FLIP, CRC_OUT_FLIP and INV_CRC_VAL bits in CTRL register to select the desired CRC method (CCITT or normal)
- Use the CRC_EN bit in CTRL register to enable the CRC block
- Use the XMT_CRC_LRC bit to enable the transmission of the CRC Characters after the last character in the Transmit FIFO is sent.

14.7.6 Message Handling

The EMV SIM module has FIFOs on both transmit and receive side for handling all message bytes.

14.7.6.1 Transmit FIFO

A 16-byte deep FIFO is implemented in the transmitter. The transmit FIFO can be written into by the software but reads to this FIFO will return zeros. To take care of clock synchronization, each write access will have wait states inserted by the module but this is transparent to the software. The Tx FIFO can be written to while the transmitter is enabled, however, the transmitter does not start transmitting while the FIFO is empty. A write to the Tx FIFO will initiate the transmit operation. Software can store more bytes into the Tx FIFO until the programmed threshold is reached. Software can enable DMA operation by setting TX_DMA_EN = 1 in the CTRL register. On setting this bit, a DMA request will be asserted and remain asserted till the number of bytes in FIFO reached the programmed threshold value (TDT[3:0] in the TX_THD register).

The transmit FIFO can be flushed by setting the FLSH_TX bit in the CTRL register. A transmit NACK threshold error (TNTE) will halt the transmitter, and flush the transmit FIFO. The flush operation resets the transmit read and write pointers to equal values. Everything in the transmitter block is reset by the transmit flush operation. This does not include the control registers associated with the transmitter. The Transmit data threshold flag (TDTF) does not get cleared by the FLSH_TX operation.

14.7.6.2 Receive FIFO

A 16-byte deep FIFO is implemented in the receiver. Since more than 16-bytes can be received in the FIFO, the software must ensure that the Rx FIFO is periodically read. The Rx FIFO can be read by software via normal CPU accesses on interrupt (RX_DATA interrupt flag) or via DMA by setting the DMA_RX_EN bit in CTRL register. The software cannot write to this register. A write access may terminate in a transfer error.

The Rx FIFO is loaded by the receiver when it receives an error free message byte with correct parity. The FIFO stores the message byte only. When the total bytes in the Rx FIFO equals the programmed threshold value (RDT[3:0] in RX_THD register), the RDTF bit is asserted in the RX_STATUS register. An interrupt will be asserted if the RDT_IM bit in the INT_MASK is cleared. If DMA access to RX_FIFO is enabled, the DMA request to read the Rx FIFO will be asserted when the threshold flag is set and will clear when all the bytes are read out from the Rx FIFO.

The receive FIFO can be flushed by setting the FLSH_RX bit in the CTRL register. The flush operation resets the receiver except for receiver control registers.

NOTE

Since the register bits and the receiver logic are in different clock domains, the software needs to make sure to allow clock synchronization time when sampling the RDTF bit. One proposed method could be to read one byte at a time (from Rx FIFO) when RDTF is asserted, allow a time of 3 bus clock cycles after Rx FIFO read and then check the status of the RDTF bit and repeat as necessary. Alternatively, one could set the receiver threshold one more than actually needed.

Rx FIFO Overflow Detection: When a byte is received by the receiver and the Rx FIFO is not read and already contains 16 bytes, a FIFO overflow error will be asserted (RFO bit set to 1 in RX_STATUS register). The received byte will be discarded leaving the FIFO with the first 16 bytes received. If the ONACK bit in the CTRL register is asserted, the

transmitter will generate a NACK for the overflow condition every time a new byte is received that cannot be stored in the Rx FIFO. The EMV SIM module will continually send NACKs till the Rx FIFO is read or NACK transmit threshold is reached.

14.7.7 Protocol Timers

The EMV SIM module has several 16-bit and 32-bit timers to allow the software perform protocol timing checks and measurements like:

- Guard Time
- Character Wait Time
- Block Wait Time
- Block Guard Time
- Work Wait Time

In addition to the above, the EMV SIM module also contains two 32-bit general purpose counters which can be used by software for any application specific time measurement.

14.7.7.1 Transmit Guard Time

The Transmit Guard time is the number of ETUs inserted by the transmitter between each character being transmitted. The number of ETUs is controlled by programming the TX_GETU register. The Guard Time is measured as the time (in ETUs) between the STOP bits of previous character and the START bit of next character.

For a Guard Time programmed as 0xFF in GETU[7:0] bits of TX_GETU register, the transmitter inserts one STOP bit instead of two.

14.7.7.2 Character Wait Time

The EMV SIM receiver block includes a 16-bit counter that counts the number of bit times (ETUs) between received characters (i.e. time between start bits in the consecutive characters received). When enabled, the Character Wait Time Counter (CWT) will not start counting until after the START bit(s) of a valid character has been received. The counter is synchronized to the receive character bit positions so that an accurate count of the number of ETUs between characters can be made. The Character Wait Time Counter has a 16-bit programmable comparator. Software can write the expected number of ETUs between characters to the comparator. If the time between characters exceeds this value, an interrupt flag will be set and an interrupt generated if the mask is clear. The CWT can be configured as follows:

- Program the CWT_VAL register to the value that needs to be enforced
- Activate CWT function by setting the CWT_EN bit in the CTRL register
- Enable the CWT_ERR interrupt by clearing the CWT_IM bit in the INT_MASK register
- If an interrupt occurs, then the software should read the RX_STATUS register to determine if the error was caused by a CWT_ERR violation. The software should clear the CWT_ERR interrupt by writing a “1” to the CWT_ERR bit in the RX_STATUS register
- Software can disable the CWT counter by clearing the CWT_EN bit in the CTRL register at any time

14.7.7.3 Block Wait Time and Block Guard Time

A 32-bit timer is used to measure both Block Wait Time (BWT) and Block Guard Time (BGT). The BWT and BGT timer are used to measure the delay between the leading edge of the last character of the block received by the card and the leading edge of the first character of the next block sent by the card. If this time measured is larger than the value in BWT_VAL register, then the BWT_ERR flag will be set and an interrupt generated. If the time is less than the value in the BGT_VAL register, then the BGT_ERR flag will be set and an interrupt generated. The block wait timer can be configured as follows:

- Program the BWT_VAL and BGT_VAL registers to the value that need to be enforced
- Activate the block wait timer by writing '1' to BWT_EN bit in the CTRL register
- Enable the BWT_ERR and BGT_ERR interrupts by clearing the BWT_IM and BGT_IM bits in the INT_MASK register
- If an interrupt occurs, then the software should read the RX_STATUS register to determine if the error was caused by a BWT_ERR or BGT_ERR violation. The software should clear the interrupt status bit by writing a “1” to the BWT_ERR bit or BGT_ERR, respectively, in the RX_STATUS register
- Software can disable the BWT counter by clearing the BWT_EN bit in the CTRL register at any time

NOTE

BWT should be enabled after all bytes are written to in Tx FIFO. This is necessary to avoid the condition where writes to Tx FIFO are very slow (more than 1 ETU time apart) and if BWT is enabled earlier, then BWT error can trigger falsely.

14.7.7.4 Work Wait Time

There is no separate counter to measure the work wait time (WWT). Work wait time is a combination of BWT and CWT. To measure WWT, software must configure both CWT_VAL and BWT_VAL to the value to enforced. When measuring WWT, the BGT timer will not be used so it should be masked (inactive) by the BGT_IM bit. Below is the sequence to program the EMV SIM to send and receive data while checking WWT:

- Program both the CWT_VAL and the BWT_VAL registers to the WWT (work wait time) value that needs to be enforced. Set BGT register to 0 (this is the default value).
- Activate both the CWT and BWT functions by setting the CWT_EN and BWT_EN bit in the CTRL register
- Enable the CWT_ERR and BWT_ERR interrupts by clearing the CWT_IM and BWT_IM bits in the INT_MASK register
- Program the EMV SIM module to enable Receiver or Transmitter as desired by application
- If an interrupt occurs, then the software should consider that a WWT violation if the CWT_ERR or BWT_ERR bits are asserted. The software should clear the CWT_ERR or BWT_ERR interrupt by writing a “1” to the BWT_ERR or CWT_ERR bit in the RX_STATUS register

14.7.7.5 General Purpose Timers

The EMV SIM module provides TWO 16-bit counters for use, when timing events during Smart Card communication. The clock source for the counter is selectable between three sources: Card Clock, Receiver Clock (16 times the ETU clock) or ETU Clock (for Transmitter). The GPCNT0/1_CLKSEL[1:0] bits in the CTRL register are used to select the clock input. The counter is enabled as soon as the input clock is selected. The starting of the counter is immediate once the input clock is running. Software can control the three input clock sources by using the SCEN, KILL_CLOCKS, RCV_EN and XMT_EN bits provided in the CTRL register.

To run the counters from the card clock source the following conditions must be met:

- ‘KILL_CLOCKS = 0’ in the CTRL register
- ‘RCV_EN = 1’ or XMT_EN = 1 in the CTRL register
- 'SCEN = 1' in the PCSR register
- ‘GPCNT_CLKSET[1:0] = 01’ in the CLKCFG register

The counter will begin to count at the card clock rate as soon as these conditions are met.

To run the counters from the receive clock source the following conditions must be met:

- ‘KILL_CLOCKS = 0’ in the CTRL register.

- ‘RCV_EN = 1’ or XMT_EN = 1 in the CTRL register
- ‘GPCNT0/1_CLKSET[1:0] = 10’ in the CLKCFG register

The counter will begin to count at the receive clock rate as soon as these conditions are met.

To run the counters from the ETU (transmit) clock source the following conditions must be met:

- ‘KILL_CLOCKS = 0’ in the CTRL register.
- ‘RCV_EN = 1’ or XMT_EN = 1 in the CTRL register
- ‘GPCNT0/1_CLKSET[1:0] = 11’ in the CLKCFG register

The counters will begin to count at the ETU clock rate as soon as these conditions are met.

The counters can be reset by setting GPCNT0/1_CLKSEL[1:0] to 00. A 16-bit comparator value (GPCNT0/1_VAL register) is provided that allows the software to select a count value at which the interrupt flags (GPCNT0/1_TO bit in TX_STATUS register) can be set and an interrupt generated if the mask (GPCNT0/1_IM bits in INT_MASK register) are clear.

The following sequence should be followed to configure both general purpose timers:

- Program counter comparator using the GPCNT0/1_VAL register
- Enable the conditions described above using the CTRL register
- Select desired clock source for the General Purpose Counter using the CLKCFG register
- Enable interrupts in the INT_MASK register

14.7.8 Answer To Reset (ATR) Detection

The first communication that occurs between the Smart Card and the EMV SIM module will be a block of data sent from the Smart Card to the EMV SIM module after the card is powered up and the card reset is removed (As described in Smart Card Interface and Control Section). This block is called the Answer To Reset (ATR). To receive the ATR, the EMV SIM module should be configured for 12 ETU character reception. According to the ISO 7816-3 spec, both “T=0” and “T=1” cards will communicate initially using 12 ETU character durations.

The following steps provide a suggested approach to configure the EMV SIM module to receive the ATR.

- Clear RCVR11 bit in the CTRL register

- Set ANACK bit in the CTRL register to enable NACK generation. The ISO 7816-3 spec allows the EMV SIM module to NACK any communication errors that occur during the initial communication at 12 ETU.

NOTE

The Europay MasterCard and VISA (EMV) cards are similar to "T=1", but do not allow the EMV SIM module to NACK during the initial communication

- Enable RDTF and RFO interrupts by setting RDT_IM and RFO_IM bits in INT_MASK register. These interrupts are enabled to notify the software when characters are received. A threshold can be set for the number of characters to receive before generating an interrupt.
- Set desired threshold for received characters by writing RDT in RX_THD register

The EMV SIM should be setup to perform in initial character mode during ATR detection. This will cause the hardware to identify the first valid character sent during the ATR as an initial character. This character will automatically configure the hardware for the data convention used by the Smart Card.

- Set Initial Character Mode by setting the ICM bit in the CTRL register

The ISO 7816-3 spec requires that Smart Cards meet certain timing restrictions. One of these is the time from the de-assertion of the card reset to the beginning of the ATR sequence. The EMV SIM module's General Purpose Counter can be used to verify that the Smart Card begins its ATR within the 400 to 40,000 card clock cycle range.

- Set the General Purpose Counter 0 or 1 Timeout Value to 0x9C40 using GPCNT0/1_VAL register.
- Enable General Purpose Counter Interrupt by clearing GPCNT0/1_IM in INT_MASK register.
- Enable General Purpose Counter by programming the GPCNT0/1_CLKSEL[1:0] bits to 01, so that the card clock is used for counting.

The ISO7816-3 spec states that the maximum allowed time between two characters during the ATR is 9600 ETUs (Initial Waiting Time). The Character Wait Time Counter should be setup to detect any errors for this condition.

- Set Character Wait Time Counter Comparator to 9600 using the CWT_VAL register
- Enable the Character Wait Time Counter Interrupt by clearing CWT_ERR_IM in INT_MASK register
- Enable the Character Wait Time Counter by setting the CWT_EN bit in the CTRL register

The last step in preparing for ATR reception is to enable the receiver.

- Set RCV_EN bit in CTRL register

The EMV SIM module will generate interrupts once a threshold number of characters is received. The software should react to these interrupts and read the characters from the receive FIFO (RX_BUF) until the complete ATR has been received. If a General Purpose Counter interrupt occurs before the final ATR character is received, then the card should be deactivated according to the ISO 7816-3 spec (See Section Smart Card Interface and Control). Otherwise, once a valid ATR is received, the software will know from the ATR information the specific characteristics for this card (refer to the ISO 7816-3 spec for details).

14.7.8.1 Programming Considerations for T=0 Smart Cards

If the card is of type T=0, the software should adjust the following parameters according to the information in the ATR:

- Adjust the baud rate by changing the values of DIVISOR register based on the values of 'F' and 'D' received in the ATR.
- Adjust the guard time between characters by changing the value of GETU[7:0] in the TX_GETU register.
- Adjust NACK capability by modifying the values of the ONACK and ANACK bits in the CTRL register.
- Adjust the stop clock polarity by modifying the value of the SCSP bit in the PCSR register.
- Adjust the level of transmit NACK re-transmissions allowed by modifying the value of the TNCK_THD[3:0] bits in the TX_THD register.
- Adjust the level for the Receive NACK threshold by modifying the RNCK_THD[3:0] bits in the RX_THD register.

If a negotiation with the Smart Card is desired, the software sends a PPS response to the Smart Card. To send the response, the following steps should be performed:

- Set the desired transmit FIFO threshold level by writing the TDT[3:0] bits in the TX_THD register.
- Clear all transmit interrupt flags in the TX_STATUS register by writing a one to the bits of the register.
- Enable the transmit interrupts desired by clearing the mask bits in the INT_MASK register. If more than 16 characters are to be sent, it is suggested that the TDTF interrupt be used to signify when to write more characters to the transmit FIFO. This results in the most efficient transfer times to the Smart Card.
- Enable the transmitter by setting the XMT_EN bit in the CTRL register.
- Write the characters to be sent as response (max 16) to the transmit FIFO using the TX_BUF register.

At this point, the EMV SIM module will transmit the characters in the transmit FIFO. If more than 16 characters are to be sent, the transmit threshold interrupt will be set when the threshold number of characters are remaining in the FIFO. The software can then write an additional number of characters to be sent without interrupting transmission to the Smart Card.

Once the transmission is complete, the EMV SIM module should be completely configured for standard operation with the T=0 Smart Card. The software can continue to service RDTF interrupts for received characters, and TDTF interrupts for transmitted characters.

14.7.8.2 Programming Considerations for T=1 Smart Cards

If the card is of type T=1, the software should adjust the following parameters according to the information in the ATR:

- Adjust the baud rate by changing the values of DIVISOR register based on the values of 'F' and 'D' received in the ATR.
- Adjust the guard time between characters by changing the value of GETU[7:0] in the TX_GETU register. Setting GETU[7:0] to 0xFF configures the SIM transmitter for 11 ETU transmissions.
- Disable NACK capability by clearing the ONACK and ANACK bits in the CTRL register. T=1 cards do not allow NACKs.
- Adjust the stop clock polarity by modifying the value of the SCSP bit in the PCSR register.
- Set Character Wait Time Counter Timeout Value to value specified in the ATR by using the CWT_VAL register.
- Enable the Character Wait Time Counter Interrupt by clearing CWT_ERR_IM in INT_MASK register.
- Enable the Character Wait Time Counter by setting the CWT_EN bit in the CTRL register.
- Enable CRC or LRC error checking according to the ATR information by setting either the CRC_EN or LRC_EN bit in the CTRL register. These bits should never be set at the same time!

For T=1 cards, the ATR is sent using a T=0 type of structure (12 ETU, no LRC or CRC). If a negotiation with the Smart Card is desired, the software will send a PPS response to the Smart Card. Otherwise, the protocol is initiated with a block transfer from the EMV SIM module. In order to send the response or the first block, the following steps should be performed:

- Set the desired transmit FIFO threshold level by writing the TDT[3:0] bits in the TX_THD register.

- Clear all transmit interrupt flags in the TX_STATUS register by writing a one to the bits of the register.
- Enable the transmit interrupts desired by clearing the mask bits in the INT_MASK register. If more than 16 characters are to be sent, it is suggested that the TDTF interrupt be used to signify when to write more characters to the transmit FIFO. This results in the most efficient transfer times to the Smart Card.
- Enable the transmission of the error checking characters (LRC or CRC) by setting the XMT_CRC_LRC bit in the CTRL register.

NOTE

If the card supports PPS, the software may not be allowed to send the LRC/CRC information until the PPS exchange is completed. If so, do not set the XMT_CRC_LRC bit during the PPS exchange.

- Enable the transmitter by setting the XMT_EN bit in the CTRL register.
- Write the characters to be sent as response (max 16) to the transmit FIFO using the TX_BUF register.

At this point, the EMV SIM module will transmit the characters in the transmit FIFO. If more than 16 characters are to be sent, the transmit threshold interrupt will be set when the threshold number of characters are remaining in the FIFO. The software can then write an additional number of characters to be sent without interrupting transmission to the Smart Card.

Once the transmission is complete, the EMV SIM module should be completely configured for standard operation with the T=1 Smart Card. The software can continue to service RDTF interrupts for received characters, and TDTF interrupts for transmitted characters.

Chapter 15
Reset Control Module (RCM)

15.1 Introduction

Information found here 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.

See AN4503: Power Management for Kinetis MCUs for further details on using the RCM.

15.2 Reset memory map and register descriptions

The RCM Memory Map/Register Definition can be found here.

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	15.2.1/340
4007_F001	System Reset Status Register 1 (RCM_SRS1)	8	R	00h	15.2.2/341
4007_F004	Reset Pin Filter Control register (RCM_RPFC)	8	R/W	00h	15.2.3/343
4007_F005	Reset Pin Filter Width register (RCM_RPFW)	8	R/W	00h	15.2.4/344

Table continues on the next page...

RCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4007_F006	Force Mode Register (RCM_FM)	8	R/W	00h	15.2.5/345
4007_F007	Mode Register (RCM_MR)	8	R/W	See section	15.2.6/346
4007_F008	Sticky System Reset Status Register 0 (RCM_SSRS0)	8	R/W	82h	15.2.7/347
4007_F009	Sticky System Reset Status Register 1 (RCM_SSRS1)	8	R/W	00h	15.2.8/348

15.2.1 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 POR	Power-On Reset 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. 0 Reset not caused by POR 1 Reset caused by POR
6 PIN	External Reset Pin Indicates a reset has been caused by an active-low level on the external $\overline{\text{RESET}}$ pin. 0 Reset not caused by external reset pin 1 Reset caused by external reset pin

Table continues on the next page...

RCM_SRS0 field descriptions (continued)

Field	Description
5 WDOG	Watchdog Indicates a reset has been caused by the watchdog timer timing out. This reset source can be blocked by disabling the watchdog. 0 Reset not caused by watchdog timeout 1 Reset caused by watchdog timeout
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 LOL	Loss-of-Lock Reset 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. 0 Reset not caused by a loss of lock in the PLL 1 Reset caused by a loss of lock in the PLL
2 LOC	Loss-of-Clock Reset 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. 0 Reset not caused by a loss of external clock. 1 Reset caused by a loss of external clock.
1 LVD	Low-Voltage Detect Reset If PMC_LVDSC1[LVDRE] is set and the supply drops below the LVD trip voltage, an LVD reset occurs. This field is also set by POR. 0 Reset not caused by LVD trip or POR 1 Reset caused by LVD trip or POR
0 WAKEUP	Low Leakage Wakeup Reset 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. 0 Reset not caused by LLWU module wakeup source 1 Reset caused by LLWU module wakeup source

15.2.2 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	0	0	SACKERR	0	MDM_AP	SW	LOCKUP	0
Write								
Reset	0	0	0	0	0	0	0	0

RCM_SRS1 field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SACKERR	Stop Mode Acknowledge Error Reset 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. 0 Reset not caused by peripheral failure to acknowledge attempt to enter stop mode 1 Reset caused by peripheral failure to acknowledge attempt to enter stop mode
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 MDM_AP	MDM-AP System Reset Request Indicates a reset has been caused by the host debugger system setting of the System Reset Request bit in the MDM-AP Control Register. 0 Reset not caused by host debugger system setting of the System Reset Request bit 1 Reset caused by host debugger system setting of the System Reset Request bit
2 SW	Software Indicates a reset has been caused by software setting of SYSRESETREQ bit in Application Interrupt and Reset Control Register in the ARM core. 0 Reset not caused by software setting of SYSRESETREQ bit 1 Reset caused by software setting of SYSRESETREQ bit
1 LOCKUP	Core Lockup Indicates a reset has been caused by the ARM core indication of a LOCKUP event. 0 Reset not caused by core LOCKUP event 1 Reset caused by core LOCKUP event
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

15.2.3 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 .

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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 RSTFLTSS	Reset Pin Filter Select in Stop Mode Selects how the reset pin filter is enabled in Stop and VLPS modes , and also during LLS and VLLS modes. On exit from VLLS mode, this bit should be reconfigured before clearing PMC_REGSC[ACKISO]. 0 All filtering disabled 1 LPO clock filter enabled
RSTFLTSRW	Reset Pin Filter Select in Run and Wait Modes Selects how the reset pin filter is enabled in run and wait modes. 00 All filtering disabled 01 Bus clock filter enabled for normal operation 10 LPO clock filter enabled for normal operation 11 Reserved

15.2.4 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
RSTFLTSEL	Reset Pin Filter Bus Clock Select Selects the reset pin bus clock filter width. 00000 Bus clock filter count is 1 00001 Bus clock filter count is 2 00010 Bus clock filter count is 3 00011 Bus clock filter count is 4 00100 Bus clock filter count is 5 00101 Bus clock filter count is 6 00110 Bus clock filter count is 7 00111 Bus clock filter count is 8 01000 Bus clock filter count is 9 01001 Bus clock filter count is 10 01010 Bus clock filter count is 11 01011 Bus clock filter count is 12 01100 Bus clock filter count is 13 01101 Bus clock filter count is 14 01110 Bus clock filter count is 15 01111 Bus clock filter count is 16 10000 Bus clock filter count is 17 10001 Bus clock filter count is 18 10010 Bus clock filter count is 19 10011 Bus clock filter count is 20 10100 Bus clock filter count is 21 10101 Bus clock filter count is 22 10110 Bus clock filter count is 23 10111 Bus clock filter count is 24 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 11010 Bus clock filter count is 27 11011 Bus clock filter count is 28 11100 Bus clock filter count is 29 11101 Bus clock filter count is 30 11110 Bus clock filter count is 31 11111 Bus clock filter count is 32

15.2.5 Force Mode Register (RCM_FM)

NOTE

The reset values of the bits in the FORCEROM field are for Chip POR only. They are unaffected by other reset types.

Address: 4007_F000h base + 6h offset = 4007_F006h

Bit	7	6	5	4	3	2	1	0
Read	0					FORCEROM		0
Write								
Reset	0	0	0	0	0	0	0	0

RCM_FM field descriptions

Field	Description
7–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–1 FORCEROM	Force ROM Boot When either bit is set, will force boot from ROM during all subsequent system resets. 00 No effect 01 Force boot from ROM with RCM_MR[1] set. 10 Force boot from ROM with RCM_MR[2] set. 11 Force boot from ROM with RCM_MR[2:1] set.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

15.2.6 Mode Register (RCM_MR)

This register includes 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					BOOTROM		0
Write						w1c		
Reset	0	0	0	0	0	*	*	0

* Notes:
- BOOTROM field: The reset state of this register depends on the boot mode.

RCM_MR field descriptions

Field	Description
7–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–1 BOOTROM	Boot ROM Configuration Indicates the boot source, the boot source remains set until the next System Reset or software can write logic one to clear the corresponding mode bit. While either bit is set, the NMI input is disabled and the vector table is relocated to the ROM base address at 0x1C00_0000. These bits should be cleared by writing logic one before executing any code from either Flash or SRAM. 00 Boot from Flash 01 Boot from ROM due to BOOTCFG0 pin assertion 10 Boot form ROM due to FOPT[7] configuration 11 Boot from ROM due to both BOOTCFG0 pin assertion and FOPT[7] configuration
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

15.2.7 Sticky System Reset Status Register 0 (RCM_SSRS0)

This register includes status flags to indicate all reset sources since the last POR, LVD or VLLS Wakeup that have not been cleared by software. Software can clear the status flags by writing a logic one to a flag.

Address: 4007_F000h base + 8h offset = 4007_F008h

Bit	7	6	5	4	3	2	1	0
Read	SPOR	SPIN	SWDOG	0	SLOL	SLOC	SLVD	SWAKEUP
Write	w1c	w1c	w1c		w1c	w1c	w1c	w1c
Reset	1	0	0	0	0	0	1	0

RCM_SSRS0 field descriptions

Field	Description
7 SPOR	Sticky Power-On Reset 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. 0 Reset not caused by POR 1 Reset caused by POR
6 SPIN	Sticky External Reset Pin Indicates a reset has been caused by an active-low level on the external $\overline{\text{RESET}}$ pin. 0 Reset not caused by external reset pin 1 Reset caused by external reset pin
5 SWDOG	Sticky Watchdog Indicates a reset has been caused by the watchdog timer timing out.This reset source can be blocked by disabling the watchdog. 0 Reset not caused by watchdog timeout 1 Reset caused by watchdog timeout
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 SLOL	Sticky Loss-of-Lock Reset 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. 0 Reset not caused by a loss of lock in the PLL 1 Reset caused by a loss of lock in the PLL
2 SLOC	Sticky Loss-of-Clock Reset

Table continues on the next page...

RCM_SSRS0 field descriptions (continued)

Field	Description
	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. 0 Reset not caused by a loss of external clock. 1 Reset caused by a loss of external clock.
1 SLVD	Sticky Low-Voltage Detect Reset If PMC_LVDSC1[LVDRE] is set and the supply drops below the LVD trip voltage, an LVD reset occurs. This field is also set by POR. 0 Reset not caused by LVD trip or POR 1 Reset caused by LVD trip or POR
0 SWAKEUP	Sticky Low Leakage Wakeup Reset Indicates a reset has been caused by an enabled LLWU modulewakeup 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. 0 Reset not caused by LLWU module wakeup source 1 Reset caused by LLWU module wakeup source

15.2.8 Sticky System Reset Status Register 1 (RCM_SSRS1)

This register includes status flags to indicate all reset sources since the last POR, LVD or VLLS Wakeup that have not been cleared by software. Software can clear the status flags by writing a logic one to a flag.

Address: 4007_F000h base + 9h offset = 4007_F009h

Bit	7	6	5	4	3	2	1	0
Read	0	0	SSACKERR	0	SMDM_AP	SSW	SLOCKUP	0
Write			w1c		w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0

RCM_SSRS1 field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SSACKERR	Sticky Stop Mode Acknowledge Error Reset

Table continues on the next page...

RCM_SSRS1 field descriptions (continued)

Field	Description
	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. 0 Reset not caused by peripheral failure to acknowledge attempt to enter stop mode 1 Reset caused by peripheral failure to acknowledge attempt to enter stop mode
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 SMDM_AP	Sticky MDM-AP System Reset Request Indicates a reset has been caused by the host debugger system setting of the System Reset Request bit in the MDM-AP Control Register. 0 Reset not caused by host debugger system setting of the System Reset Request bit 1 Reset caused by host debugger system setting of the System Reset Request bit
2 SSW	Sticky Software Indicates a reset has been caused by software setting of SYSRESETREQ bit in Application Interrupt and Reset Control Register in the ARM core. 0 Reset not caused by software setting of SYSRESETREQ bit 1 Reset caused by software setting of SYSRESETREQ bit
1 SLOCKUP	Sticky Core Lockup Indicates a reset has been caused by the ARM core indication of a LOCKUP event. 0 Reset not caused by core LOCKUP event 1 Reset caused by core LOCKUP event
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Chapter 16
System Mode Controller (SMC)

16.1 Introduction

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.

See AN4503: Power Management for Kinetis MCUs for further details on using the SMC.

16.2 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 Kinetis microcontrollers.

The following table shows the translation between the ARM CPU modes and the Kinetis 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 Kinetis MCU documentation normally uses wait and stop.

In addition, Kinetis 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 16-1. 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.
HSRUN	The MCU can be run at a faster frequency compared with RUN mode and the internal supply is fully regulated. See the Power Management chapter for details about the maximum allowable frequencies.
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 16-1. Power modes (continued)

Mode	Description
LLS3	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.
LLS2	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 and powering down the system RAM3 partition. The system RAM1 partition, internal logic and I/O states are retained.[1]
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 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 STOPCTRL[PORPO].

1. See the devices' chip configuration details for the size and location of the system RAM partitions.

16.3 Memory map and register descriptions

Information about the registers related to the system mode controller can be found here.

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.

NOTE

Before executing the WFI instruction, the last register written to must be read back. This ensures that all register writes associated with setting up the low power mode being entered have completed before the MCU enters the low power mode.

Failure to do this may result in the low power mode not being entered correctly.

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	See section	16.3.1/354
4007_E001	Power Mode Control register (SMC_PMCTRL)	8	R/W	See section	16.3.2/355
4007_E002	Stop Control Register (SMC_STOPCTRL)	8	R/W	03h	16.3.3/357
4007_E003	Power Mode Status register (SMC_PMSTAT)	8	R	See section	16.3.4/358

16.3.1 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 PMCTRL[RUNM] remains 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	AHSRUN	0	AVLP	0	ALLS	0	AVLLS	0
Write								
Reset	0	0	*	0	0	0	0	0

* Notes:
- AVLP field: When booting in run mode, the reset value is 0. When booting in VLPR mode, the reset value is 1.

SMC_PMPROT field descriptions

Field	Description
7 AHSRUN	Allow High Speed Run mode

Table continues on the next page...

SMC_PMPROT field descriptions (continued)

Field	Description
	Provided the appropriate control bits are set up in PMCTRL, this write-once field allows the MCU to enter High Speed Run mode (HSRUN). 0 HSRUN is not allowed 1 HSRUN is allowed
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 AVLP	Allow Very-Low-Power Modes Provided the appropriate control bits are set up in PMCTRL, this write-once field allows the MCU to enter any very-low-power mode (VLPR, VLPW, and VLPS). 0 VLPR, VLPW, and VLPS are not allowed. 1 VLPR, VLPW, and VLPS are allowed.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 ALLS	Allow Low-Leakage Stop Mode Provided the appropriate control bits are set up in PMCTRL, this write-once field allows the MCU to enter any low-leakage stop mode (LLS). 0 Any LLSx mode is not allowed 1 Any LLSx mode is allowed
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 AVLLS	Allow Very-Low-Leakage Stop Mode 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). 0 Any VLLSx mode is not allowed 1 Any VLLSx mode is allowed
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

16.3.2 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	Reserved	RUNM		0	STOPA	STOPM		
Write								
Reset	0	*	*	0	0	0	0	0

* Notes:
- RUNM field: When booting in run mode, the reset value is 00. When booting in VLPR mode, the reset value is 10.

SMC_PMCTRL field descriptions

Field	Description
7 Reserved	This field is reserved. This bit is reserved for future expansion and should always be written zero.
6–5 RUNM	Run Mode Control 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. **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. **NOTE:** RUNM may be set to HSRUN only when PMSTAT=RUN. After being programmed to HSRUN, RUNM should not be programmed back to RUN until PMSTAT=HSRUN. Also, stop mode entry should not be attempted while RUNM=HSRUN or PMSTAT=HSRUN. 00 Normal Run mode (RUN) 01 Reserved 10 Very-Low-Power Run mode (VLPR) 11 High Speed Run mode (HSRUN)
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 STOPA	Stop Aborted When set, this read-only status bit indicates an interrupt occured during the previous stop mode entry sequence, preventing the system from entering that mode. This field is cleared by reset or by hardware at the beginning of any stop mode entry sequence and is set if the sequence was aborted. 0 The previous stop mode entry was successsful. 1 The previous stop mode entry was aborted.
STOPM	Stop Mode Control 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. **NOTE:** When set to VLLSxor LLSx, the LLSM in the STOPCTRL register is used to further select the particular VLLSor LLS submode which will be entered. **NOTE:** When set to STOP, the PSTOPO bits in the STOPCTRL register can be used to select a Partial Stop mode if desired. 000 Normal Stop (STOP) 001 Reserved

Table continues on the next page...

SMC_PMCTRL field descriptions (continued)

Field	Description
	010 Very-Low-Power Stop (VLPS) 011 Low-Leakage Stop (LLSx) 100 Very-Low-Leakage Stop (VLLSx) 101 Reserved 110 Reseved 111 Reserved

16.3.3 Stop Control Register (SMC_STOPCTRL)

The STOPCTRL register provides various control bits allowing the user to fine tune power consumption during the stop mode selected by the STOPM field.

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	PSTOPO		PORPO	0	LPOPO	LLSM		
Write								
Reset	0	0	0	0	0	0	1	1

SMC_STOPCTRL field descriptions

Field	Description
7–6 PSTOPO	Partial Stop Option These bits control whether a Partial Stop mode is entered when STOPM=STOP. When entering a Partial Stop mode from RUN (or VLPR) mode, the PMC, MCG and flash remain fully powered, allowing the device to wakeup almost instantaneously at the expense of higher power consumption. In PSTOP2, only system clocks are gated allowing peripherals running on bus clock to remain fully functional. In PSTOP1, both system and bus clocks are gated. 00 STOP - Normal Stop mode 01 PSTOP1 - Partial Stop with both system and bus clocks disabled 10 PSTOP2 - Partial Stop with system clock disabled and bus clock enabled 11 Reserved
5 PORPO	POR Power Option This bit controls whether the POR detect circuit is enabled in VLLS0 mode. 0 POR detect circuit is enabled in VLLS0 1 POR detect circuit is disabled in VLLS0

Table continues on the next page...

SMC_STOPCTRL field descriptions (continued)

Field	Description
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 LPOPO	LPO Power Option Controls whether the 1 kHz LPO clock is enabled in LLS/VLLSx modes. **NOTE:** During VLLS0 mode, the LPO clock is disabled by hardware and this bit has no effect. 0 LPO clock is enabled in LLS/VLLSx 1 LPO clock is disabled in LLS/VLLSx
LLSM	LLS or VLLS Mode Control This field controls which LLS orVLLS sub-mode to enter if STOPM = LLSx orVLLSx. 000 VLLS0 if PMCTRL[STOPM]=VLLSx, reserved if PMCTRL[STOPM]=LLSx 001 VLLS1 if PMCTRL[STOPM]=VLLSx, reserved if PMCTRL[STOPM]=LLSx 010 VLLS2 if PMCTRL[STOPM]=VLLSx, LLS2 if PMCTRL[STOPM]=LLSx 011 VLLS3 if PMCTRL[STOPM]=VLLSx, LLS3 if PMCTRL[STOPM]=LLSx 100 Reserved 101 Reserved 110 Reserved 111 Reserved

16.3.4 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	PMSTAT							
Write								
Reset	*	*	*	*	*	*	*	*

* Notes:
- PMSTAT field: When booting in run mode, the reset value is 0x01. When booting in VLPR mode, the reset value is 0x04.

SMC_PMSTAT field descriptions

Field	Description
PMSTAT	Power Mode Status **NOTE:** When debug is enabled, the PMSTAT will not update to STOP or VLPS **NOTE:** When a PSTOP mode is enabled, the PMSTAT will not update to STOP or VLPS **NOTE:** Since the RUNM bits in the PMCTRL register are reset to VLPR on any Chip Reset not VLLS, the PMSTAT will update to VLPR shortly after the reset sequence is complete. 0000_0001 Current power mode is RUN. 0000_0010 Current power mode is STOP. 0000_0100 Current power mode is VLPR. 0000_1000 Current power mode is VLPW. 0001_0000 Current power mode is VLPS. 0010_0000 Current power mode is LLS. 0100_0000 Current power mode is VLLS. 1000_0000 Current power mode is HSRUN

16.4 Functional description

16.4.1 Power mode transitions

The following figure shows the power mode state transitions available on the chip. Any reset will initially bring the MCU back to the normal RUN state. However, in order to minimize peak power consumption, the RUNM bits in the PMCTRL register can be reset to VLPR via Flash IFR settings, causing the SMC to begin transitioning the MCU into VLPR mode during the reset recovery sequence.

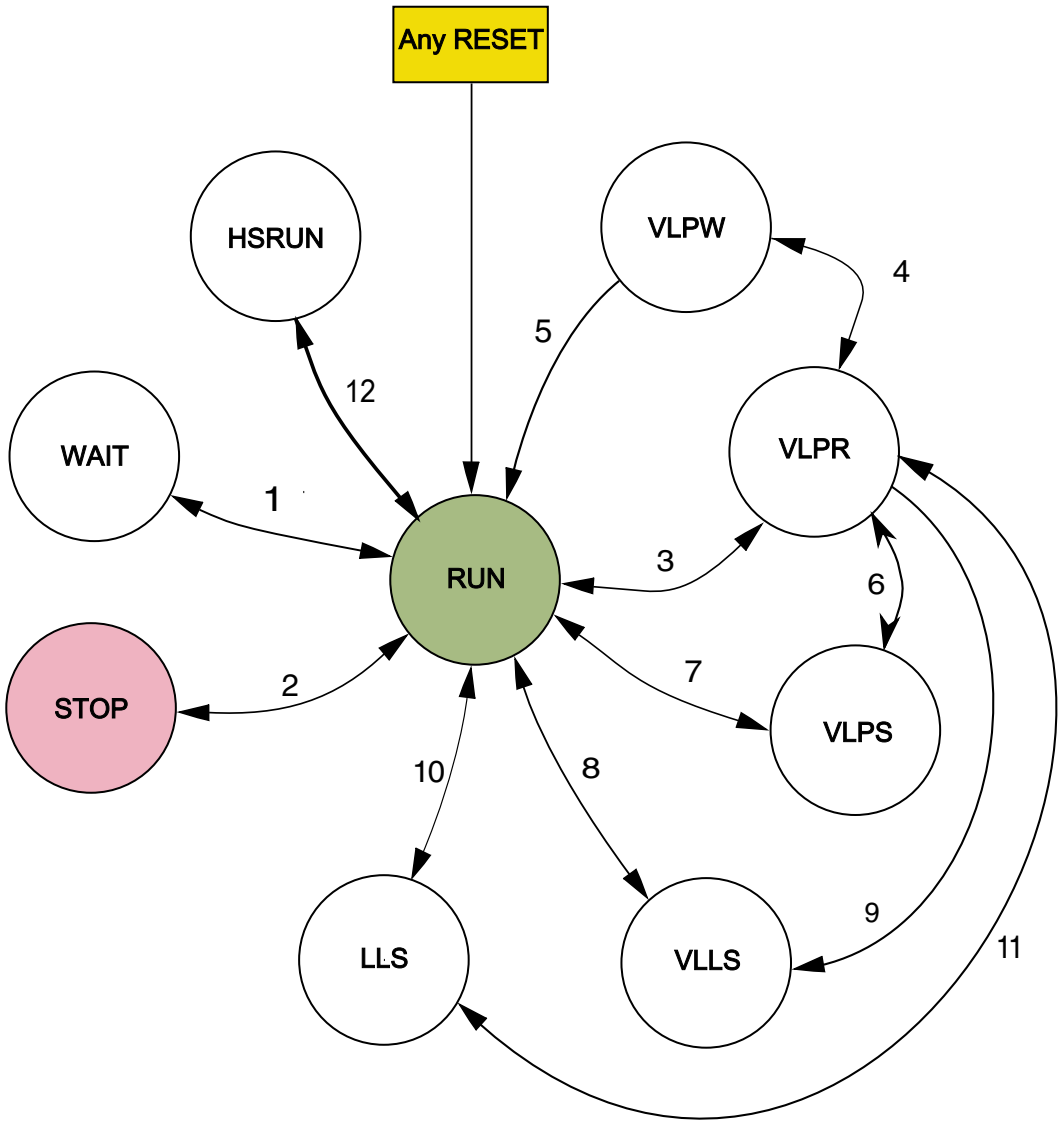


Figure 16-1. Power mode state diagram

The following table defines triggers for the various state transitions shown in the previous figure.

Table 16-2. 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. See note.[1]
	WAIT	RUN	Interrupt or Reset
2	RUN	STOP	PMCTRL[RUNM]=00, PMCTRL[STOPM]=000[2]

Table continues on the next page...

Table 16-2. Power mode transition triggers (continued)

Transition #	From	To	Trigger conditions
			Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core. 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. Set PMPROT[AVLP]=1, PMCTRL[RUNM]=10. **NOTE:** In order to limit peak current, PMPROT[AVLP] and PMCTRL[RUNM] bits can be set on any Reset via Flash IFR settings, causing the SMC to transition the MCU from RUN->VLPR during the reset recovery sequence.
	VLPR	RUN	Set PMCTRL[RUNM]=00 or 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. See note.[1]
	VLPW	VLPR	Interrupt
5	VLPW	RUN	Reset
6	VLPR	VLPS	PMCTRL[STOPM]=000[3] or 010, Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core. See note.[1]
	VLPS	VLPR	Interrupt **NOTE:** If VLPS was entered directly from RUN (transition #7), hardware forces exit back to RUN and does not allow a transition to VLPR.
7	RUN	VLPS	PMPROT[AVLP]=1, PMCTRL[STOPM]=010, Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core. See note.[1]
	VLPS	RUN	Interrupt and VLPS mode was entered directly from RUN or Reset
8	RUN	VLLSx	PMPROT[AVLLS]=1, PMCTRL[STOPM]=100, STOPCTRL[LLSM]=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

Table continues on the next page...

Table 16-2. Power mode transition triggers (continued)

Transition #	From	To	Trigger conditions
9	VLPR	VLLSx	PMPROT[AVLLS]=1, PMCTRL[STOPM]=100, STOPCTRL[LLSM]=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	LLSx	PMPROT[ALLS]=1, PMCTRL[STOPM]=011, STOPCTRL[LLSM]=x (LLSx), Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core.
	LLSx	RUN	Wakeup from enabled LLWU input source and LLSx mode was entered directly from RUN or RESET pin.
11	VLPR	LLSx	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.
	LLSx	VLPR	Wakeup from enabled LLWU input source and LLSx mode was entered directly from VLPR **NOTE:** If LLSx was entered directly from RUN, hardware will not allow this transition and will force exit back to RUN
12	RUN	HSRUN	Set PMPROT[AHSRUN]=1, PMCTRL[RUNM]=11.
	HSRUN	RUN	Set PMCTRL[RUNM]=00 or Reset

1. If debug is enabled, the core clock remains to support debug.
2. If PMCTRL[STOPM]=000 and STOPCTRL[PSTOPO]=01 or 10, then only a Partial Stop mode is entered instead of STOP
3. If PMCTRL[STOPM]=000 and STOPCTRL[PSTOPO]=00, then VLPS mode is entered instead of STOP. If PMCTRL[STOPM]=000 and STOPCTRL[PSTOPO]=01 or 10, then only a Partial Stop mode is entered instead of VLPS

16.4.2 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. This 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.

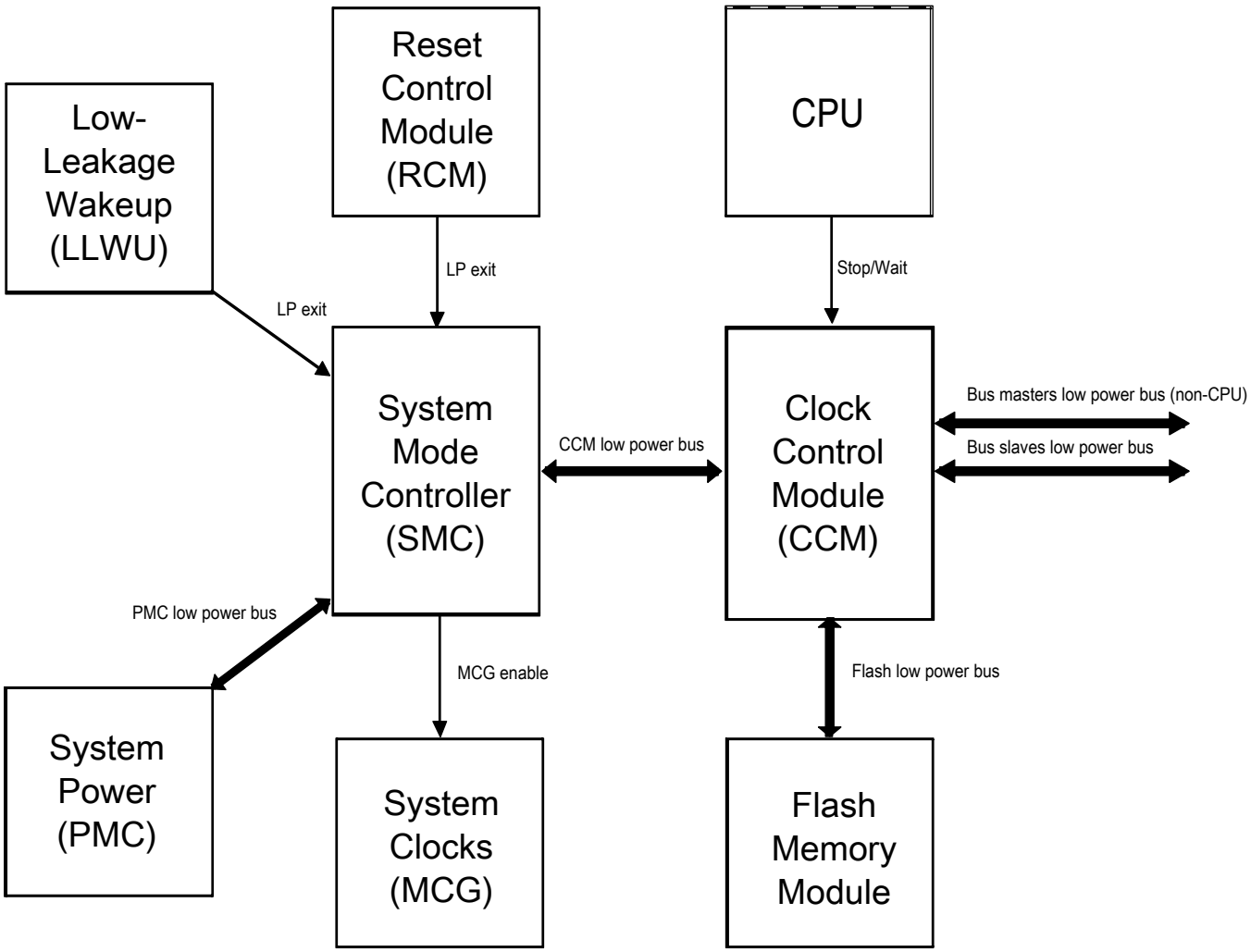


Figure 16-2. Low-power system components and connections

16.4.2.1 Stop mode entry sequence

Entry into a low-power stop mode (Stop, VLPS, LLS, VLLSx) is initiated by a CPU executing 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.

16.4.2.2 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.

16.4.2.3 Aborted stop mode entry

If an interrupt 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 interrupt occurs before the PMC begins the transition to stop mode regulation. After this point, the interrupt is ignored until the PMC has completed its transition to stop mode regulation. When an aborted stop mode entry sequence occurs, SMC_PMCTRL[STOPA] is set to 1.

16.4.2.4 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.

16.4.2.5 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. As part of this transition, system clocking is re-established and is equivalent to the normal RUN and VLPR mode clocking configuration.

16.4.3 Run modes

The run modes supported by this device can be found here.

- Run (RUN)

- Very Low-Power Run (VLPR)
- High Speed Run (HSRUN)

16.4.3.1 RUN mode

This mode is selected after any reset. In order to reduce peak power consumption, however, the SMC will begin transitioning the MCU into VLPR mode during the reset recovery sequence. 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 (or PCC's) registers.

16.4.3.2 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] must be set to 10b to enter VLPR.
- Flash programming/erasing is not allowed.

NOTE

Do not increase the clock frequency while in VLPR mode, because the regulator is slow in responding and cannot manage fast load transitions. In addition, do not modify the clock source in the MCG module or any clock divider registers. Module clock enables in the SIM can be set, however should only be cleared one at a time to limit load transitions.

To reenter Normal Run mode, clear PMCTRL[RUNM]. PMSTAT 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 PMSTAT until it is set to RUN when returning from VLPR mode.

Any reset will initially cause the device to exit to RUN mode, however Flash IFR settings can be used to return the part to VLPR during the MCU reset flow.

16.4.3.3 High Speed Run (HSRUN) mode

In HSRUN mode, the on-chip voltage regulator remains in a run regulation state, but with a slightly elevated voltage output. In this state, the MCU is able to operate at a faster frequency compared to normal RUN mode. For the maximum allowable frequencies, see the Power Management chapter.

While in this mode, the following restrictions must be adhered to:

- The maximum allowable change in frequency of the system, bus, flash or core clocks is restricted to 2x (double the frequency).
- Before exiting HSRUN mode, clock frequencies should be reduced back down to those acceptable in RUN mode.
- Stop mode entry is not supported from HSRUN.
- Modifications to clock gating control bits are prohibited.
- Flash programming/erasing is not allowed.

To enter HSRUN mode, set PMPORT[AHSRUN]=HSRUN and set PMCTRL[RUNM]=HSRUN. Before increasing clock frequencies, the PMSTAT register should be polled to determine when the system has completed entry into HSRUN mode. To reenter normal RUN mode, clear RUNM. Any reset will also clear RUNM and cause the system to exit to normal RUN mode after the MCU exits its reset flow.

16.4.4 Wait modes

This device contains two different wait modes which are listed here.

- Wait
- Very-Low Power Wait (VLPW)

16.4.4.1 WAIT mode

WAIT mode is entered when the ARM core enters the Sleep-Now or Sleep-On-Exit modes while SLEEPDEEP 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 module.

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, temporarily returning the device to RUN mode before transitioning into VLPR during the MCU reset flow.

16.4.4.2 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 (or PCC).

VLPR mode restrictions also apply to VLPW.

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, temporarily returning the device to RUN mode before transitioning into VLPR during the MCU reset flow.

16.4.5 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, LLSand 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)

16.4.5.1 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, temporarily returning the device to RUN mode before transitioning into VLPR during the MCU reset flow.

16.4.5.2 Very-Low-Power Stop (VLPS) mode

The two ways in which VLPS mode can be entered are listed here.

- 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 PMCTRL[STOPM] = 010 or 000.
- 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 PMCTRL[STOPM] = 010. 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.

A system reset will cause an exit from VLPS mode, temporarily returning the device to RUN mode before transitioning into VLPR during the MCU reset flow.

16.4.5.3 Low-Leakage Stop (LLSx) modes

This device contains two Low-Leakage Stop modes: LLS3 and LLS2. LLS or LLSx is often used in this document to refer to both modes. All LLS modes 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 16-2.

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 Wake-up (LLWU) module to enable the desired wake-up sources. The available wake-up 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. If LLS was entered from VLPR mode, then the MCU will begin VLPR entry shortly after wake-up. In the LLWU interrupt service routine (ISR), the user can poll the LLWU module wake-up 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, temporarily returning the device to normal RUN mode before transitioning into VLPR during the MCU reset flow. When LLS is exiting via the $\overline{\text{RESET}}$ pin, RCM_SRS[PIN] and RCM_SRS[WAKEUP] are set.

16.4.5.4 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 16-2.

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 Wake-up (LLWU) module to enable the desired wakeup sources. The available wake-up 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. If VLLS was entered from VLPR mode, then the MCU will begin VLPR entry shortly after wakeup. In the LLWU interrupt service routine (ISR), the user can poll the LLWU module wake-up flags to determine the source of the wake-up.

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 PMC_REGSC[ACKISO] is set.

An asserted $\overline{\text{RESET}}$ pin will cause an exit from any VLLS mode, temporarily returning the device to normal RUN mode before transitioning into VLPR during the MCU reset flow. When exiting VLLS via the $\overline{\text{RESET}}$ pin, RCM_SRS[PIN] and RCM_SRS[WAKEUP] are set.

16.4.6 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 17
Power Management Controller (PMC)

17.1 Introduction

The power management controller (PMC) contains the internal voltage regulator, power on reset (POR), low voltage detect system (LVD), and high voltage detect system (HVD).

See AN4503: Power Management for Kinetis MCUs for further details on using the PMC.

17.2 Features

A list of included PMC features can be found here.

- 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
- High-voltage detect supporting two high-voltage trip points

17.3 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 LVDSC1[LVDV]. 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 in the Low Voltage Status and Control 1 Register (LVDSC1[LVDF]) operates in a level sensitive manner. LVDSC1[LVDF] is set when the supply voltage falls below the selected trip point (VLVD). LVDSC1[LVDF] is cleared by writing 1 to LVDSC1[LVDACK], but only if the internal supply has returned above the trip point; otherwise, LVDSC1[LVDF] remains set.
- The Low Voltage Warning Flag (LVWF) in the Low Voltage Status and Control 2 Register (LVDSC2[LVWF]) operates in a level sensitive manner. LVDSC2[LVWF] is set when the supply voltage falls below the selected monitor trip point (VLVW). LVDSC2[LVWF] is cleared by writing one to LVDSC2[LVWACK], but only if the internal supply has returned above the trip point; otherwise, LVDSC2[LVWF] remains set.

17.3.1 LVD reset operation

By setting LVDSC1[LVDRE], the LVD generates a reset upon detection of a low-voltage condition. The low-voltage detection threshold is determined by LVDSC1[LVDV]. After an LVD reset occurs, the LVD system holds the MCU in reset until the supply voltage rises above this threshold. The LVD field in the SRS register of the RCM module (RCM_SRS[LVD]) is set following an LVD or power-on reset.

17.3.2 LVD interrupt operation

By configuring the LVD circuit for interrupt operation (LVDSC1[LVDIE] set and LVDSC1[LVDRE] clear), LVDSC1[LVDF] is set and an LVD interrupt request occurs upon detection of a low voltage condition. LVDSC1[LVDF] is cleared by writing 1 to LVDSC1[LVDACK].

17.3.3 Low-voltage warning (LVW) interrupt operation

The LVD system contains a Low-Voltage Warning Flag (LVWF) in the Low Voltage Detect Status and Control 2 Register 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 LVDSC2[LVWIE]. If enabled, an LVW interrupt request occurs when LVDSC2[LVWF] is set. LVDSC2[LVWF] is cleared by writing 1 to LVDSC2[LVWACK].

LVDSC2[LVWV] selects one of the four trip voltages:

- Highest: V_{LVW4}
- Two mid-levels: V_{LVW3} and V_{LVW2}
- Lowest: V_{LVW1}

17.4 High-voltage detect (HVD) system

This device includes a system to guard against high-voltage conditions.

The system is comprised of a HVD circuit with a user-selectable trip voltage: high (V_{HVDH}) or low (V_{HVDL}). The trip voltage is selected by HVDSC1[HVDV]. The HVD is disabled upon entering VLPx, LLS, and VLLSx modes.

A flag is available to indicate the status of the high-voltage detect system:

- The High Voltage Detect Flag in the High Voltage Status and Control 1 Register (HVDSC1[HVDF]) operates in a level sensitive manner. HVDSC1[HVDF] is set when the supply voltage rises above the selected trip point (VHVD). HVDSC1[HVDF] is cleared by writing 1 to HVDSC1[HVDACK], but only if the internal supply has returned below the trip point; otherwise, HVDSC1[LVDF] remains set.

17.4.1 HVD reset operation

By setting HVDSC1[HVDRE], the HVD generates a reset upon detection of a high-voltage condition. The high-voltage detection threshold is determined by HVDSC1[HVDV]. After an HVD reset occurs, the HVD system holds the MCU in reset until the supply voltage falls below this threshold. The LVD field in the SRS register of the RCM module (RCM_SRS[LVD]) is set following an HVD reset.

17.4.2 HVD interrupt operation

By configuring the HVD circuit for interrupt operation (HVDSC1[HVDIE] set and HVDSC1[HVDRE] clear), HVDSC1[HVDF] is set and an HVD interrupt request occurs upon detection of a high voltage condition. HVDSC1[HVDF] is cleared by writing 1 to HVDSC1[HVDACK].

17.5 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 wake-up 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 wake-up event (with the exception of wake-up by reset event) until the wake-up has been acknowledged via a write to REGSC[ACKISO]. 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 REGSC[ACKISO] is needed.

17.6 Memory map and register descriptions

Details about the PMC registers can be found here.

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	17.6.1/377
4007_D001	Low Voltage Detect Status And Control 2 register (PMC_LVDSC2)	8	R/W	00h	17.6.2/378
4007_D002	Regulator Status And Control register (PMC_REGSC)	8	R/W	See section	17.6.3/379
4007_D00B	High Voltage Detect Status And Control 1 register (PMC_HVDSC1)	8	R/W	01h	17.6.4/381

17.6.1 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 Power Mode Protection (PMPROT) register of the SMC module (SMC_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 LVDF	Low-Voltage Detect Flag This read-only status field indicates a low-voltage detect event. 0 Low-voltage event not detected 1 Low-voltage event detected
6 LVDACK	Low-Voltage Detect Acknowledge This write-only field is used to acknowledge low voltage detection errors. Write 1 to clear LVDF. Reads always return 0.
5 LVDIE	Low-Voltage Detect Interrupt Enable Enables hardware interrupt requests for LVDF. 0 Hardware interrupt disabled (use polling) 1 Request a hardware interrupt when LVDF = 1

Table continues on the next page...

PMC_LVDSC1 field descriptions (continued)

Field	Description
4 LVDRE	Low-Voltage Detect Reset Enable This write-once bit enables LVDF events to generate a hardware reset. Additional writes are ignored. 0 LVDF does not generate hardware resets 1 Force an MCU reset when LVDF = 1
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
LVDV	Low-Voltage Detect Voltage Select Selects the LVD trip point voltage (V_{LVD}). 00 Low trip point selected ($V_{LVD} = V_{LVDL}$) 01 High trip point selected ($V_{LVD} = V_{LVDH}$) 10 Reserved 11 Reserved

17.6.2 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.

NOTE

LVWV is reset solely on a POR Only event. The other fields of the register 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 LVWF	Low-Voltage Warning Flag This read-only status field 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 may be 1 after power-on reset, therefore, to use LVW interrupt function, before enabling LVWIE, LVWF must be cleared by writing LVWACK first. 0 Low-voltage warning event not detected 1 Low-voltage warning event detected
6 LVWACK	Low-Voltage Warning Acknowledge This write-only field is used to acknowledge low voltage warning errors. Write 1 to clear LVWF. Reads always return 0.
5 LVWIE	Low-Voltage Warning Interrupt Enable Enables hardware interrupt requests for LVWF. 0 Hardware interrupt disabled (use polling) 1 Request a hardware interrupt when LVWF = 1
4–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
LVWV	Low-Voltage Warning Voltage Select Selects the LVW trip point voltage (V_{LVW}). The actual voltage for the warning depends on LVDSC1[LVDV]. 00 Low trip point selected ($V_{LVW} = V_{LVW1}$) 01 Mid 1 trip point selected ($V_{LVW} = V_{LVW2}$) 10 Mid 2 trip point selected ($V_{LVW} = V_{LVW3}$) 11 High trip point selected ($V_{LVW} = V_{LVW4}$)

17.6.3 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	VLPO	HPSB	Reserved	BGEN	ACKISO	REGONS	Reserved
Write						w1c		
Reset	0	0		0	0	1	0	0

PMC_REGSC field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 VLPO	VLPx Option When used in conjunction with BGEN, this bit allows additional clock sources and higher frequency operation (at the cost of higher power) to be selected during VLPx modes. 0 Operating frequencies and MCG clocking modes are restricted during VLPx modes as listed in the Power Management chapter. 1 If BGEN is also set, operating frequencies and MCG clocking modes are unrestricted during VLPx modes. Note that flash access frequency is still restricted however.
5 HPSB	High Power Source Bias in VLPS Operation HPSB configures the source bias circuit for high power mode during VLPS operation.When transition to the VLPS mode, clear this bit to reduce power consumption 0 Source bias circuit set for low power mode during VLPS 1 Source bias circuit set for high power mode during VLPS
5 Reserved	This field is reserved.
4 BGEN	Bandgap Enable In VLPx Operation 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. **NOTE:** When the bandgap voltage reference is not needed in low power modes, clear BGEN to avoid excess power consumption. 0 Bandgap voltage reference is disabled in VLPx , LLS , and VLLSx modes. 1 Bandgap voltage reference is enabled in VLPx , LLS , and VLLSx modes.
3 ACKISO	Acknowledge Isolation Reading this field 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 1 to this field when it is set releases the I/O pads and certain peripherals to their normal run mode state. **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. 0 Peripherals and I/O pads are in normal run state. 1 Certain peripherals and I/O pads are in an isolated and latched state.
2 REGONS	Regulator In Run Regulation Status

Table continues on the next page...

PMC_REGSC field descriptions (continued)

Field	Description
	This read-only field provides the current status of the internal voltage regulator. 0 Regulator is in stop regulation or in transition to/from it 1 Regulator is in run regulation
1 Reserved	This field is reserved. **NOTE:** This reserved bit must remain cleared (set to 0).
0 BGBE	Bandgap Buffer Enable Enables the bandgap buffer. 0 Bandgap buffer not enabled 1 Bandgap buffer enabled

17.6.4 High Voltage Detect Status And Control 1 register (PMC_HVDSC1)

This register contains status and control bits to support the high 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 HVD system is disabled regardless of HVDSC1 settings. To protect systems that must have HVD always on, configure the Power Mode Protection (PMPROT) register of the SMC module (SMC_PMPROT) to disallow any very low power or low leakage modes from being enabled.

See the device's data sheet for the exact HVD trip voltages.

NOTE

The HVDV field is reset solely on a POR only event. The register's other fields are reset on Chip Reset instead of VLLS. For more information about these reset types, refer to the Reset section details.

Address: 4007_D000h base + Bh offset = 4007_D00Bh

Bit	7	6	5	4	3	2	1	0
Read	HVDF	0	HVDIE	HVDRE	0			HVDV
Write		HVDACK						
Reset	0	0	0	0	0	0	0	1

PMC_HVDSC1 field descriptions

Field	Description
7 HVDF	High-Voltage Detect Flag This read-only status field indicates a high-voltage detect event. 0 High-voltage event not detected 1 High-voltage event detected
6 HVDACK	High-Voltage Detect Acknowledge This write-only field is used to acknowledge high voltage detection errors. Write 1 to clear HVDF. Reads always return 0.
5 HVDIE	High-Voltage Detect Interrupt Enable Enables hardware interrupt requests for HVDF. 0 Hardware interrupt disabled (use polling) 1 Request a hardware interrupt when HVDF = 1
4 HVDRE	High-Voltage Detect Reset Enable This write-once bit enables HVDF events to generate a hardware reset. Additional writes are ignored until the next chip reset. 0 HVDF does not generate hardware resets 1 Force an MCU reset when HVDF = 1
3–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 HVDV	High-Voltage Detect Voltage Select Selects the HVD trip point voltage (V_{HVD}). 0 Low trip point selected ($V_{HVD} = V_{HVDL}$) 1 High trip point selected ($V_{HVD} = V_{HVDH}$)

Chapter 18
Interrupt Multiplexer (INTMUX)

18.1 About this module

18.1.1 Introduction

The Interrupt Multiplexer (INTMUX) expands the number of peripherals that can interrupt a core via 4 channels of an NVIC module.

- Each INTMUX module has 4 channels that are assigned to 4 NVIC interrupt slots.
- Each INTMUX channel can provide up to 32 additional interrupt sources, which can be logically OR'd or ANDed.
- Each INTMUX routes the interrupt sources to the interrupt outputs.

See INTMUX0 input mux assignment for the interrupt multiplexer input channels assignment.

18.1.2 Features

INTMUX features:

- Supports 4 multiplex channels
- Each channel receives 32 interrupt sources and has 1 interrupt output
- Each interrupt source can be enabled or disabled
- Each channel supports *Logic AND* or *Logic OR* of all enabled interrupt sources

18.1.3 Block diagram

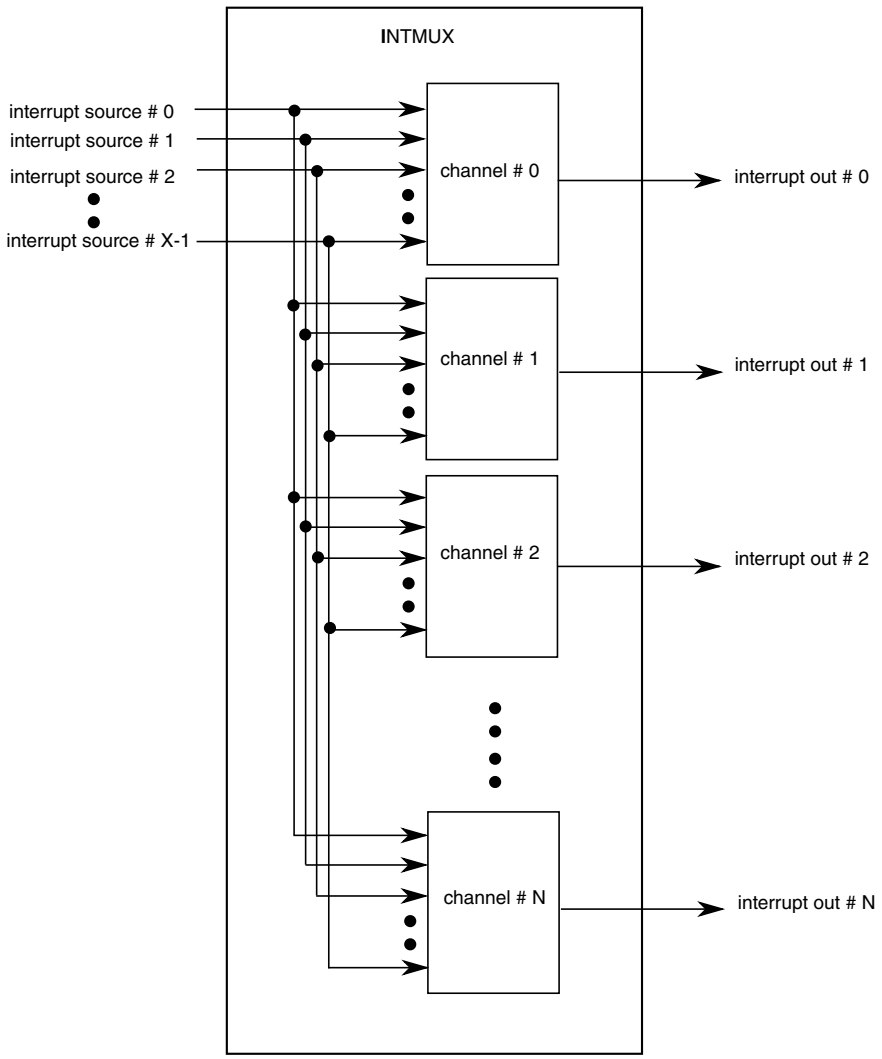


Figure 18-1. INTMUX Block diagram

N: Interrupt Channel Instance Number (N=3)

X: Interrupt Source Number for each channel (X=32)

18.2 Memory Map and register definition

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

INTMUX memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_4000	Channel n Control Status Register (INTMUX0_CH0_CSR)	32	R/W	See section	18.2.1/385
4002_4004	Channel n Vector Number Register (INTMUX0_CH0_VEC)	32	R	0000_0000h	18.2.2/386
4002_4010	Channel n Interrupt Enable Register (INTMUX0_CH0_IER_31_0)	32	R/W	0000_0000h	18.2.3/387
4002_4020	Channel n Interrupt Pending Register (INTMUX0_CH0_IPR_31_0)	32	R	0000_0000h	18.2.4/387
4002_4040	Channel n Control Status Register (INTMUX0_CH1_CSR)	32	R/W	See section	18.2.1/385
4002_4044	Channel n Vector Number Register (INTMUX0_CH1_VEC)	32	R	0000_0000h	18.2.2/386
4002_4050	Channel n Interrupt Enable Register (INTMUX0_CH1_IER_31_0)	32	R/W	0000_0000h	18.2.3/387
4002_4060	Channel n Interrupt Pending Register (INTMUX0_CH1_IPR_31_0)	32	R	0000_0000h	18.2.4/387
4002_4080	Channel n Control Status Register (INTMUX0_CH2_CSR)	32	R/W	See section	18.2.1/385
4002_4084	Channel n Vector Number Register (INTMUX0_CH2_VEC)	32	R	0000_0000h	18.2.2/386
4002_4090	Channel n Interrupt Enable Register (INTMUX0_CH2_IER_31_0)	32	R/W	0000_0000h	18.2.3/387
4002_40A0	Channel n Interrupt Pending Register (INTMUX0_CH2_IPR_31_0)	32	R	0000_0000h	18.2.4/387
4002_40C0	Channel n Control Status Register (INTMUX0_CH3_CSR)	32	R/W	See section	18.2.1/385
4002_40C4	Channel n Vector Number Register (INTMUX0_CH3_VEC)	32	R	0000_0000h	18.2.2/386
4002_40D0	Channel n Interrupt Enable Register (INTMUX0_CH3_IER_31_0)	32	R/W	0000_0000h	18.2.3/387
4002_40E0	Channel n Interrupt Pending Register (INTMUX0_CH3_IPR_31_0)	32	R	0000_0000h	18.2.4/387

18.2.1 Channel n Control Status Register (INTMUXx_CH*n*_CSR)

Address: 4002_4000h base + 0h offset + (64d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IRQP	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				CHIN				0		IRQN		0		AND	RST
W																
Reset	0	0	0	0	*	*	*	*	0	0	0	0	0	0	0	0

* Notes:

- CHIN field: The reset value of CHIN depends on the channel number.

INTMUX*x*_CH*n*_CSR field descriptions

Field	Description
31 IRQP	Channel Interrupt Request Pending This bit is available for polling purposes and represents the channel output. 0 No interrupt is pending. 1 The interrupt output of this channel is pending.
30–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–8 CHIN	Channel Instance Number These bits represent the instance number of this channel. N Channel N
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–4 IRQN	Channel Input Number These bits represent the interrupt input number of this channel. 00 32 interrupt inputs 01 Reserved 10 Reserved 11 Reserved
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 AND	Logic AND This bit used to Logic AND or Logic OR all enabled interrupt inputs of this channel and generate the interrput output of this channel. 0 Logic OR all enabled interrupt inputs. 1 Logic AND all enabled interrupt inputs.
0 RST	Software Reset This bit is used to software reset this channel. This bit always reads as 0. 0 No operation. 1 Perform a software reset on this channel.

18.2.2 Channel n Vector Number Register (INTMUX*x*_CH*n*_VEC)

Address: 4002_4000h base + 4h offset + (64d × 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	0																VECN														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

INTMUXx_CHn_VEC field descriptions

Field	Description
31–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–2 VECN	Vector Number These bits represent the vector number for the highest priority interrupt source of this channel. VECN[13:2] = (CPU vectors + NVIC vectors) + Number of the highest priority interrupt source of this channel. **NOTE:** The interrupt priority is based on the Interrupt Pending Register. The smaller the interrupt source number, the higher the priority.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

18.2.3 Channel n Interrupt Enable Register (INTMUXx_CHn_IER_31_0)

Address: 4002_4000h base + 10h offset + (64d × 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 W	INTE																															
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

INTMUXx_CHn_IER_31_0 field descriptions

Field	Description
INTE	Interrupt Enable These bits are used to enable the interrupt sources of this channel. 0 Interrupt is disabled. 1 Interrupt is enabled.

18.2.4 Channel n Interrupt Pending Register (INTMUXx_CHn_IPR_31_0)

Address: 4002_4000h base + 20h offset + (64d × 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	INTP																															
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

INTMUX*x*_CH*n*_IPR_31_0 field descriptions

Field	Description
INTP	Interrupt Pending These bits represent the interrupt source pending status of this channel. 0 No interrupt. 1 Interrupt is pending.

18.3 Functional Description

The Interrupt Multiplexer (INTMUX) routes the interrupt sources to the interrupt outputs.

There is one interrupt output for each channel available in the INTMUX. All interrupt sources are available to each channel. There are 4 multiplex channels, and there are 32 interrupt sources.

The INTMUX module provides interrupt status registers to monitor interrupt pending status and vector numbers. This module also implements the ability to logical AND or OR enabled interrupts on a given channel. Software resets can also be performed on a given channel.

See the chip specific Interrupt section for more information regarding the interrupt vectors and sources.

18.3.1 Configuring Output Channels

There is a single set of 32 source interrupts for the INTMUX. Each of the 4 output channels can be configured to output any combination of the source interrupts.

To enable source interrupts on a channel, write a mask value to the Channel n Interrupt Enable Register (CHn_IER_31_0). Each bit of these registers corresponds to one of the 32 source interrupts. Bit n enables source interrupt n on the given channel.

Further control for each output channel is available by selecting the Boolean operation to perform on pending interrupts. The default is logical OR, but logical AND can be selected via the CHn_CSR[AND] register bit.

18.3.2 INTMUX Vectors

The INTMUX is designed so that it integrates with the standard NVIC vector table. Each INTMUX source interrupt has a vector associated with it. These source vectors are intended to immediately follow the NVIC vectors in the vector table pointed to by the CPU's VTOR register. Interrupt service routine addresses can be placed in the INTMUX source vectors, just like NVIC vectors.

The ISR for each of the INTMUX output channels should read the CHn_VEC register to determine the highest priority active source interrupt out of those enabled for the given channel. The CHn_VEC[VECN] bitfield returns the source interrupt number. The VECN field starts at bit 2 of the CHn_VEC register, so reading the register as a whole will return the VECN field multiplied by 4. This is the offset in bytes of the active source vector from the start of the vector table (i.e., VTOR).

If multiple source interrupts become pending simultaneously on a single channel, the CHn_VEC[VECN] field will read as the highest priority vector number. Priority of source interrupts is determined by the source interrupt number. Lower numbers are higher priortity, with source interrupt 0 being highest priority.

NOTE

Unlike the NVIC, the INTMUX does not latch pending source interrupts. This means that the INTMUX output channel ISRs must check for and handle a 0 value of the CHn_VEC register to account for spurious interrupts.

Chapter 19
Low-Leakage Wakeup Unit (LLWU)

19.1 Chip-specific LLWU information

19.1.1 Wake-up sources

The device uses the following internal peripherals and external pin inputs as wakeup sources to the LLWU module. LLWU_Px are external pin inputs, and LLWU_M0IF-M7IF are connections to the internal peripheral interrupt flags.

NOTE

In addition to the LLWU wakeup sources, the device also wakes from low power modes when $\overline{\text{NMI}}$ or $\overline{\text{RESET}}$ pins are enabled and the respective pin is asserted.

Table 19-1. Wakeup sources for LLWU inputs

LLWU pins	Module sources or pin names
LLWU_P0	PTE1
LLWU_P1	PTE2
LLWU_P2	PTE4
LLWU_P3	PTA4
LLWU_P4	PTA13
LLWU_P5	PTB0
LLWU_P6	PTC1
LLWU_P7	PTC3
LLWU_P8	PTC4
LLWU_P9	PTC5
LLWU_P10	PTC6
LLWU_P11	PTC11
LLWU_P12	PTD0

Table continues on the next page...

Table 19-1. Wakeup sources for LLWU inputs (continued)

LLWU pins	Module sources or pin names
LLWU_P13	PTD2
LLWU_P14	PTD4
LLWU_P15	PTD6
LLWU_P16	PTE6
LLWU_P17	PTE9
LLWU_P18	PTE10
LLWU_P19	Reserved
LLWU_P20	Reserved
LLWU_P21	Reserved
LLWU_P22	PTA10
LLWU_P23	PTA11
LLWU_P24	PTD8
LLWU_P25	PTD11
LLWU_P26	Reserved
LLWU_P27	USB0_DP
LLWU_P28	USB0_DM[1]
LLWU_P29	Reserved
LLWU_P30	Reserved
LLWU_P31	Reserved
LLWU_M0IF	LPTMR0 or LPTMR1[2]
LLWU_M1IF	CMP0
LLWU_M2IF	Reserved
LLWU_M3IF	Reserved
LLWU_M4IF	TSI0[2]
LLWU_M5IF	RTC alarm
LLWU_M6IF	Reserved
LLWU_M7IF	RTC second

1. A wakeup source of LLWU, USB0_DP or USB0_DM is available only when the chip is in USB host mode.
2. Requires the peripheral and the peripheral interrupt to be enabled. The LLWU_ME[WUMEn] (n=0-7) bit enables the internal module flag a wakeup inputs. After wakeup, the flags are cleared based on the peripheral clearing mechanism.

19.2 Introduction

The LLWU module allows the user to select up to 25 external pins and up to 5 internal modules as interrupt wake-up sources from low-leakage power modes.

The input sources are described in the device's chip configuration details. Each of the available wake-up 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 module also includes optional digital pin filters for the external wakeup pins.

See AN4503: Power Management for Kinetis MCUs for further details on using the LLWU.

19.2.1 Features

The LLWU module features include:

- Support for up to 25 external input pins and up to 5 internal modules with individual enable bits for MCU interrupt from low leakage modes
- 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 wake-up inputs, each of which is programmable as falling-edge, rising-edge, or any change
- Wake-up inputs that are activated after MCU enters a low-leakage power mode
- Optional digital filters provided to qualify an external pin detect. Note that when the LPO clock is disabled, the filters are disabled and bypassed.

19.2.2 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 wake-up events until the user has acknowledged the wake-up via a write to PMC_REGSC[ACKISO].

19.2.2.1 LLS mode

Wake-up events due to external pin inputs (LLWU_Px) and internal module interrupt inputs (LLWU_MxIF) result in an interrupt 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.

19.2.2.2 VLLS modes

All wakeup and reset events result in VLLS exit via a reset flow.

19.2.2.3 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 wake-up pin filters are enabled, filter operation begins immediately. If a low leakage mode is entered within five LPO clock cycles of an active edge, the edge event will be detected by the LLWU.

19.2.2.4 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.

19.2.3 Block diagram

The following figure is the block diagram for the LLWU module.

19.3 LLWU signal descriptions

The signal properties of LLWU are shown in the table found here.

The external wakeup input pins can be enabled to detect either rising-edge, falling-edge, or on any change.

Table 19-2. LLWU signal descriptions

Signal	Description	I/O
LLWU_Pn	Wakeup inputs (n = 0-)	I

19.4 Memory map/register definition

The LLWU includes the following registers:

- Wake-up source enable registers
  - Enable external pin input sources
  - Enable internal peripheral interrupt sources
- Wake-up flag registers
  - Indication of wakeup source that caused exit from a low-leakage power mode includes external pin or internal module interrupt
- Wake-up pin filter enable registers

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	19.4.1/396
4007_C001	LLWU Pin Enable 2 register (LLWU_PE2)	8	R/W	00h	19.4.2/397
4007_C002	LLWU Pin Enable 3 register (LLWU_PE3)	8	R/W	00h	19.4.3/398
4007_C003	LLWU Pin Enable 4 register (LLWU_PE4)	8	R/W	00h	19.4.4/399
4007_C004	LLWU Pin Enable 5 register (LLWU_PE5)	8	R/W	00h	19.4.5/400
4007_C005	LLWU Pin Enable 6 register (LLWU_PE6)	8	R/W	00h	19.4.6/401
4007_C006	LLWU Pin Enable 7 register (LLWU_PE7)	8	R/W	00h	19.4.7/402
4007_C007	LLWU Pin Enable 8 register (LLWU_PE8)	8	R/W	00h	19.4.8/404
4007_C008	LLWU Module Enable register (LLWU_ME)	8	R/W	00h	19.4.9/405
4007_C009	LLWU Pin Flag 1 register (LLWU_PF1)	8	R/W	00h	19.4.10/406
4007_C00A	LLWU Pin Flag 2 register (LLWU_PF2)	8	R/W	00h	19.4.11/408
4007_C00B	LLWU Pin Flag 3 register (LLWU_PF3)	8	R/W	00h	19.4.12/410

Table continues on the next page...

LLWU memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4007_C00C	LLWU Pin Flag 4 register (LLWU_PF4)	8	R/W	00h	19.4.13/411
4007_C00D	LLWU Module Flag 5 register (LLWU_MF5)	8	R	00h	19.4.14/413
4007_C00E	LLWU Pin Filter 1 register (LLWU_FILT1)	8	R/W	00h	19.4.15/415
4007_C00F	LLWU Pin Filter 2 register (LLWU_FILT2)	8	R/W	00h	19.4.16/416
4007_C010	LLWU Pin Filter 3 register (LLWU_FILT3)	8	R/W	00h	19.4.17/417
4007_C011	LLWU Pin Filter 4 register (LLWU_FILT4)	8	R/W	00h	19.4.18/418

19.4.1 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 Write	WUPE3		WUPE2		WUPE1		WUPE0	
Reset	0	0	0	0	0	0	0	0

LLWU_PE1 field descriptions

Field	Description
7–6 WUPE3	Wakeup Pin Enable For LLWU_P3 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
5–4 WUPE2	Wakeup Pin Enable For LLWU_P2 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection

Table continues on the next page...

LLWU_PE1 field descriptions (continued)

Field	Description
3–2 WUPE1	Wakeup Pin Enable For LLWU_P1 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
WUPE0	Wakeup Pin Enable For LLWU_P0 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection

19.4.2 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 Write	WUPE7		WUPE6		WUPE5		WUPE4	
Reset	0	0	0	0	0	0	0	0

LLWU_PE2 field descriptions

Field	Description
7–6 WUPE7	Wakeup Pin Enable For LLWU_P7 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection

Table continues on the next page...

LLWU_PE2 field descriptions (continued)

Field	Description
5–4 WUPE6	Wakeup Pin Enable For LLWU_P6 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
3–2 WUPE5	Wakeup Pin Enable For LLWU_P5 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
WUPE4	Wakeup Pin Enable For LLWU_P4 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection

19.4.3 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 Write	WUPE11		WUPE10		WUPE9		WUPE8	
Reset	0	0	0	0	0	0	0	0

LLWU_PE3 field descriptions

Field	Description
7–6 WUPE11	Wakeup Pin Enable For LLWU_P11

Table continues on the next page...

LLWU_PE3 field descriptions (continued)

Field	Description
	Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
5–4 WUPE10	Wakeup Pin Enable For LLWU_P10 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
3–2 WUPE9	Wakeup Pin Enable For LLWU_P9 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
WUPE8	Wakeup Pin Enable For LLWU_P8 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection

19.4.4 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 Write	WUPE15		WUPE14		WUPE13		WUPE12	
Reset	0	0	0	0	0	0	0	0

LLWU_PE4 field descriptions

Field	Description
7–6 WUPE15	Wakeup Pin Enable For LLWU_P15 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
5–4 WUPE14	Wakeup Pin Enable For LLWU_P14 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
3–2 WUPE13	Wakeup Pin Enable For LLWU_P13 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
WUPE12	Wakeup Pin Enable For LLWU_P12 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection

19.4.5 LLWU Pin Enable 5 register (LLWU_PE5)

LLWU_PE5 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P19-LLWU_P16.

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 Write	WUPE19		WUPE18		WUPE17		WUPE16	
Reset	0	0	0	0	0	0	0	0

LLWU_PE5 field descriptions

Field	Description
7–6 WUPE19	Wakeup Pin Enable For LLWU_P19 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
5–4 WUPE18	Wakeup Pin Enable For LLWU_P18 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
3–2 WUPE17	Wakeup Pin Enable For LLWU_P17 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
WUPE16	Wakeup Pin Enable For LLWU_P16 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection

19.4.6 LLWU Pin Enable 6 register (LLWU_PE6)

LLWU_PE6 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P23-LLWU_P20.

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 Write	WUPE23		WUPE22		WUPE21		WUPE20	
Reset	0	0	0	0	0	0	0	0

LLWU_PE6 field descriptions

Field	Description
7–6 WUPE23	Wakeup Pin Enable For LLWU_P23 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
5–4 WUPE22	Wakeup Pin Enable For LLWU_P22 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
3–2 WUPE21	Wakeup Pin Enable For LLWU_P21 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
WUPE20	Wakeup Pin Enable For LLWU_P20 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection

19.4.7 LLWU Pin Enable 7 register (LLWU_PE7)

LLWU_PE7 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P27-LLWU_P24.

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 Write	WUPE27		WUPE26		WUPE25		WUPE24	
Reset	0	0	0	0	0	0	0	0

LLWU_PE7 field descriptions

Field	Description
7–6 WUPE27	Wakeup Pin Enable For LLWU_P27 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
5–4 WUPE26	Wakeup Pin Enable For LLWU_P26 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
3–2 WUPE25	Wakeup Pin Enable For LLWU_P25 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
WUPE24	Wakeup Pin Enable For LLWU_P24 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection

19.4.8 LLWU Pin Enable 8 register (LLWU_PE8)

LLWU_PE8 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P31-LLWU_P28.

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 Write	WUPE31		WUPE30		WUPE29		WUPE28	
Reset	0	0	0	0	0	0	0	0

LLWU_PE8 field descriptions

Field	Description
7–6 WUPE31	Wakeup Pin Enable For LLWU_P31 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
5–4 WUPE30	Wakeup Pin Enable For LLWU_P30 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
3–2 WUPE29	Wakeup Pin Enable For LLWU_P29 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
WUPE28	Wakeup Pin Enable For LLWU_P28 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection

Table continues on the next page...

LLWU_PE8 field descriptions (continued)

Field	Description
	10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection

19.4.9 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 + 8h offset = 4007_C008h

Bit	7	6	5	4	3	2	1	0
Read 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 WUME7	Wakeup Module Enable For Module 7 Enables an internal module as a wakeup source input. 0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source
6 WUME6	Wakeup Module Enable For Module 6 Enables an internal module as a wakeup source input. 0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source
5 WUME5	Wakeup Module Enable For Module 5 Enables an internal module as a wakeup source input. 0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source
4 WUME4	Wakeup Module Enable For Module 4 Enables an internal module as a wakeup source input.

Table continues on the next page...

LLWU_ME field descriptions (continued)

Field	Description
	0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source
3 WUME3	Wakeup Module Enable For Module 3 Enables an internal module as a wakeup source input. 0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source
2 WUME2	Wakeup Module Enable For Module 2 Enables an internal module as a wakeup source input. 0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source
1 WUME1	Wakeup Module Enable for Module 1 Enables an internal module as a wakeup source input. 0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source
0 WUME0	Wakeup Module Enable For Module 0 Enables an internal module as a wakeup source input. 0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source

19.4.10 LLWU Pin Flag 1 register (LLWU_PF1)

LLWU_PF1 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 + 9h offset = 4007_C009h

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_PF1 field descriptions

Field	Description
7 WUF7	Wakeup Flag For LLWU_P7 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. 0 LLWU_P7 input was not a wakeup source 1 LLWU_P7 input was a wakeup source
6 WUF6	Wakeup Flag For LLWU_P6 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. 0 LLWU_P6 input was not a wakeup source 1 LLWU_P6 input was a wakeup source
5 WUF5	Wakeup Flag For LLWU_P5 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. 0 LLWU_P5 input was not a wakeup source 1 LLWU_P5 input was a wakeup source
4 WUF4	Wakeup Flag For LLWU_P4 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. 0 LLWU_P4 input was not a wakeup source 1 LLWU_P4 input was a wakeup source
3 WUF3	Wakeup Flag For LLWU_P3 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. 0 LLWU_P3 input was not a wakeup source 1 LLWU_P3 input was a wakeup source
2 WUF2	Wakeup Flag For LLWU_P2 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. 0 LLWU_P2 input was not a wakeup source 1 LLWU_P2 input was a wakeup source
1 WUF1	Wakeup Flag For LLWU_P1

Table continues on the next page...

LLWU_PF1 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 WUF1. 0 LLWU_P1 input was not a wakeup source 1 LLWU_P1 input was a wakeup source
0 WUF0	Wakeup Flag For LLWU_P0 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. 0 LLWU_P0 input was not a wakeup source 1 LLWU_P0 input was a wakeup source

19.4.11 LLWU Pin Flag 2 register (LLWU_PF2)

LLWU_PF2 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 + Ah offset = 4007_C00Ah

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_PF2 field descriptions

Field	Description
7 WUF15	Wakeup Flag For LLWU_P15 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.

Table continues on the next page...

LLWU_PF2 field descriptions (continued)

Field	Description
	0 LLWU_P15 input was not a wakeup source 1 LLWU_P15 input was a wakeup source
6 WUF14	Wakeup Flag For LLWU_P14 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. 0 LLWU_P14 input was not a wakeup source 1 LLWU_P14 input was a wakeup source
5 WUF13	Wakeup Flag For LLWU_P13 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. 0 LLWU_P13 input was not a wakeup source 1 LLWU_P13 input was a wakeup source
4 WUF12	Wakeup Flag For LLWU_P12 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. 0 LLWU_P12 input was not a wakeup source 1 LLWU_P12 input was a wakeup source
3 WUF11	Wakeup Flag For LLWU_P11 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. 0 LLWU_P11 input was not a wakeup source 1 LLWU_P11 input was a wakeup source
2 WUF10	Wakeup Flag For LLWU_P10 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. 0 LLWU_P10 input was not a wakeup source 1 LLWU_P10 input was a wakeup source
1 WUF9	Wakeup Flag For LLWU_P9 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. 0 LLWU_P9 input was not a wakeup source 1 LLWU_P9 input was a wakeup source
0 WUF8	Wakeup Flag For LLWU_P8 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. 0 LLWU_P8 input was not a wakeup source 1 LLWU_P8 input was a wakeup source

19.4.12 LLWU Pin Flag 3 register (LLWU_PF3)

LLWU_PF3 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 + Bh offset = 4007_C00Bh

Bit	7	6	5	4	3	2	1	0
Read	WUF23	WUF22	WUF21	WUF20	WUF19	WUF18	WUF17	WUF16
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

LLWU_PF3 field descriptions

Field	Description
7 WUF23	Wakeup Flag For LLWU_P23 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 WUF23. 0 LLWU_P23 input was not a wakeup source 1 LLWU_P23 input was a wakeup source
6 WUF22	Wakeup Flag For LLWU_P22 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 WUF22. 0 LLWU_P22 input was not a wakeup source 1 LLWU_P22 input was a wakeup source
5 WUF21	Wakeup Flag For LLWU_P21 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 WUF21. 0 LLWU_P21 input was not a wakeup source 1 LLWU_P21 input was a wakeup source

Table continues on the next page...

LLWU_PF3 field descriptions (continued)

Field	Description
4 WUF20	Wakeup Flag For LLWU_P20 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 WUF20. 0 LLWU_P20 input was not a wakeup source 1 LLWU_P20 input was a wakeup source
3 WUF19	Wakeup Flag For LLWU_P19 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 WUF19. 0 LLWU_P19 input was not a wakeup source 1 LLWU_P19 input was a wakeup source
2 WUF18	Wakeup Flag For LLWU_P18 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 WUF18. 0 LLWU_P18 input was not a wakeup source 1 LLWU_P18 input was a wakeup source
1 WUF17	Wakeup Flag For LLWU_P17 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 WUF17. 0 LLWU_P17 input was not a wakeup source 1 LLWU_P17 input was a wakeup source
0 WUF16	Wakeup Flag For LLWU_P16 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 WUF16. 0 LLWU_P16 input was not a wakeup source 1 LLWU_P16 input was a wakeup source

19.4.13 LLWU Pin Flag 4 register (LLWU_PF4)

LLWU_PF4 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 + Ch offset = 4007_C00Ch

Bit	7	6	5	4	3	2	1	0
Read	WUF31	WUF30	WUF29	WUF28	WUF27	WUF26	WUF25	WUF24
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

LLWU_PF4 field descriptions

Field	Description
7 WUF31	Wakeup Flag For LLWU_P31 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 WUF31. 0 LLWU_P31 input was not a wakeup source 1 LLWU_P31 input was a wakeup source
6 WUF30	Wakeup Flag For LLWU_P30 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 WUF30. 0 LLWU_P30 input was not a wakeup source 1 LLWU_P30 input was a wakeup source
5 WUF29	Wakeup Flag For LLWU_P29 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 WUF29. 0 LLWU_P29 input was not a wakeup source 1 LLWU_P29 input was a wakeup source
4 WUF28	Wakeup Flag For LLWU_P28 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 WUF28. 0 LLWU_P28 input was not a wakeup source 1 LLWU_P28 input was a wakeup source
3 WUF27	Wakeup Flag For LLWU_P27 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 WUF27. 0 LLWU_P27 input was not a wakeup source 1 LLWU_P27 input was a wakeup source
2 WUF26	Wakeup Flag For LLWU_P26

Table continues on the next page...

LLWU_PF4 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 WUF26. 0 LLWU_P26 input was not a wakeup source 1 LLWU_P26 input was a wakeup source
1 WUF25	Wakeup Flag For LLWU_P25 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 WUF25. 0 LLWU_P25 input was not a wakeup source 1 LLWU_P25 input was a wakeup source
0 WUF24	Wakeup Flag For LLWU_P24 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 WUF24. 0 LLWU_P24 input was not a wakeup source 1 LLWU_P24 input was a wakeup source

19.4.14 LLWU Module Flag 5 register (LLWU_MF5)

LLWU_MF5 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 + Dh offset = 4007_C00Dh

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_MF5 field descriptions

Field	Description
7 MWUF7	Wakeup flag For module 7 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. 0 Module 7 input was not a wakeup source 1 Module 7 input was a wakeup source
6 MWUF6	Wakeup flag For module 6 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. 0 Module 6 input was not a wakeup source 1 Module 6 input was a wakeup source
5 MWUF5	Wakeup flag For module 5 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. 0 Module 5 input was not a wakeup source 1 Module 5 input was a wakeup source
4 MWUF4	Wakeup flag For module 4 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. 0 Module 4 input was not a wakeup source 1 Module 4 input was a wakeup source
3 MWUF3	Wakeup flag For module 3 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. 0 Module 3 input was not a wakeup source 1 Module 3 input was a wakeup source
2 MWUF2	Wakeup flag For module 2 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. 0 Module 2 input was not a wakeup source 1 Module 2 input was a wakeup source
1 MWUF1	Wakeup flag For module 1 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. 0 Module 1 input was not a wakeup source 1 Module 1 input was a wakeup source
0 MWUF0	Wakeup flag For module 0

Table continues on the next page...

LLWU_MF5 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. 0 Module 0 input was not a wakeup source 1 Module 0 input was a wakeup source

19.4.15 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 + Eh offset = 4007_C00Eh

Bit	7	6	5	4	3	2	1	0
Read	FILTF	FILTE		FILTSEL				
Write	w1c							
Reset	0	0	0	0	0	0	0	0

LLWU_FILT1 field descriptions

Field	Description
7 FILTF	Filter Detect Flag 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. 0 Pin Filter 1 was not a wakeup source 1 Pin Filter 1 was a wakeup source
6–5 FILTE	Digital Filter On External Pin Controls the digital filter options for the external pin detect. 00 Filter disabled 01 Filter posedge detect enabled 10 Filter negedge detect enabled 11 Filter any edge detect enabled
FILTSEL	Filter Pin Select Selects 1 of the wakeup pins to be muxed into the filter.

Table continues on the next page...

LLWU_FILT1 field descriptions (continued)

Field	Description
	00000 Select LLWU_P0 for filter ... ... 11111 Select LLWU_P31 for filter

19.4.16 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 + Fh offset = 4007_C00Fh

Bit	7	6	5	4	3	2	1	0
Read	FILTF	FILTE			FILTSEL			
Write	w1c							
Reset	0	0	0	0	0	0	0	0

LLWU_FILT2 field descriptions

Field	Description
7 FILTF	Filter Detect Flag 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. 0 Pin Filter 2 was not a wakeup source 1 Pin Filter 2 was a wakeup source
6–5 FILTE	Digital Filter On External Pin Controls the digital filter options for the external pin detect. 00 Filter disabled 01 Filter posedge detect enabled 10 Filter negedge detect enabled 11 Filter any edge detect enabled
FILTSEL	Filter Pin Select Selects 1 of the wakeup pins to be muxed into the filter. 00000 Select LLWU_P0 for filter

Table continues on the next page...

LLWU_FILT2 field descriptions (continued)

Field	Description
	... ... 11111 Select LLWU_P31 for filter

19.4.17 LLWU Pin Filter 3 register (LLWU_FILT3)

LLWU_FILT3 is a control and status register that is used to enable/disable the digital filter 3 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 + 10h offset = 4007_C010h

Bit	7	6	5	4	3	2	1	0
Read	FILTF	FILTE			FILTSEL			
Write	w1c							
Reset	0	0	0	0	0	0	0	0

LLWU_FILT3 field descriptions

Field	Description
7 FILTF	Filter Detect Flag 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. 0 Pin Filter 3 was not a wakeup source 1 Pin Filter 3 was a wakeup source
6–5 FILTE	Digital Filter On External Pin Controls the digital filter options for the external pin detect. 00 Filter disabled 01 Filter posedge detect enabled 10 Filter negedge detect enabled 11 Filter any edge detect enabled
FILTSEL	Filter Pin Select Selects 1 of the wakeup pins to be muxed into the filter. 00000 Select LLWU_P0 for filter ... ... 11111 Select LLWU_P31 for filter

19.4.18 LLWU Pin Filter 4 register (LLWU_FILT4)

LLWU_FILT4 is a control and status register that is used to enable/disable the digital filter 4 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 + 11h offset = 4007_C011h

Bit	7	6	5	4	3	2	1	0
Read	FILTF	FILTE			FILTSEL			
Write	w1c							
Reset	0	0	0	0	0	0	0	0

LLWU_FILT4 field descriptions

Field	Description
7 FILTF	Filter Detect Flag 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. 0 Pin Filter 4 was not a wakeup source 1 Pin Filter 4 was a wakeup source
6–5 FILTE	Digital Filter On External Pin Controls the digital filter options for the external pin detect. 00 Filter disabled 01 Filter posedge detect enabled 10 Filter negedge detect enabled 11 Filter any edge detect enabled
FILTSEL	Filter Pin Select Selects 1 of the wakeup pins to be muxed into the filter. 00000 Select LLWU_P0 for filter ... ... 11111 Select LLWU_P31 for filter

19.5 Functional description

Thie low-leakage wakeup unit (LLWU) module 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 with the following options:

- 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 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. wakeup detect filters are available for selected external pins. Glitch filtering is not provided on the internal modules.

For internal module interrupts, the WUMEx bit enables the associated module interrupt as a wakeup source.

19.5.1 LLS mode

Wakeup events triggered from either an external pin input or an internal module interrupt, result in a CPU interrupt flow to begin user code execution.

19.5.2 VLLS modes

For any wakeup from VLLS, recovery is always via a reset flow and 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.

19.5.3 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 five 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 PMC_REGSC[ACKISO]. In particular, pin configuration for enabled LLWU wake-up pins must be restored to avoid any LLWU flag from being falsely set when PMC_REGSC[ACKISO] is cleared.

The signal selected as a wake-up source pin must be a digital pin, as selected in the pin mux control.

Chapter 20
Miscellaneous Control Module (MCM)

20.1 Introduction

The Miscellaneous Control Module (MCM) provides a myriad of miscellaneous control functions.

20.1.1 Features

The MCM includes the following features:

- Program-visible information on the platform configuration
- Crossbar master arbitration policy selection
- Flash controller speculation buffer and cache configurations

20.2 Memory map/register descriptions

The memory map and register descriptions found here describe the registers using byte addresses. The registers can be written only when in supervisor mode.

MCM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F000_3008	Crossbar Switch (AXBS) Slave Configuration (MCM_PLASC)	16	R	000Fh	20.2.1/422
F000_300A	Crossbar Switch (AXBS) Master Configuration (MCM_PLAMC)	16	R	000Dh	20.2.2/422
F000_300C	Platform Control Register (MCM_PLACR)	32	R/W	0000_0240h	20.2.3/423
F000_3040	Compute Operation Control Register (MCM_CPO)	32	R/W	0000_0000h	20.2.4/426

20.2.1 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: F000_3000h base + 8h offset = F000_3008h

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	0	1	1	1	1

MCM_PLASC field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ASC	Each bit in the ASC field indicates whether there is a corresponding connection to the crossbar switch's slave input port. 0 A bus slave connection to AXBS input port *n* is absent. 1 A bus slave connection to AXBS input port *n* is present.

20.2.2 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: F000_3000h base + Ah offset = F000_300Ah

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	0	0	1	1	0	1

MCM_PLAMC field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
AMC	Each bit in the AMC field indicates whether there is a corresponding connection to the AXBS master input port.

Table continues on the next page...

MCM_PLAMC field descriptions (continued)

Field	Description
	0 A bus master connection to AXBS input port *n* is absent 1 A bus master connection to AXBS input port *n* is present

20.2.3 Platform Control Register (MCM_PLACR)

The PLACR register selects the arbitration policy for the crossbar masters and configures the flash memory controller.

The speculation buffer and cache in the flash memory controller is configurable via PLACR[15:10].

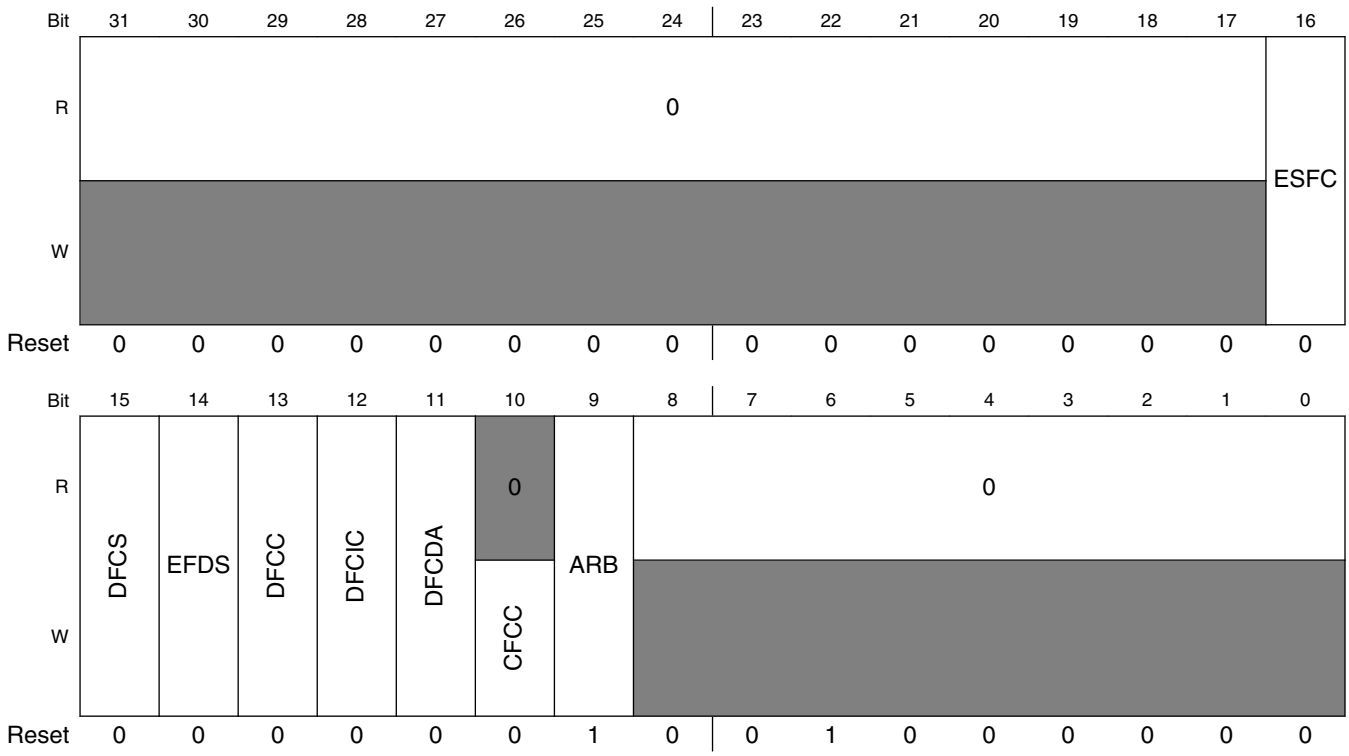
The speculation buffer is enabled only for instructions after reset. It is possible to have these states for the speculation buffer:

DFCS	EFDS	Description
0	0	Speculation buffer is on for instruction and off for data.
0	1	Speculation buffer is on for instruction and on for data.
1	X	Speculation buffer is off.

The cache in flash controller is enabled and caching both instruction and data type fetches after reset. It is possible to have these states for the cache:

DFCC	DFCIC	DFCDA	Description
0	0	0	Cache is on for both instruction and data.
0	0	1	Cache is on for instruction and off for data.
0	1	0	Cache is off for instruction and on for data.
0	1	1	Cache is off for both instruction and data.
1	X	X	Cache is off.

Address: F000_3000h base + Ch offset = F000_300Ch



MCM_PLACR field descriptions

Field	Description
31–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 ESFC	Enable Stalling Flash Controller Enables stalling flash controller when flash is busy. When software needs to access the flash memory while a flash memory resource is being manipulated by a flash command, software can enable a stall mechanism to avoid a read collision. The stall mechanism allows software to execute code from the same block on which flash operations are being performed. However, software must ensure the sector the flash operations are being performed on is not the same sector from which the code is executing. ESFC enables the stall mechanism. This bit must be set only just before the flash operation is executed and must be cleared when the operation completes. 0 Disable stalling flash controller when flash is busy. 1 Enable stalling flash controller when flash is busy.
15 DFCS	Disable Flash Controller Speculation Disables flash controller speculation. 0 Enable flash controller speculation. 1 Disable flash controller speculation.
14 EFDS	Enable Flash Data Speculation Enables flash data speculation.

Table continues on the next page...

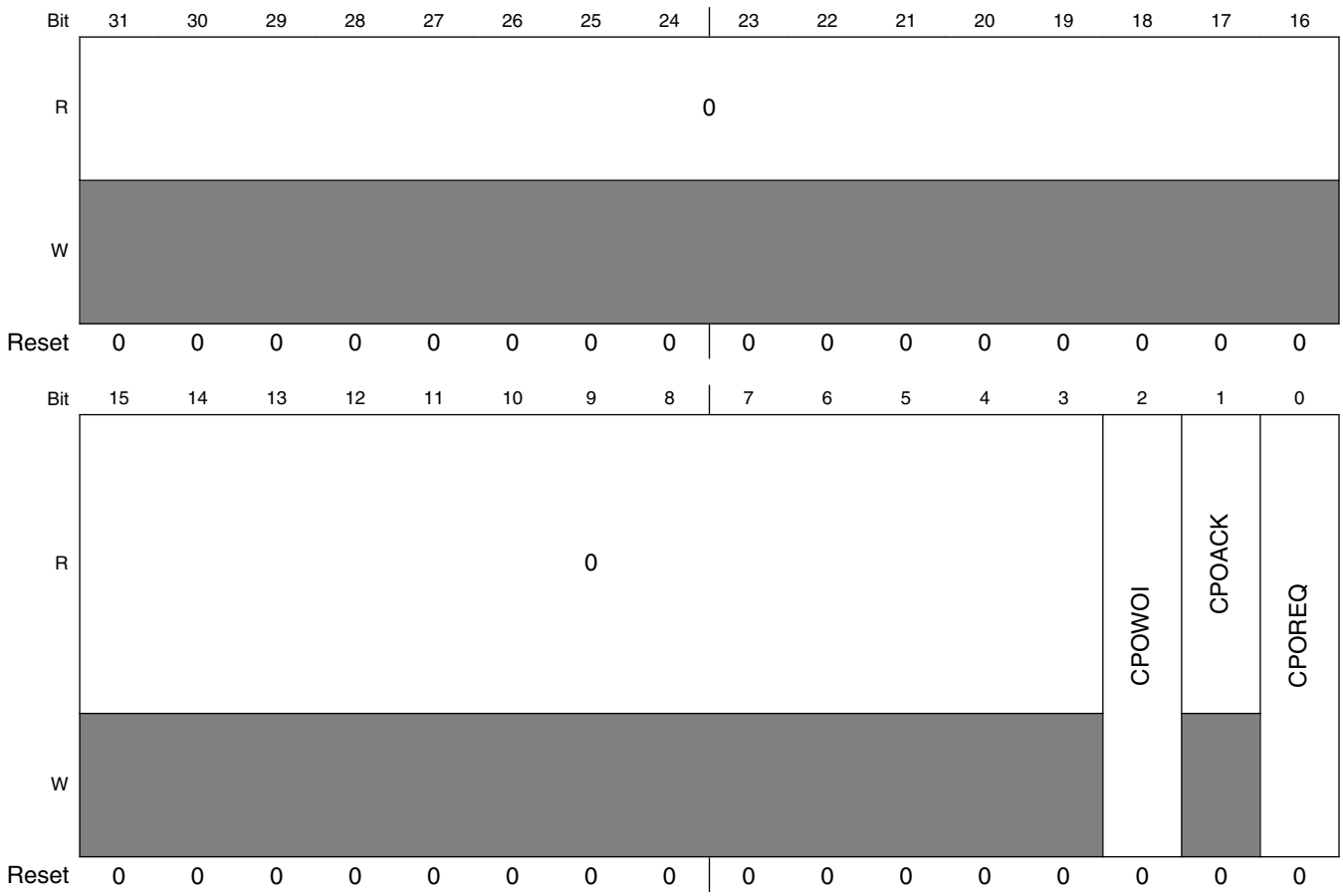
MCM_PLACR field descriptions (continued)

Field	Description
	0 Disable flash data speculation. 1 Enable flash data speculation.
13 DFCC	Disable Flash Controller Cache Disables flash controller cache. 0 Enable flash controller cache. 1 Disable flash controller cache.
12 DFCIC	Disable Flash Controller Instruction Caching Disables flash controller instruction caching. 0 Enable flash controller instruction caching. 1 Disable flash controller instruction caching.
11 DFCDA	Disable Flash Controller Data Caching Disables flash controller data caching. 0 Enable flash controller data caching 1 Disable flash controller data caching.
10 CFCC	Clear Flash Controller Cache Writing a 1 to this field clears the cache. Writing a 0 to this field is ignored. This field always reads as 0.
9 ARB	Arbitration select 0 Fixed-priority arbitration for the crossbar masters 1 Round-robin arbitration for the crossbar masters
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

20.2.4 Compute Operation Control Register (MCM_CPO)

This register controls the Compute Operation.

Address: F000_3000h base + 40h offset = F000_3040h



MCM_CPO field descriptions

Field	Description
31–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 CPOWOI	Compute Operation Wake-up on Interrupt 0 No effect. 1 When set, the CPOREQ is cleared on any interrupt or exception vector fetch.
1 CPOACK	Compute Operation Acknowledge 0 Compute operation entry has not completed or compute operation exit has completed. 1 Compute operation entry has completed or compute operation exit has not completed.
0 CPOREQ	Compute Operation Request This bit is auto-cleared by vector fetching if CPOWOI = 1.

Table continues on the next page...

MCM_CPO field descriptions (continued)

Field	Description
	0 Request is cleared. 1 Request Compute Operation.

Chapter 21
Crossbar-Lite Switch (AXBS-Lite)

21.1 Chip-specific AXBS-Lite information

21.1.1 Crossbar-Light Switch Master Assignments - with System MPU

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

Master port number	Master module	Default Priority
0	ARM core	0
2	DMA	2
3	USB FS/LS OTG	3

21.1.2 Crossbar-Light Switch Slave Assignments - with System MPU

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

Slave module	Slave port number	Protected by FSL MPU?
0	Flash memory controller / Boot ROM	Yes
1	SRAM	Yes
2a	USB SRAM	Yes
2b	Peripheral bridge 0	No. Protection built into bridge
3	QSPI	Yes

21.2 Introduction

The information found here 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 up to four bus masters to access different bus slaves simultaneously, while providing arbitration among the bus masters when they access the same slave.

21.2.1 Features

The crossbar switch includes these features:

- Symmetric crossbar bus switch implementation
  - Allows concurrent accesses from different masters to different slaves
- Up to single-clock 32-bit transfer
- Programmable configuration for fixed-priority or round-robin slave port arbitration (see the chip-specific information).

21.3 Memory Map / Register Definition

This crossbar switch is designed for minimal gate count. It, therefore, has no memory-mapped configuration registers.

Please see the chip-specific information for information on whether the arbitration method in the crossbar switch is programmable, and by which module.

21.4 Functional Description

21.4.1 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.

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.

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 remains parked with the last master to use the slave port. This is done to save the initial clock of arbitration delay that otherwise would be seen if the same master had to arbitrate to gain control of the slave port.

Chapter 22
Peripheral Bridge (AIPS-Lite)

22.1 Chip-specific AIPS-Lite information

22.1.1 Number of peripheral bridges

This device contains one peripheral bridge.

22.1.2 Memory maps

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

22.1.3 AIPS_Lite MPRA register reset value

- AIPS_MPRA reset value is 0x7770_0000

Therefore, masters 0, 1, and 2 are trusted bus masters after reset.

22.1.4 AIPS_Lite PACRE-P register reset values

The AIPS*x*_PACRE-P reset values depend on if the module is available on your particular device. For each populated slot in slots 32-127 in Peripheral bridge 0 (AIPS-Lite 0) memory map the corresponding module's PACR[32:127] field resets to 0x4.

22.2 Introduction

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 peripheral slots of 4 KB. (It might be possible that all the peripheral slots are not used. See the memory map chapter for details on slot assignments.) The bridge includes separate clock enable inputs for each of the slots to accommodate slower peripherals.

22.2.1 Features

Key features of the peripheral bridge are:

- Supports peripheral slots with 8-, 16-, and 32-bit datapath width
- Programming model provides memory protection functionality

22.2.2 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 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.

22.3 Memory map/register definition

The 32-bit peripheral bridge registers can be accessed only in supervisor mode by trusted bus masters. Additionally, these registers must be read from or written to only by a 32-bit aligned access. The peripheral bridge registers are mapped into the Peripheral Access Control Register A PACRA[PACR0] address space.

AIPS memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4000_0000	Master Privilege Register A (AIPS_MPRA)	32	R/W	See section	22.3.1/435
4000_0020	Peripheral Access Control Register (AIPS_PACRA)	32	R/W	See section	22.3.2/438
4000_0024	Peripheral Access Control Register (AIPS_PACRB)	32	R/W	See section	22.3.2/438
4000_0028	Peripheral Access Control Register (AIPS_PACRC)	32	R/W	See section	22.3.2/438
4000_002C	Peripheral Access Control Register (AIPS_PACRD)	32	R/W	See section	22.3.2/438
4000_0040	Peripheral Access Control Register (AIPS_PACRE)	32	R/W	See section	22.3.3/444
4000_0044	Peripheral Access Control Register (AIPS_PACRF)	32	R/W	See section	22.3.3/444
4000_0048	Peripheral Access Control Register (AIPS_PACRG)	32	R/W	See section	22.3.3/444
4000_004C	Peripheral Access Control Register (AIPS_PACRH)	32	R/W	See section	22.3.3/444
4000_0050	Peripheral Access Control Register (AIPS_PACRI)	32	R/W	See section	22.3.3/444
4000_0054	Peripheral Access Control Register (AIPS_PACRJ)	32	R/W	See section	22.3.3/444
4000_0058	Peripheral Access Control Register (AIPS_PACRK)	32	R/W	See section	22.3.3/444
4000_005C	Peripheral Access Control Register (AIPS_PACRL)	32	R/W	See section	22.3.3/444
4000_0060	Peripheral Access Control Register (AIPS_PACRM)	32	R/W	See section	22.3.3/444
4000_0064	Peripheral Access Control Register (AIPS_PACRN)	32	R/W	See section	22.3.3/444
4000_0068	Peripheral Access Control Register (AIPS_PACRO)	32	R/W	See section	22.3.3/444
4000_006C	Peripheral Access Control Register (AIPS_PACRP)	32	R/W	See section	22.3.3/444

22.3.1 Master Privilege Register A (AIPS_MPRA)

The MPRA specifies identical 4-bit fields defining the access-privilege level associated with a bus master to various peripherals on the chip. The register provides one field per bus master.

NOTE

At reset, the default value loaded into the MPRA fields is chip-specific. See the chip configuration details for the value of a particular device.

A register field that maps to an unimplemented master or peripheral behaves as read-only-zero.

Each master is assigned a logical ID from 0 to 15. See the master logical ID assignment table in the chip-specific AIPS information.

Address: 4000_0000h base + 0h offset = 4000_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MTR0	MTW0	MPL0	0	MTR1	MTW1	MPL1	0	MTR2	MTW2	MPL2	0	MTR3	MTW3	MPL3
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	MTR4	MTW4	MPL4	Reserved				Reserved				Reserved			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value is chip-dependent and can be found in the chip-specific AIPS information.

AIPS_MPRA field descriptions

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 MTR0	Master 0 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
29 MTW0	Master 0 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
28 MPL0	Master 0 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 MTR1	Master 1 Trusted for Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
25 MTW1	Master 1 Trusted for Writes Determines whether the master is trusted for write accesses.

Table continues on the next page...

AIPS_MPRA field descriptions (continued)

Field	Description
	0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
24 MPL1	Master 1 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 MTR2	Master 2 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
21 MTW2	Master 2 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
20 MPL2	Master 2 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 MTR3	Master 3 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
17 MTW3	Master 3 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
16 MPL3	Master 3 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

AIPS_MPRA field descriptions (continued)

Field	Description
14 MTR4	Master 4 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
13 MTW4	Master 4 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
12 MPL4	Master 4 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
11–8 Reserved	This field is reserved.
7–4 Reserved	This field is reserved.
Reserved	This field is reserved.

22.3.2 Peripheral Access Control Register (AIPS_PACR*n*)

Each PACR register consists of eight 4-bit PACR fields. Each PACR field defines the access levels for a particular peripheral. The mapping between a peripheral and its PACR field is shown in the table below. The peripheral assignment to each PACR is defined by the memory map slot that the peripheral is assigned to. See this chip's memory map for the assignment of a particular peripheral.

The following table shows the location of each peripheral slot's PACR field in the PACR registers.

Offset	Register	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x20	PACRA	PACR0	PACR1	PACR2	PACR3	PACR4	PACR5	PACR6	PACR7
0x24	PACRB	PACR8	PACR9	PACR10	PACR11	PACR12	PACR13	PACR14	PACR15
0x28	PACRC	PACR16	PACR17	PACR18	PACR19	PACR20	PACR21	PACR22	PACR23
0x2C	PACRD	PACR24	PACR25	PACR26	PACR27	PACR28	PACR29	PACR30	PACR31
0x30	Reserved								
0x34	Reserved								
0x38	Reserved								
0x3C	Reserved								

Table continues on the next page...

Offset	Register	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x40	PACRE	PACR32	PACR33	PACR34	PACR35	PACR36	PACR37	PACR38	PACR39
0x44	PACRF	PACR40	PACR41	PACR42	PACR43	PACR44	PACR45	PACR46	PACR47
0x48	PACRG	PACR48	PACR49	PACR50	PACR51	PACR52	PACR53	PACR54	PACR55
0x4C	PACRH	PACR56	PACR57	PACR58	PACR59	PACR60	PACR61	PACR62	PACR63
0x50	PACRI	PACR64	PACR65	PACR66	PACR67	PACR68	PACR69	PACR70	PACR71
0x54	PACRJ	PACR72	PACR73	PACR74	PACR75	PACR76	PACR77	PACR78	PACR79
0x58	PACRK	PACR80	PACR81	PACR82	PACR83	PACR84	PACR85	PACR86	PACR87
0x5C	PACRL	PACR88	PACR89	PACR90	PACR91	PACR92	PACR93	PACR94	PACR95
0x60	PACRM	PACR96	PACR97	PACR98	PACR99	PACR100	PACR101	PACR102	PACR103
0x64	PACRN	PACR104	PACR105	PACR106	PACR107	PACR108	PACR109	PACR110	PACR111
0x68	PACRO	PACR112	PACR113	PACR114	PACR115	PACR116	PACR117	PACR118	PACR119
0x6C	PACRP	PACR120	PACR121	PACR122	PACR123	PACR124	PACR125	PACR126	PACR127

NOTE

The register field descriptions for PACR A-D, which control peripheral slots 0-31, are shown below. The following section, Peripheral Access Control Register (AIPS_PACR*n*), shows the register field descriptions for PACR E-P. All PACR registers are identical. They are divided into two sections because they occupy two non-contiguous address spaces.

Address: 4000_0000h base + 20h 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	SP0	WP0	TP0	0	SP1	WP1	TP1	0	SP2	WP2	TP2	0	SP3	WP3	TP3
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	SP4	WP4	TP4	0	SP5	WP5	TP5	0	SP6	WP6	TP6	0	SP7	WP7	TP7
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value is chip-dependent and can be found in the AIPS chip-specific information.

AIPS_PACR*n* field descriptions

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 SP0	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR*x*[MPL*n*] control field

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
	for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
29 WP0	Write Protect Determines whether the peripheral allows write accesss. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
28 TP0	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 SP1	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR*x*[MPL*n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
25 WP1	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
24 TP1	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 SP2	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR*x*[MPL*n*] control field

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
	for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
21 WP2	Write Protect Determines whether the peripheral allows write accesss. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
20 TP2	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 SP3	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this bit is set, the master privilege level must indicate the supervisor access attribute, and the MPR*x*[MPL*n*] control bit for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
17 WP3	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
16 TP3	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 SP4	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR*x*[MPL*n*] control field

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
	for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
13 WP4	Write Protect Determines whether the peripheral allows write accesss. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
12 TP4	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 SP5	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR*x*[MPL*n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
9 WP5	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
8 TP5	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 SP6	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR*x*[MPL*n*] control field

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
	for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
5 WP6	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
4 TP6	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 SP7	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR*x*[MPL*n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
1 WP7	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
0 TP7	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

22.3.3 Peripheral Access Control Register (AIPS_PACR*n*)

This section describes PACR registers E-P, which control peripheral slots 32-127. See Peripheral Access Control Register (AIPS_PACR*n*) for the description of these registers.

Address: 4000_0000h base + 40h 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	0	SP0	WP0	TP0	0	SP1	WP1	TP1	0	SP2	WP2	TP2	0	SP3	WP3	TP3
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	SP4	WP4	TP4	0	SP5	WP5	TP5	0	SP6	WP6	TP6	0	SP7	WP7	TP7
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value is chip-dependent and can be found in the AIPS chip-specific information.

AIPS_PACR*n* field descriptions

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 SP0	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR*x*[MPL*n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
29 WP0	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
28 TP0	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 SP1	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR*x*[MPL*n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
25 WP1	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
24 TP1	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 SP2	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this bit is set, the master privilege level must indicate the supervisor access attribute, and the MPR*x*[MPL*n*] control bit for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
21 WP2	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
20 TP2	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 SP3	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR*x*[MPL*n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
17 WP3	Write Protect Determines whether the peripheral allows write accesss. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
16 TP3	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 SP4	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this bit is set, the master privilege level must indicate the supervisor access attribute, and the MPR*x*[MPL*n*] control bit for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
13 WP4	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
12 TP4	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 SP5	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR*x*[MPL*n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
9 WP5	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
8 TP5	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 SP6	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR*x*[MPL*n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
5 WP6	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
4 TP6	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 SP7	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR*x*[MPL*n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
1 WP7	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
0 TP7	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

22.4 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.

22.4.1 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 23
Memory Protection Unit (MPU)

23.1 Chip-specific MPU information

23.1.1 MPU slave port assignments

The memory-mapped resources protected by the MPU are:

Table 23-1. MPU slave port assignments

Source	MPU slave port assignment	Destination
Crossbar slave port 0	MPU slave port 0	Flash, boot ROM
Crossbar slave port 1	MPU slave port 1	SRAM, MTB, MTBDWT, ROM, MCM
Crossbar slave port 2	MPU slave port 2	USB RAM
Crossbar slave port 3	MPU slave port 3	QSPI

23.1.2 MPU logical bus master assignments

The logical bus master assignments for the MPU are:

Table 23-2. MPU logical bus master assignments

MPU logical bus master number	Bus master
0	Core
1	Debugger
2	DMA
3	USB

23.1.3 MPU access violation indications

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

Table 23-3. Access violation indications

Bus master	Core indication
Core	MPU violations result in a hard fault (interrupt vector #3).
Debugger	The STICKYERROR flag is set in the Debug Port Control/Status Register.
DMA	DMA Error Interrupt vector
USB_OTG	USB Interrupt vector

23.1.4 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 23-4. Reset values for RGD0 registers

Register	Reset value
RGD0_WORD0	0000_0000h
RGD0_WORD1	FFFF_FFFFh
RGD0_WORD2	0001_F01Fh
RGD0_WORD3	0000_0001h
RGDAAC0	0001_F01Fh

23.1.5 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 23-5. Write Access to RGD0 Registers

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

23.2 Introduction

The memory protection unit (MPU) provides hardware access control for all memory references generated in the device.

23.3 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.

23.3.1 Block diagram

A simplified block diagram of the MPU module is shown in the following figure.

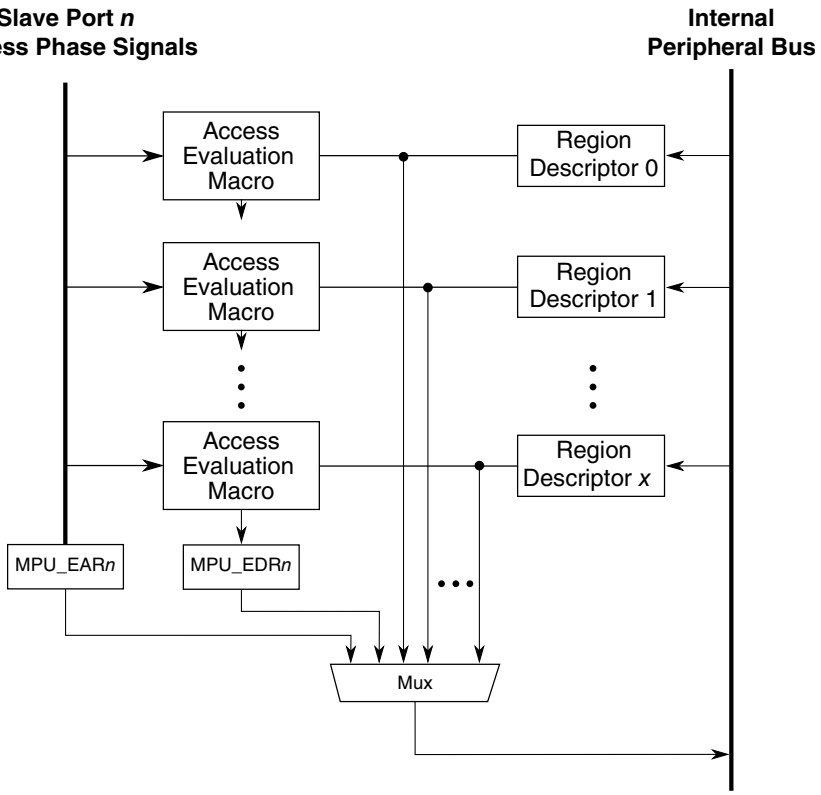


Figure 23-1. MPU block diagram

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.

23.3.2 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:

- 8 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

23.4 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 in 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_4001h	23.4.1/457
4000_D010	Error Address Register, slave port n (MPU_EAR0)	32	R	0000_0000h	23.4.2/459
4000_D014	Error Detail Register, slave port n (MPU_EDR0)	32	R	0000_0000h	23.4.3/459
4000_D018	Error Address Register, slave port n (MPU_EAR1)	32	R	0000_0000h	23.4.2/459
4000_D01C	Error Detail Register, slave port n (MPU_EDR1)	32	R	0000_0000h	23.4.3/459
4000_D020	Error Address Register, slave port n (MPU_EAR2)	32	R	0000_0000h	23.4.2/459
4000_D024	Error Detail Register, slave port n (MPU_EDR2)	32	R	0000_0000h	23.4.3/459
4000_D028	Error Address Register, slave port n (MPU_EAR3)	32	R	0000_0000h	23.4.2/459
4000_D02C	Error Detail Register, slave port n (MPU_EDR3)	32	R	0000_0000h	23.4.3/459
4000_D030	Error Address Register, slave port n (MPU_EAR4)	32	R	0000_0000h	23.4.2/459
4000_D034	Error Detail Register, slave port n (MPU_EDR4)	32	R	0000_0000h	23.4.3/459
4000_D400	Region Descriptor n, Word 0 (MPU_RGD0_WORD0)	32	R/W	0000_0000h	23.4.4/460
4000_D404	Region Descriptor n, Word 1 (MPU_RGD0_WORD1)	32	R/W	See section	23.4.5/461
4000_D408	Region Descriptor n, Word 2 (MPU_RGD0_WORD2)	32	R/W	See section	23.4.6/461
4000_D40C	Region Descriptor n, Word 3 (MPU_RGD0_WORD3)	32	R/W	0000_0000h	23.4.7/464
4000_D410	Region Descriptor n, Word 0 (MPU_RGD1_WORD0)	32	R/W	0000_0000h	23.4.4/460
4000_D414	Region Descriptor n, Word 1 (MPU_RGD1_WORD1)	32	R/W	See section	23.4.5/461
4000_D418	Region Descriptor n, Word 2 (MPU_RGD1_WORD2)	32	R/W	See section	23.4.6/461
4000_D41C	Region Descriptor n, Word 3 (MPU_RGD1_WORD3)	32	R/W	0000_0000h	23.4.7/464
4000_D420	Region Descriptor n, Word 0 (MPU_RGD2_WORD0)	32	R/W	0000_0000h	23.4.4/460
4000_D424	Region Descriptor n, Word 1 (MPU_RGD2_WORD1)	32	R/W	See section	23.4.5/461
4000_D428	Region Descriptor n, Word 2 (MPU_RGD2_WORD2)	32	R/W	See section	23.4.6/461
4000_D42C	Region Descriptor n, Word 3 (MPU_RGD2_WORD3)	32	R/W	0000_0000h	23.4.7/464
4000_D430	Region Descriptor n, Word 0 (MPU_RGD3_WORD0)	32	R/W	0000_0000h	23.4.4/460
4000_D434	Region Descriptor n, Word 1 (MPU_RGD3_WORD1)	32	R/W	See section	23.4.5/461
4000_D438	Region Descriptor n, Word 2 (MPU_RGD3_WORD2)	32	R/W	See section	23.4.6/461
4000_D43C	Region Descriptor n, Word 3 (MPU_RGD3_WORD3)	32	R/W	0000_0000h	23.4.7/464
4000_D440	Region Descriptor n, Word 0 (MPU_RGD4_WORD0)	32	R/W	0000_0000h	23.4.4/460
4000_D444	Region Descriptor n, Word 1 (MPU_RGD4_WORD1)	32	R/W	See section	23.4.5/461
4000_D448	Region Descriptor n, Word 2 (MPU_RGD4_WORD2)	32	R/W	See section	23.4.6/461
4000_D44C	Region Descriptor n, Word 3 (MPU_RGD4_WORD3)	32	R/W	0000_0000h	23.4.7/464
4000_D450	Region Descriptor n, Word 0 (MPU_RGD5_WORD0)	32	R/W	0000_0000h	23.4.4/460
4000_D454	Region Descriptor n, Word 1 (MPU_RGD5_WORD1)	32	R/W	See section	23.4.5/461
4000_D458	Region Descriptor n, Word 2 (MPU_RGD5_WORD2)	32	R/W	See section	23.4.6/461
4000_D45C	Region Descriptor n, Word 3 (MPU_RGD5_WORD3)	32	R/W	0000_0000h	23.4.7/464
4000_D460	Region Descriptor n, Word 0 (MPU_RGD6_WORD0)	32	R/W	0000_0000h	23.4.4/460
4000_D464	Region Descriptor n, Word 1 (MPU_RGD6_WORD1)	32	R/W	See section	23.4.5/461
4000_D468	Region Descriptor n, Word 2 (MPU_RGD6_WORD2)	32	R/W	See section	23.4.6/461

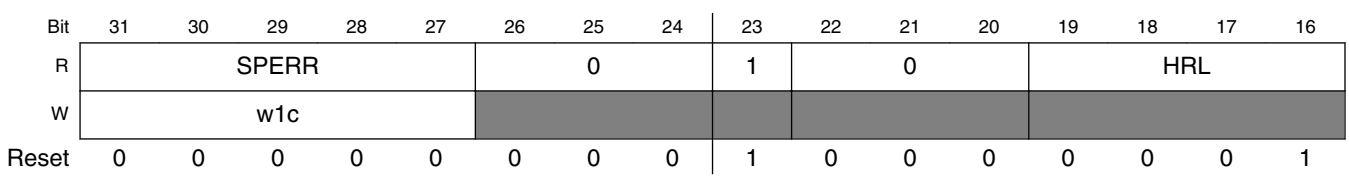
Table continues on the next page...

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	0000_0000h	23.4.7/464
4000_D470	Region Descriptor n, Word 0 (MPU_RGD7_WORD0)	32	R/W	0000_0000h	23.4.4/460
4000_D474	Region Descriptor n, Word 1 (MPU_RGD7_WORD1)	32	R/W	See section	23.4.5/461
4000_D478	Region Descriptor n, Word 2 (MPU_RGD7_WORD2)	32	R/W	See section	23.4.6/461
4000_D47C	Region Descriptor n, Word 3 (MPU_RGD7_WORD3)	32	R/W	0000_0000h	23.4.7/464
4000_D800	Region Descriptor Alternate Access Control n (MPU_RGDAAC0)	32	R/W	See section	23.4.8/465
4000_D804	Region Descriptor Alternate Access Control n (MPU_RGDAAC1)	32	R/W	See section	23.4.8/465
4000_D808	Region Descriptor Alternate Access Control n (MPU_RGDAAC2)	32	R/W	See section	23.4.8/465
4000_D80C	Region Descriptor Alternate Access Control n (MPU_RGDAAC3)	32	R/W	See section	23.4.8/465
4000_D810	Region Descriptor Alternate Access Control n (MPU_RGDAAC4)	32	R/W	See section	23.4.8/465
4000_D814	Region Descriptor Alternate Access Control n (MPU_RGDAAC5)	32	R/W	See section	23.4.8/465
4000_D818	Region Descriptor Alternate Access Control n (MPU_RGDAAC6)	32	R/W	See section	23.4.8/465
4000_D81C	Region Descriptor Alternate Access Control n (MPU_RGDAAC7)	32	R/W	See section	23.4.8/465

23.4.1 Control/Error Status Register (MPU_CESR)

Address: 4000_D000h base + 0h offset = 4000_D000h



MPU_CESR field descriptions

Field	Description
31–27 SPERR	Slave Port n Error 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

Table continues on the next page...

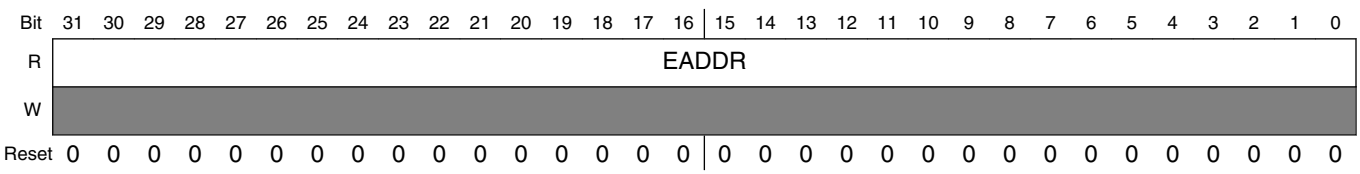
MPU_CESR field descriptions (continued)

Field	Description
	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. The following shows the correspondence between the bit number and slave port number: • Bit 31 corresponds to slave port 0. • Bit 30 corresponds to slave port 1. • Bit 29 corresponds to slave port 2. • Bit 28 corresponds to slave port 3. • Bit 27 corresponds to slave port 4. 0 No error has occurred for slave port n. 1 An error has occurred for slave port n.
26–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
22–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–16 HRL	Hardware Revision Level 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 NSP	Number Of Slave Ports Specifies the number of slave ports connected to the MPU.
11–8 NRGD	Number Of Region Descriptors Indicates the number of region descriptors implemented in the MPU. 0000 8 region descriptors 0001 12 region descriptors 0010 16 region descriptors
7–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 VLD	Valid Global enable/disable for the MPU. 0 MPU is disabled. All accesses from all bus masters are allowed. 1 MPU is enabled

23.4.2 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



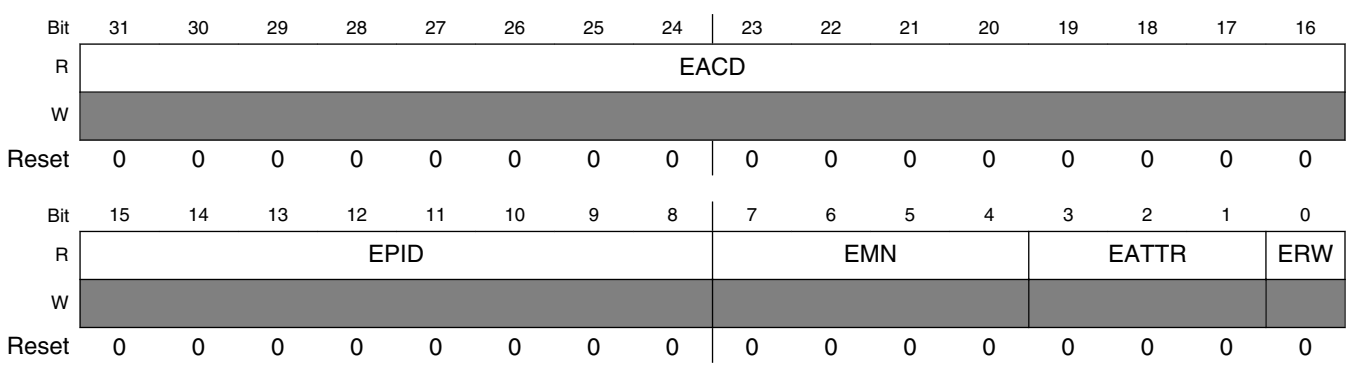
MPU_EAR*n* field descriptions

Field	Description
EADDR	Error Address Indicates the reference address from slave port n that generated the access error

23.4.3 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



MPU_EDR*n* field descriptions

Field	Description
31–16 EACD	Error Access Control Detail Indicates the region descriptor with the access error. • If EDRn contains a captured error and EACD is cleared, an access did not hit in any region descriptor. • If only a single EACD bit is set, the protection error was caused by a single non-overlapping region descriptor. • If two or more EACD bits are set, the protection error was caused by an overlapping set of region descriptors.
15–8 EPID	Error Process Identification 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.
7–4 EMN	Error Master Number Indicates the bus master that generated the access error.
3–1 EATTR	Error Attributes Indicates attribute information about the faulting reference. **NOTE:** All other encodings are reserved. 000 User mode, instruction access 001 User mode, data access 010 Supervisor mode, instruction access 011 Supervisor mode, data access
0 ERW	Error Read/Write Indicates the access type of the faulting reference. 0 Read 1 Write

23.4.4 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 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	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 SRTADDR	Start Address Defines the most significant bits of the 0-modulo-32 byte start address of the memory region.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

23.4.5 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 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	ENDADDR																											Reserved				
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*	1*	1*	1*	1*	1*

* Notes:
- Reset value of RGD0_WORD1 is FFFF_FFFFh

  Reset value of RGD[1:7]_WORD1 is 0000_001Fh

MPU_RGD*n*_WORD1 field descriptions

Field	Description
31–5 ENDADDR	End Address Defines the most significant bits of the 31-modulo-32 byte end address of the memory region. **NOTE:** The MPU does not verify that ENDADDR ≥ SRTADDR.
Reserved	This field is reserved.

23.4.6 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 7d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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*	0*

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	M2SM	M2UM			M1PE	M1SM		M1UM			M0PE	M0SM		M0UM		
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- Reset value of RGD0_WORD2 is 0001_F01Fh

  Reset value of RGD[1:7]_WORD2 is 0000_0000h

MPU_RGD*n*_WORD2 field descriptions

Field	Description
31 M7RE	Bus Master 7 Read Enable 0 Bus master 7 reads terminate with an access error and the read is not performed 1 Bus master 7 reads allowed
30 M7WE	Bus Master 7 Write Enable 0 Bus master 7 writes terminate with an access error and the write is not performed 1 Bus master 7 writes allowed
29 M6RE	Bus Master 6 Read Enable 0 Bus master 6 reads terminate with an access error and the read is not performed 1 Bus master 6 reads allowed

Table continues on the next page...

MPU_RGD*n*_WORD2 field descriptions (continued)

Field	Description
28 M6WE	Bus Master 6 Write Enable 0 Bus master 6 writes terminate with an access error and the write is not performed 1 Bus master 6 writes allowed
27 M5RE	Bus Master 5 Read Enable 0 Bus master 5 reads terminate with an access error and the read is not performed 1 Bus master 5 reads allowed
26 M5WE	Bus Master 5 Write Enable 0 Bus master 5 writes terminate with an access error and the write is not performed 1 Bus master 5 writes allowed
25 M4RE	Bus Master 4 Read Enable 0 Bus master 4 reads terminate with an access error and the read is not performed 1 Bus master 4 reads allowed
24 M4WE	Bus Master 4 Write Enable 0 Bus master 4 writes terminate with an access error and the write is not performed 1 Bus master 4 writes allowed
23 M3PE	Bus Master 3 Process Identifier Enable 0 Do not include the process identifier in the evaluation 1 Include the process identifier and mask (RGDn_WORD3) in the region hit evaluation
22–21 M3SM	Bus Master 3 Supervisor Mode Access Control Defines the access controls for bus master 3 in Supervisor mode. 00 r/w/x; read, write and execute allowed 01 r/x; read and execute allowed, but no write 10 r/w; read and write allowed, but no execute 11 Same as User mode defined in M3UM
20–18 M3UM	Bus Master 3 User Mode Access Control 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. In M3UM[2:0]: M3UM[2] controls read permissions, M3UM[1] controls write permissions, and M3UM[0] controls execute permissions. 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. 1 Allows the given access type to occur
17 M2PE	Bus Master 2 Process Identifier Enable See M3PE description.
16–15 M2SM	Bus Master 2 Supervisor Mode Access Control See M3SM description.
14–12 M2UM	Bus Master 2 User Mode Access control See M3UM description.

Table continues on the next page...

MPU_RGD*n*_WORD2 field descriptions (continued)

Field	Description
11 M1PE	Bus Master 1 Process Identifier enable See M3PE description.
10–9 M1SM	Bus Master 1 Supervisor Mode Access Control See M3SM description.
8–6 M1UM	Bus Master 1 User Mode Access Control See M3UM description.
5 M0PE	Bus Master 0 Process Identifier enable See M0PE description.
4–3 M0SM	Bus Master 0 Supervisor Mode Access Control See M3SM description.
M0UM	Bus Master 0 User Mode Access Control See M3UM description.

23.4.7 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 7d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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	0

MPU_RGD*n*_WORD3 field descriptions

Field	Description
31–24 PID	Process Identifier 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 PIDMASK	Process Identifier Mask 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

Table continues on the next page...

MPU_RGD*n*_WORD3 field descriptions (continued)

Field	Description
	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.”
15–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 VLD	Valid Signals the region descriptor is valid. Any write to RGDn_WORD0–2 clears this bit. 0 Region descriptor is invalid 1 Region descriptor is valid

23.4.8 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 7d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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*	0*

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	M2SM	M2UM			M1PE	M1SM			M1UM		M0PE	M0SM			M0UM	
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- Reset value of RGDAAC0 is

  Reset value of RGDAAC[1:7] is 0000_0000h

MPU_RGDAAC*n* field descriptions

Field	Description
31 M7RE	Bus Master 7 Read Enable

Table continues on the next page...

MPU_RGDAAC*n* field descriptions (continued)

Field	Description
	0 Bus master 7 reads terminate with an access error and the read is not performed 1 Bus master 7 reads allowed
30 M7WE	Bus Master 7 Write Enable 0 Bus master 7 writes terminate with an access error and the write is not performed 1 Bus master 7 writes allowed
29 M6RE	Bus Master 6 Read Enable 0 Bus master 6 reads terminate with an access error and the read is not performed 1 Bus master 6 reads allowed
28 M6WE	Bus Master 6 Write Enable 0 Bus master 6 writes terminate with an access error and the write is not performed 1 Bus master 6 writes allowed
27 M5RE	Bus Master 5 Read Enable 0 Bus master 5 reads terminate with an access error and the read is not performed 1 Bus master 5 reads allowed
26 M5WE	Bus Master 5 Write Enable 0 Bus master 5 writes terminate with an access error and the write is not performed 1 Bus master 5 writes allowed
25 M4RE	Bus Master 4 Read Enable 0 Bus master 4 reads terminate with an access error and the read is not performed 1 Bus master 4 reads allowed
24 M4WE	Bus Master 4 Write Enable 0 Bus master 4 writes terminate with an access error and the write is not performed 1 Bus master 4 writes allowed
23 M3PE	Bus Master 3 Process Identifier Enable 0 Do not include the process identifier in the evaluation 1 Include the process identifier and mask (RGDn.RGDAAC) in the region hit evaluation
22–21 M3SM	Bus Master 3 Supervisor Mode Access Control Defines the access controls for bus master 3 in Supervisor mode. 00 r/w/x; read, write and execute allowed 01 r/x; read and execute allowed, but no write 10 r/w; read and write allowed, but no execute 11 Same as User mode defined in M3UM
20–18 M3UM	Bus Master 3 User Mode Access Control 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. In M3UM[2:0]: M3UM[2] controls read permissions, M3UM[1] controls write permissions, and M3UM[0] controls execute permissions.

Table continues on the next page...

MPU_RGDAAC*n* field descriptions (continued)

Field	Description
	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. 1 Allows the given access type to occur
17 M2PE	Bus Master 2 Process Identifier Enable See M3PE description.
16–15 M2SM	Bus Master 2 Supervisor Mode Access Control See M3SM description.
14–12 M2UM	Bus Master 2 User Mode Access Control See M3UM description.
11 M1PE	Bus Master 1 Process Identifier Enable See M3PE description.
10–9 M1SM	Bus Master 1 Supervisor Mode Access Control See M3SM description.
8–6 M1UM	Bus Master 1 User Mode Access Control See M3UM description.
5 M0PE	Bus Master 0 Process Identifier Enable See M3PE description.
4–3 M0SM	Bus Master 0 Supervisor Mode Access Control See M3SM description.
M0UM	Bus Master 0 User Mode Access Control See M3UM description.

23.5 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.

23.5.1 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.

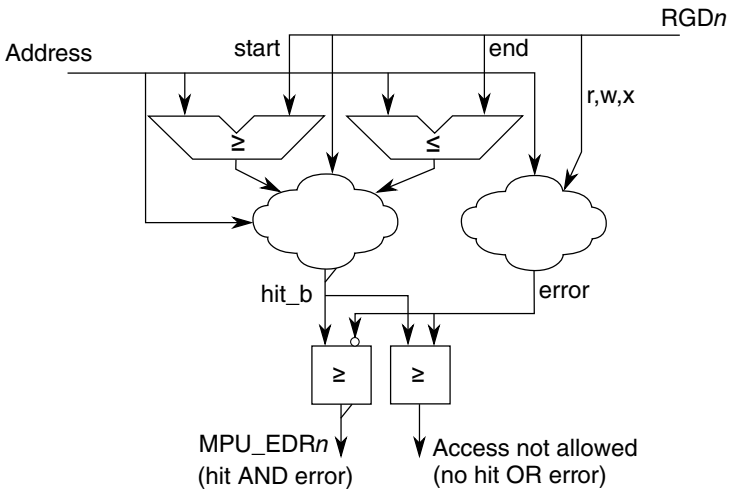


Figure 23-2. MPU access evaluation macro

23.5.1.1 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.

23.5.1.2 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 23-6. 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

23.5.2 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.

23.5.3 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.

23.6 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.

23.7 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 23-7. 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 24
Bit Manipulation Engine (BME)

24.1 Introduction

The Bit Manipulation Engine (BME) provides hardware support for atomic read-modify-write memory operations to the peripheral address space in Cortex-M0+ based microcontrollers.

This architectural capability is also known as "decorated storage" as it defines a mechanism for providing additional semantics for load and store operations to memory-mapped peripherals beyond just the reading and writing of data values to the addressed memory locations. In the BME definition, the "decoration", that is, the additional semantic information, is encoded into the peripheral address used to reference the memory.

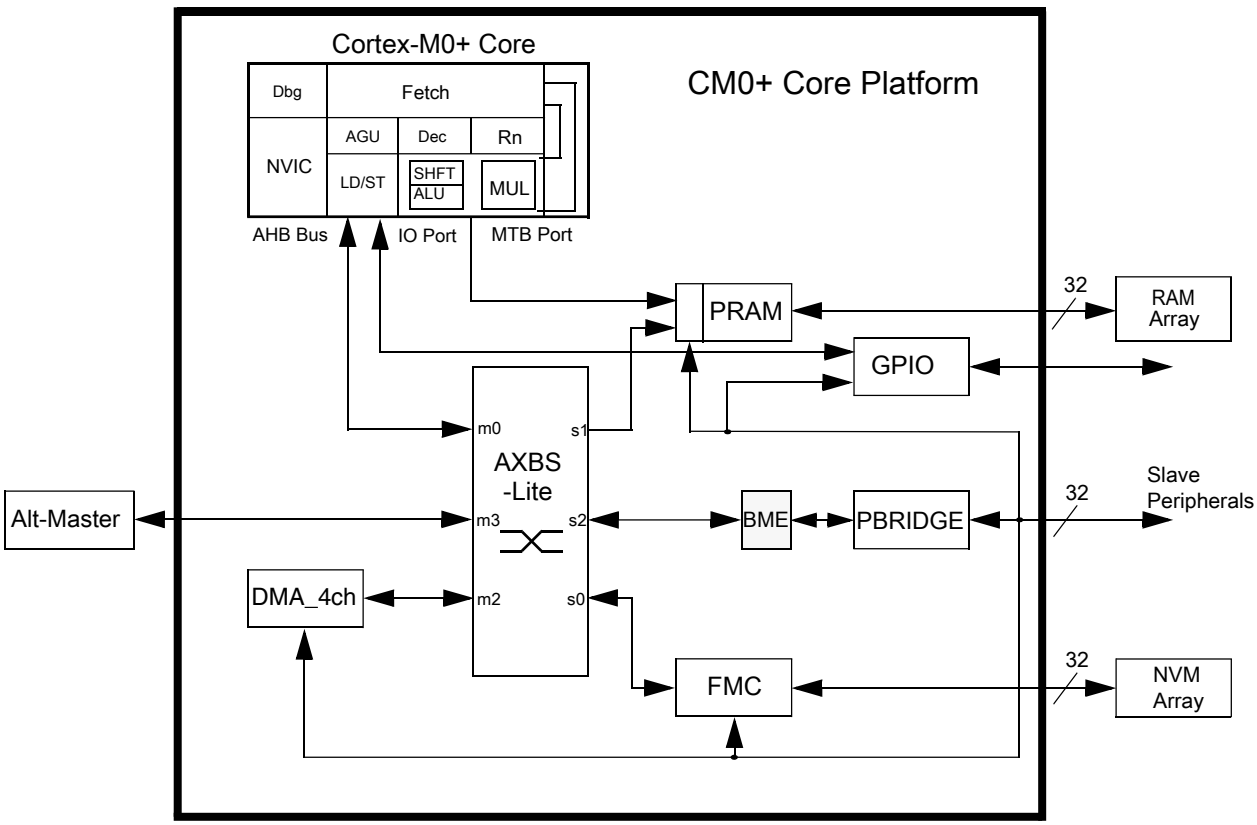
By combining the basic load and store instructions of the ARM Cortex-M instruction set architecture (v6M, v7M) with the concept of decorated storage provided by the BME, the resulting implementation provides a robust and efficient read-modify-write capability to this class of ultra low-end microcontrollers. The resulting architectural capability defined by this core platform function is targeted at the manipulation of n-bit fields in peripheral registers and is consistent with I/O hardware addressing in the Embedded C standard. For most BME commands, a single core read or write bus cycle is converted into an atomic read-modify-write, that is, an indivisible "read followed by a write" bus sequence.

BME decorated references are only available on system bus transactions generated by the processor core and targeted at the standard 512 KB peripheral address space based at 0x4000_0000[1]. The decoration semantic is embedded into address bits[28:19], creating a 448 MB space at addresses 0x4400_0000–0x5FFF_FFFF for AIPS; these bits are stripped out of the actual address sent to the peripheral bus controller and used by the BME to define and control its operation.

1. To be perfectly accurate, the peripheral address space occupies a 516 KB region: 512 KB based at 0x4000_0000 plus a 4 KB space based at 0x400F_F000 for GPIO accesses. This organization provides compatibility with the Kinetis K Family. Attempted accesses to the memory space located between 0x4008_0000 - 0x400F_EFFF are error terminated due to an illegal address.

24.1.1 Overview

The following figure is a generic block diagram of the processor core and platform for this class of ultra low-end microcontrollers.



Note: BME can be accessed only by the core.

Figure 24-1. Cortex-M0+ core platform block diagram

As shown in the block diagram, the BME module interfaces to a crossbar switch AHB slave port as its primary input and sources an AHB bus output to the Peripheral Bridge (PBRIDGE) controller. The BME hardware microarchitecture is a 2-stage pipeline design matching the protocol of the AMBA-AHB system bus interfaces. The PBRIDGE module converts the AHB system bus protocol into the IPS/APB protocol used by the attached slave peripherals.

24.1.2 Features

The key features of the BME include:

- Lightweight implementation of decorated storage for selected address spaces

- Additional access semantics encoded into the reference address
- Resides between a crossbar switch slave port and a peripheral bridge bus controller
- Two-stage pipeline design matching the AHB system bus protocol
- Combinationally passes non-decorated accesses to peripheral bridge bus controller
- Conversion of decorated loads and stores from processor core into atomic read-modify-writes
- Decorated loads support unsigned bit field extracts, load-and-{set,clear} 1-bit operations
- Decorated stores support bit field inserts, logical AND, OR, and XOR operations
- Support for byte, halfword and word-sized decorated operations
- Supports minimum signal toggling on AHB output bus to reduce power dissipation

24.1.3 Modes of operation

The BME module does not support any special modes of operation. As a memory-mapped device located on a crossbar slave AHB system bus port, BME responds strictly on the basis of memory addresses for accesses to the peripheral bridge bus controller.

All functionality associated with the BME module resides in the core platform's clock domain; this includes its connections with the crossbar slave port and the PBRIDGE bus controller.

24.2 Memory map and register definition

The BME module provides a memory-mapped capability and does not include any programming model registers.

The exact set of functions supported by the BME are detailed in the Functional description.

The peripheral address space occupies a 516 KB region: 512 KB based at 0x4000_0000 plus a 4 KB space based at 0x400F_F000 for GPIO accesses; the decorated address space is mapped to the 448 MB region located at 0x4400_0000–0x5FFF_FFFF.

24.3 Functional description

Information found here details the specific functions supported by the BME.

Recall the combination of the basic load and store instructions of the Cortex-M instruction set architecture (v6M, v7M) plus the concept of decorated storage provided by the BME, the resulting implementation provides a robust and efficient read-modify-write capability to this class of ultra low-end microcontrollers. The resulting architectural capability defined by this core platform function is targeted at the manipulation of n-bit fields in peripheral registers and is consistent with I/O hardware addressing in the Embedded C standard. For most BME commands, a single core read or write bus cycle is converted into an atomic read-modify-write, that is, an indivisible "read followed by a write" bus sequence.

Consider decorated store operations first, then decorated loads.

24.3.1 BME decorated stores

The functions supported by the BME's decorated stores include three logical operators (AND, OR, XOR) plus a bit field insert.

For all these operations, BME converts a single decorated AHB store transaction into a 2-cycle atomic read-modify-write sequence, where the combined read-modify operation is performed in the first AHB data phase, and then the write is performed in the second AHB data phase.

A generic timing diagram of a decorated store showing a peripheral bit field insert operation is shown as follows:

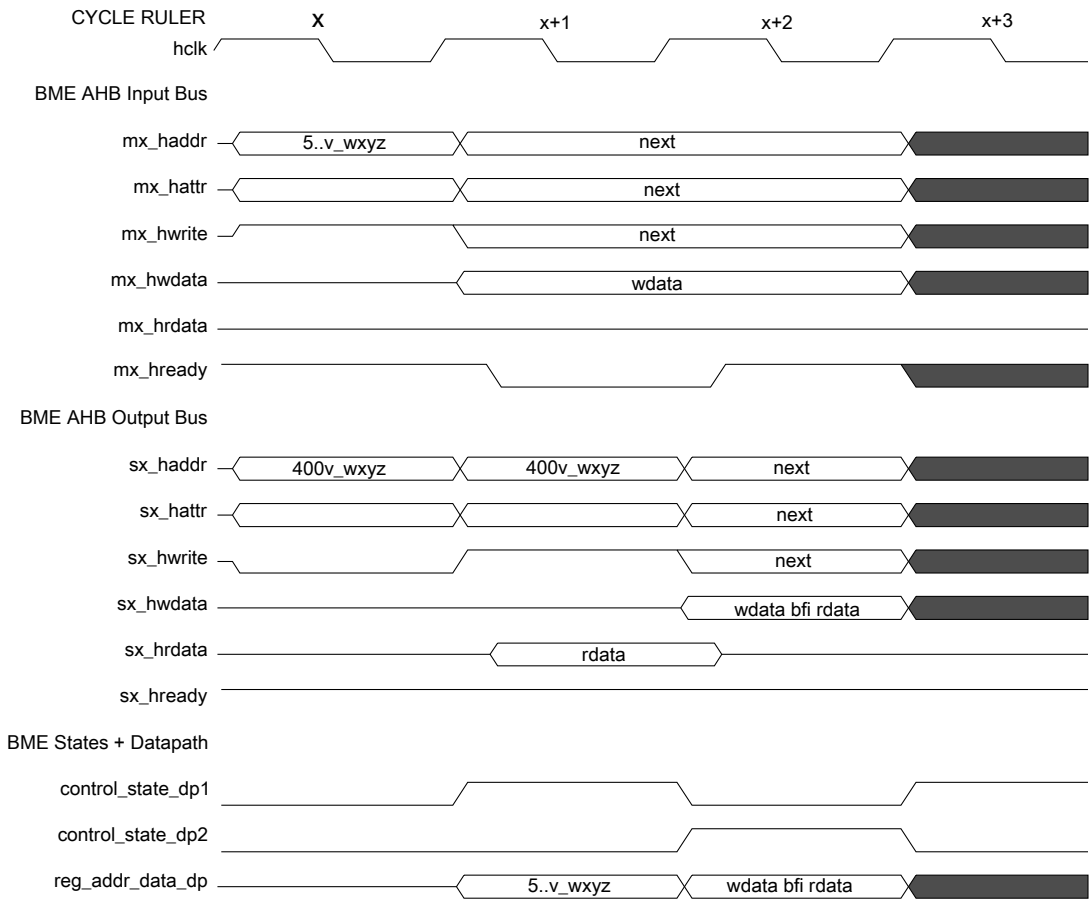


Figure 24-2. Decorated store: bit field insert timing diagram

All the decorated store operations follow the same execution template shown in Figure 24-2, a two-cycle read-modify-write operation:

1. Cycle x, 1st AHB address phase: Write from input bus is translated into a read operation on the output bus using the actual memory address (with the decoration removed) and then captured in a register.
2. Cycle x+1, 2nd AHB address phase: Write access with the registered (but actual) memory address is output
3. Cycle x+1, 1st AHB data phase: Memory read data is modified using the input bus write data and the function defined by the decoration and captured in a data register; the input bus cycle is stalled.
4. Cycle x+2, 2nd AHB data phase: Registered write data is sourced onto the output write data bus.

NOTE

Any wait states inserted by the slave device are simply passed through the BME back to the master input bus, stalling the AHB transaction cycle for cycle.

24.3.1.1 Decorated store logical AND (AND)

This command performs an atomic read-modify-write of the referenced memory location.

1. First, the location is read;
2. It is then modified by performing a logical AND operation using the write data operand sourced for the system bus cycle
3. Finally, the result of the AND operation is written back into the referenced memory location.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit). The core performs the required write data lane replication on byte and halfword transfers.

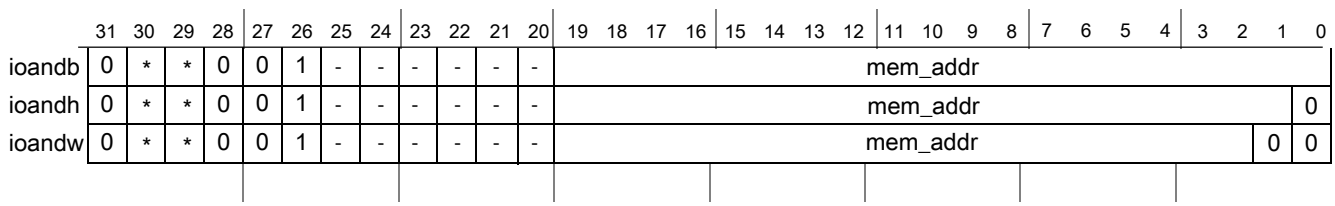


Figure 24-3. Decorated store address: logical AND

See Figure 24-3, where addr[30:29] = 10 for peripheral, addr[28:26] = 001 specifies the AND operation, and mem_addr[19:0] specifies the address offset into the space based at 0x4000_0000 for peripherals. The "-" indicates an address bit "don't care".

The decorated AND write operation is defined in the following pseudo-code as:

```
ioand<sz>(accessAddress, wdata)                  // decorated store AND
tmp   =  mem[accessAddress & 0xE00FFFFF, size]   // memory read
tmp   =  tmp & wdata                             // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp      // memory write
```

where the operand size <sz> is defined as b(yte, 8-bit), h(alfword, 16-bit) and w(ord, 32-bit). This notation is used throughout the document.

In the cycle definition tables, the notations AHB_ap and AHB_dp refer to the address and data phases of the BME AHB transaction. The cycle-by-cycle BME operations are detailed in the following table.

Table 24-1. Cycle definitions of decorated store: logical AND

Pipeline stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Convert	Recirculate captured addr + attr to memory as slave_wt	<next>

Table continues on the next page...

Table 24-1. Cycle definitions of decorated store: logical AND (continued)

Pipeline stage	Cycle		
	x	x+1	x+2
	master_wt to slave_rd; Capture address, attributes		
BME AHB_dp	<previous>	Perform memory read; Form (rdata & wdata) and capture destination data in register	Perform write sending registered data to memory

24.3.1.2 Decorated store logical OR (OR)

This command performs an atomic read-modify-write of the referenced memory location.

1. First, the location is read.
2. It is then modified by performing a logical OR operation using the write data operand sourced for the system bus cycle.
3. Finally, the result of the OR operation is written back into the referenced memory location.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit). The core performs the required write data lane replication on byte and halfword transfers.

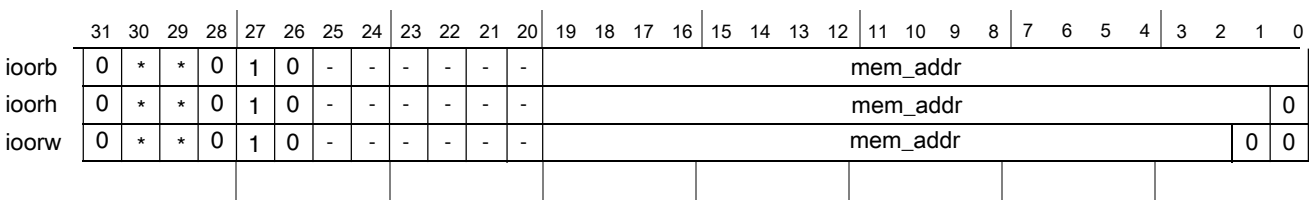


Figure 24-4. Decorated address store: logical OR

See Figure 24-4,where addr[30:29] =10 for peripheral, addr[28:26] = 010 specifies the OR operation, and mem_addr[19:0] specifies the address offset into the space based at 0x4000_0000 for peripherals. The "-" indicates an address bit "don't care".

The decorated OR write operation is defined in the following pseudo-code as:

```
ioor<sz>(accessAddress, wdata)                  // decorated store OR

tmp   =  mem[accessAddress & 0xE00FFFFF, size]  // memory read
tmp   =  tmp | wdata                            // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp     // memory write
```

The cycle-by-cycle BME operations are detailed in the following table.

Table 24-2. Cycle definitions of decorated store: logical OR

Pipeline stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Convert master_wt to slave_rd; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form (rdata I wdata) and capture destination data in register	Perform write sending registered data to memory

24.3.1.3 Decorated store logical XOR (XOR)

This command performs an atomic read-modify-write of the referenced memory location.

1. First, the location is read.
2. It is then modified by performing a logical XOR (exclusive-OR) operation using the write data operand sourced for the system bus cycle.
3. Finally, the result of the XOR operation is written back into the referenced memory location.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit). The core performs the required write data lane replication on byte and halfword transfers.

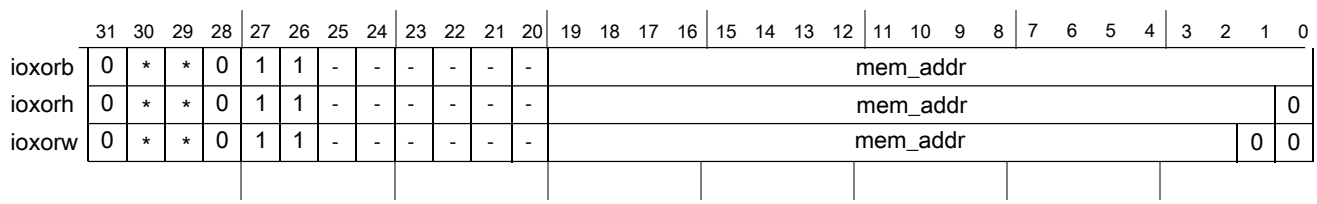


Figure 24-5. Decorated address store: logical XOR

See Figure 24-5, where addr[30:29] =10 for peripheral, addr[28:26] = 011 specifies the XOR operation, and mem_addr[19:0] specifies the address offset into the peripheral space based at 0x4000_0000 for peripherals. The "-" indicates an address bit "don't care".

The decorated XOR write operation is defined in the following pseudo-code as:

```
ioxor<sz>(accessAddress, wdata)                 // decorated store XOR

tmp   =  mem[accessAddress & 0xE00FFFFF, size]  // memory read
tmp   =  tmp ^ wdata                            // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp     // memory write
```

The cycle-by-cycle BME operations are detailed in the following table.

Table 24-3. Cycle definitions of decorated store: logical XOR

Pipeline Stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Convert master_wt to slave_rd; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form (rdata ^ wdata) and capture destination data in register	Perform write sending registered data to memory

24.3.1.4 Decorated store bit field insert (BFI)

This command inserts a bit field contained in the write data operand, defined by LSB position (b) and the bit field width (w+1), into the memory "container" defined by the access size associated with the store instruction using an atomic read-modify-write sequence.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit).

NOTE

For the word sized operation, the maximum bit field width is 16 bits. The core performs the required write data lane replication on byte and halfword transfers.

The BFI operation can be used to insert a single bit into a peripheral. For this case, the w field is simply set to 0, indicating a bit field width of 1.

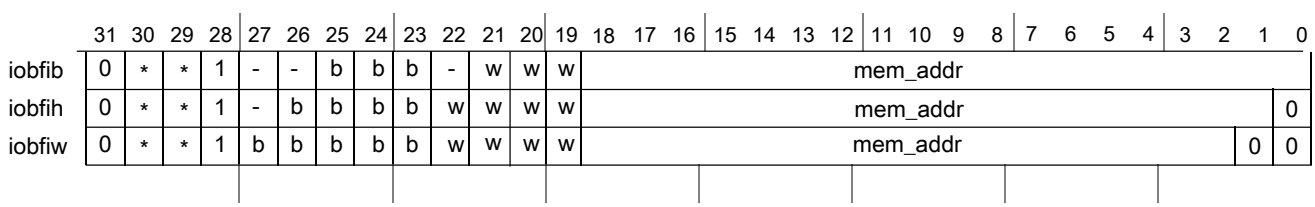


Figure 24-6. Decorated address store: bit field insert

where addr[30:29] =10 for peripheral,addr[28] = 1 signals a BFI operation, addr[27:23] is "b", the LSB identifier, addr[22:19] is "w", the bit field width minus 1 identifier, and addr[18:0] specifies the address offset into the peripheral space based at 0x4000_0000 for peripherals. The "-" indicates an address bit "don't care". Note, unlike the other decorated store operations, BFI uses addr[19] as the least significant bit in the "w" specifier and not as an address bit.

The decorated BFI write operation is defined in the following pseudo-code as:

```
iobfi<sz>(accessAddress, wdata)                  // decorated bit field insert

tmp   =  mem[accessAddress & 0xE007FFFF, size]  // memory read
mask  = ((1 << (w+1)) - 1) << b                 // generate bit mask
tmp   =  tmp   & ~mask                          // modify
      |  wdata &  mask
mem[accessAddress & 0xE007FFFF, size] = tmp     // memory write
```

The write data operand (wdata) associated with the store instruction contains the bit field to be inserted. It must be properly aligned within a right-aligned container, that is, within the lower 8 bits for a byte operation, the lower 16 bits for a halfword, or the entire 32 bits for a word operation.

To illustrate, consider the following example of the insertion of the 3-bit field "xyz" into an 8-bit memory container, initially set to "abcd_efgh". For all cases, w is 2, signaling a bit field width of 3.

```
if b = 0 and the decorated store (strb) Rt register[7:0] = ----_-xyz,
   then destination is "abcd_exyz"
if b = 1 and the decorated store (strb) Rt register[7:0] = ----_xyz-,
   then destination is "abcd_xyzh"
if b = 2 and the decorated store (strb) Rt register[7:0] = ---x_yz--,
   then destination is "abcx_yzgh"
if b = 3 and the decorated store (strb) Rt register[7:0] = --xy_z---,
   then destination is "abxy_zfgh"
if b = 4 and the decorated store (strb) Rt register[7:0] = -xyz_----,
   then destination is "axyz_efgh"
if b = 5 and the decorated store (strb) Rt register[7:0] = xyz-_----,
   then destination is "xyzd_efgh"
if b = 6 and the decorated store (strb) Rt register[7:0] = yz--_----,
   then destination is "yzcd_efgh"
if b = 7 and the decorated store (strb) Rt register[7:0] = z---_----,
   then destination is "zbcd_efgh"
```

Note from the example, when the starting bit position plus the field width exceeds the container size, only part of the source bit field is inserted into the destination memory location. Stated differently, if (b + w+1) > container_width, only the low-order "container_width - b" bits are actually inserted.

The cycle-by-cycle BME operations are detailed in the following table.

Table 24-4. Cycle definitions of decorated store: bit field insert

Pipeline stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Convert master_wt to slave_rd; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form bit mask; Form bitwise ((mask) ? wdata : rdata)) and capture destination data in register	Perform write sending registered data to memory

24.3.2 BME decorated loads

The functions supported by the BME's decorated loads include two single-bit load-and-{set, clear} operators plus unsigned bit field extracts.

For the two load-and-{set, clear} operations, BME converts a single decorated AHB load transaction into a two-cycle atomic read-modify-write sequence, where the combined read-modify operations are performed in the first AHB data phase, and then the write is performed in the second AHB data phase as the original read data is returned to the processor core. For an unsigned bit field extract, the decorated load transaction is stalled for one cycle in the BME as the data field is extracted, then aligned and returned to the processor in the second AHB data phase. This is the only decorated transaction that is not an atomic read-modify-write, as it is a simple data read.

A generic timing diagram of a decorated load showing a peripheral load-and-set 1-bit operation is shown as follows.

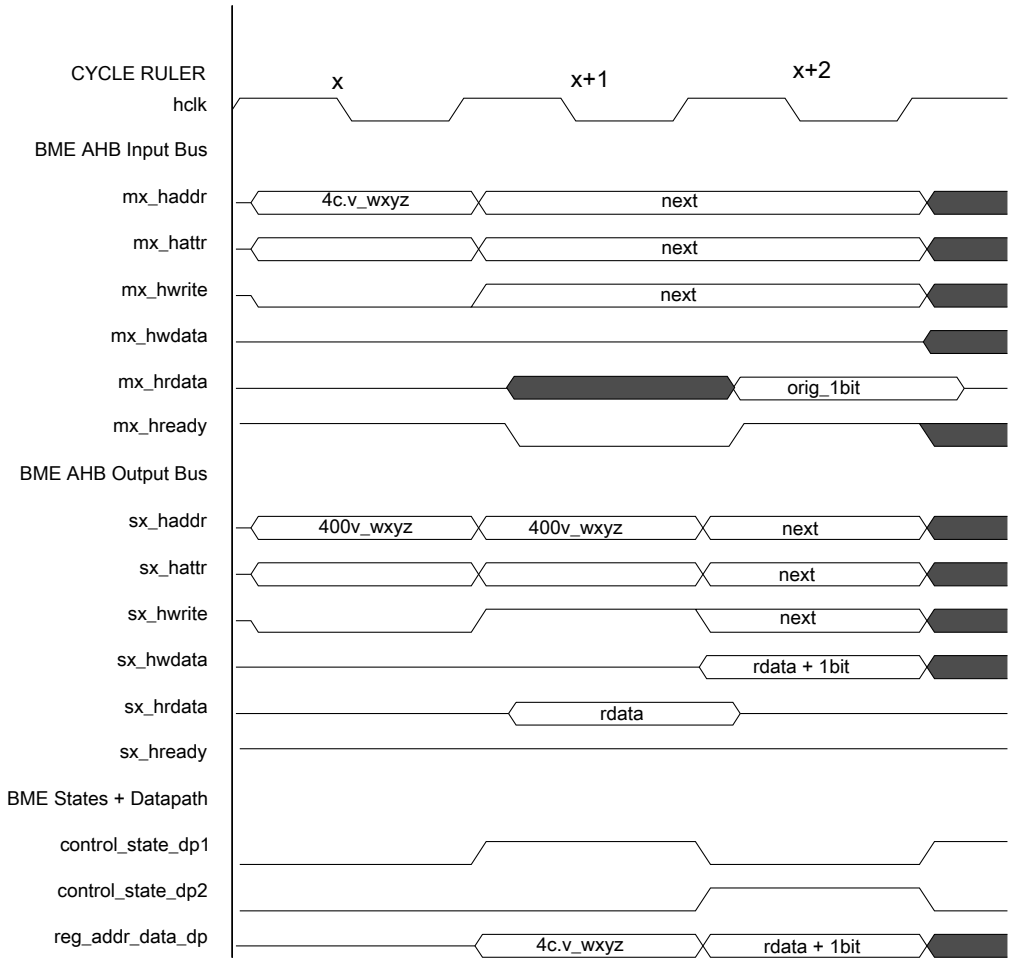


Figure 24-7. Decorated load: load-and-set 1-bit field insert timing diagram

Decorated load-and-{set, clear} 1-bit operations follow the execution template shown in the above figure: a 2-cycle read-modify-write operation:

1. Cycle x, first AHB address phase: Read from input bus is translated into a read operation on the output bus with the actual memory address (with the decoration removed) and then captured in a register
2. Cycle x+1, second AHB address phase: Write access with the registered (but actual) memory address is output
3. Cycle x+1, first AHB data phase: The "original" 1-bit memory read data is captured in a register, while the 1-bit field is set or clear based on the function defined by the decoration with the modified data captured in a register; the input bus cycle is stalled
4. Cycle x+2, second AHB data phase: The selected original 1-bit is right-justified, zero-filled and then driven onto the input read data bus, while the registered write data is sourced onto the output write data bus

NOTE

Any wait states inserted by the slave device are simply passed through the BME back to the master input bus, stalling the AHB transaction cycle for cycle.

A generic timing diagram of a decorated load showing an unsigned peripheral bit field operation is shown in the following figure.

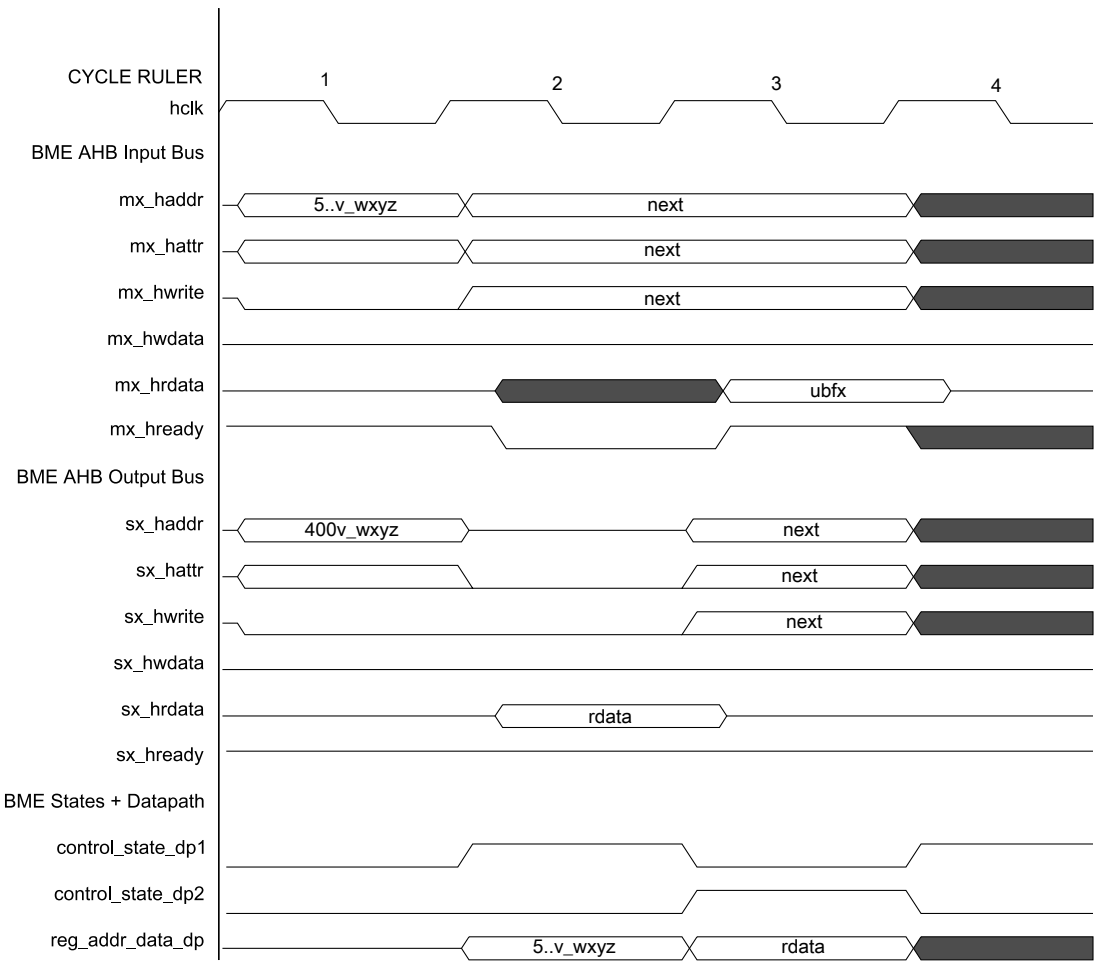


Figure 24-8. Decorated load: unsigned bit field insert timing diagram

The decorated unsigned bit field extract follows the same execution template shown in the above figure, a 2-cycle read operation:

- Cycle x, 1st AHB address phase: Read from input bus is translated into a read operation on the output bus with the actual memory address (with the decoration removed) and then captured in a register
- Cycle x+1, 2nd AHB address phase: Idle cycle

- Cycle x+1, 1st AHB data phase: A bit mask is generated based on the starting bit position and the field width; the mask is AND'ed with the memory read data to isolate the bit field; the resulting data is captured in a data register; the input bus cycle is stalled
- Cycle x+2, 2nd AHB data phase: Registered data is logically right-aligned for proper alignment and driven onto the input read data bus

NOTE

Any wait states inserted by the slave device are simply passed through the BME back to the master input bus, stalling the AHB transaction cycle for cycle.

24.3.2.1 Decorated load: load-and-clear 1 bit (LAC1)

This command loads a 1-bit field defined by the LSB position (b) into the core's general purpose destination register (Rt) and zeroes the bit in the memory space after performing an atomic read-modify-write sequence.

The extracted 1-bit data field from the memory address is right-justified and zero-filled in the operand returned to the core.

The data size is specified by the read operation and can be byte (8-bit), halfword (16-bit) or word (32-bit).

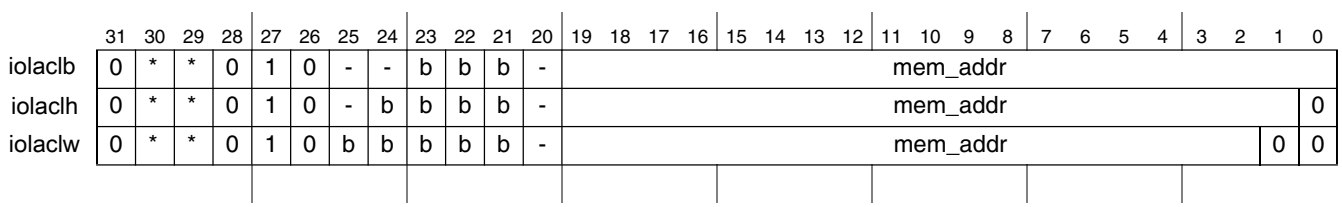


Figure 24-9. Decorated load address: load-and-clear 1 bit

See Figure 24-9, where addr[30:29] = 10 for peripheral, addr[28:26] = 010 specifies the load-and-clear 1 bit operation, addr[25:21] is "b", the bit identifier, and mem_addr[19:0] specifies the address offset into the space based at 0x4000_0000 for peripheral. The "-" indicates an address bit "don't care".

The decorated load-and-clear 1-bit read operation is defined in the following pseudo-code as:

```
rdata =  iolac1<sz>(accessAddress)              // decorated load-and-clear 1

tmp   =  mem[accessAddress & 0xE00FFFFF, size]  // memory read
mask  =  1 << b                                 // generate bit mask
rdata = (tmp &  mask) >> b                      // read data returned to core
tmp   =  tmp & ~mask                            // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp     // memory write
```

The cycle-by-cycle BME operations are detailed in the following table.

Table 24-5. Cycle definitions of decorated load: load-and-clear 1 bit

Pipeline Stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form bit mask; Extract bit from rdata; Form (rdata & ~mask) and capture destination data in register	Return extracted bit to master; Perform write sending registered data to memory

24.3.2.2 Decorated Load: Load-and-Set 1 Bit (LAS1)

This command loads a 1-bit field defined by the LSB position (b) into the core's general purpose destination register (Rt) and sets the bit in the memory space after performing an atomic read-modify-write sequence.

The extracted one bit data field from the memory address is right justified and zero filled in the operand returned to the core.

The data size is specified by the read operation and can be byte (8-bit), halfword (16-bit) or word (32-bit).

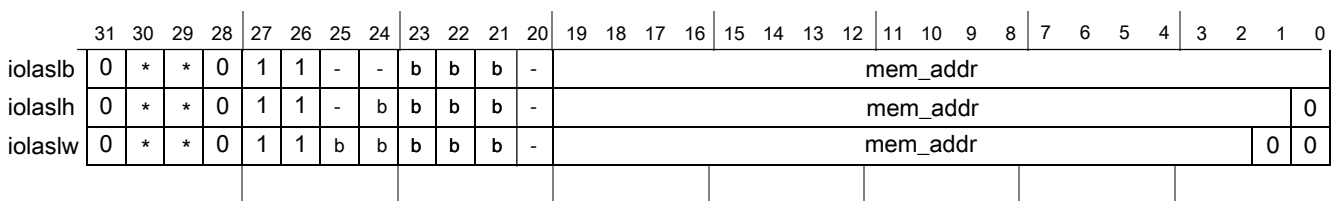


Figure 24-10. Decorated load address: load-and-set 1 bit

where addr[30:29] = 10 for peripheral, addr[28:26] = 011 specifies the load-and-set 1 bit operation, addr[25:21] is "b", the bit identifier, and mem_addr[19:0] specifies the address offset into the space based at 0x4000_0000 for peripheral. The "-" indicates an address bit "don't care".

The decorated Load-and-Set 1 Bit read operation is defined in the following pseudo-code as:

```
rdata =  iolas1<sz>(accessAddress)              // decorated load-and-set 1

tmp   =  mem[accessAddress & 0xE00FFFFF, size]  // memory read
mask  =  1 << b                                 // generate bit mask
rdata = (tmp &  mask) >> b                      // read data returned to core
```

```
tmp    =  tmp |  mask                              // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp       // memory write
```

The cycle-by-cycle BME operations are detailed in the following table.

Table 24-6. Cycle definitions of decorated load: load-and-set 1-bit

Pipeline Stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form bit mask; Extract bit from rdata; Form (rdata | mask) and capture destination data in register	Return extracted bit to master; Perform write sending registered data to memory

24.3.2.3 Decorated load unsigned bit field extract (UBFX)

This command extracts a bit field defined by LSB position (b) and the bit field width (w +1) from the memory "container" defined by the access size associated with the load instruction using a two-cycle read sequence.

The extracted bit field from the memory address is right-justified and zero-filled in the operand returned to the core. Recall this is the only decorated operation that does not perform a memory write, that is, UBFX only performs a read.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit). Note for the word sized operation, the maximum bit field width is 16 bits.

The use of a UBFX operation is recommended to extract a single bit. For this case, the w field is simply set to 0, indicating a bit field width of 1.

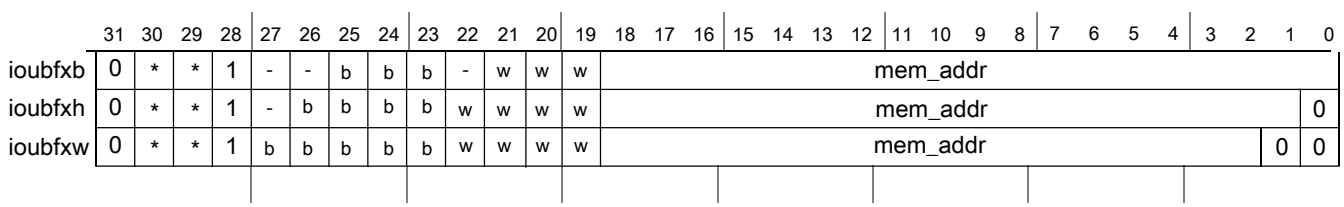


Figure 24-11. Decorated load address: unsigned bit field extract

See Figure 24-11, where addr[30:29] = 10 for peripheral, addr[28] = 1 specifies the unsigned bit field extract operation, addr[27:23] is "b", the LSB identifier, addr[22:19] is "w", the bit field width minus 1 identifier, and mem_addr[18:0] specifies the address

offset into the space based at 0x4000_0000 for peripheral. The "-" indicates an address bit "don't care". Note, unlike the other decorated load operations, UBFX uses addr[19] as the least significant bit in the "w" specifier and not as an address bit.

The decorated unsigned bit field extract read operation is defined in the following pseudo-code as:

```
rdata =  ioubfx<sz>(accessAddress)              // unsigned bit field extract

tmp   =  mem[accessAddress & 0xE007FFFF, size]  // memory read
mask  = ((1 << (w+1)) - 1) << b                 // generate bit mask
rdata = (tmp &  mask) >> b                      // read data returned to core
```

Like the BFI operation, when the starting bit position plus the field width exceeds the container size, only part of the source bit field is extracted from the destination memory location. Stated differently, if (b + w+1) > container_width, only the low-order "container_width - b" bits are actually extracted. The cycle-by-cycle BME operations are detailed in the following table.

Table 24-7. Cycle definitions of decorated load: unsigned bit field extract

Pipeline Stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Capture address, attributes	Idle AHB address phase	<next>
BME AHB_dp	<previous>	Perform memory read; Form bit mask; Form (rdata & mask) and capture destination data in register	Logically right shift registered data; Return justified rdata to master

24.3.3 Additional details on decorated addresses and GPIO accesses

As previously noted, the peripheral address space occupies a 516 KB region: 512 KB based at 0x4000_0000 plus a 4 KB space based at 0x400F_F000 for GPIO accesses. This memory layout provides compatibility with the Kinetis K Family and provides 129 address "slots", each 4 KB in size.

The GPIO address space is multiply-mapped by the hardware: it appears at the "standard" system address 0x400F_F000 and is physically located in the address slot corresponding to address 0x4000_F000. Decorated loads and stores create a slight complication involving accesses to the GPIO. Recall the use of address[19] varies by decorated operation; for AND, OR, XOR, LAC1 and LAS1, this bit functions as a true address bit, while for BFI and UBFX, this bit defines the least significant bit of the "w" bit field specifier.

As a result, undecorated GPIO references and decorated AND, OR, XOR, LAC1 and LAS1 operations can use the standard 0x400F_F000 base address, while decorated BFI and UBFX operations must use the alternate 0x4000_F000 base address. Another implementation can simply use 0x400F_F000 as the base address for all undecorated GPIO accesses and 0x4000_F000 as the base address for all decorated accesses. Both implementations are supported by the hardware.

Table 24-8. Decorated peripheral and GPIO address details

Peripheral address space	Description
0x4000_0000–0x4007_FFFF	Undecorated (normal) peripheral accesses
0x4008_0000–0x400F_EFFF	Illegal addresses; attempted references are aborted and error terminated
0x400F_F000–0x400F_FFFF	Undecorated (normal) GPIO accesses using standard address
0x4010_0000–0x43FF_FFFF	Illegal addresses; attempted references are aborted and error terminated
0x4400_0000–0x4FFF_FFFF	Decorated AND, OR, XOR, LAC1, LAS1 references to peripherals and GPIO based at either 0x4000_F000 or 0x400F_F000
0x5000_0000–0x5FFF_FFFF	Decorated BFI, UBFX references to peripherals and GPIO only based at 0x4000_F000

24.4 Application information

In this section, GNU assembler macros with C expression operands are presented as examples of the required instructions to perform decorated operations.

This section specifically presents a partial bme.h file defining the assembly language expressions for decorated logical stores: AND, OR, and XOR. Comparable functions for BFI and the decorated loads are more complex and available in the complete BME header file.

These macros use the same function names presented in Functional description.

```
#define IOANDW(ADDR,WDATA)          \
    __asm("ldr    r3, =(1<<26);"    \
          "orr    r3, %[addr];"     \
          "mov    r2, %[wdata];"    \
          "str    r2, [r3];"        \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOANDH(ADDR,WDATA)          \
    __asm("ldr    r3, =(1<<26);"    \
          "orr    r3, %[addr];"     \
          "mov    r2, %[wdata];"    \
          "strh   r2, [r3];"        \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOANDB(ADDR,WDATA)          \
    __asm("ldr    r3, =(1<<26);"    \
          "orr    r3, %[addr];"     \
          "mov    r2, %[wdata];"    \
          "strb   r2, [r3];"        \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");
```

```
#define IOORW(ADDR,WDATA)           \
    __asm("ldr    r3, =(1<<27);"    \
          "orr    r3, %[addr];"     \
          "mov    r2, %[wdata];"    \
          "str    r2, [r3];"        \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOORH(ADDR,WDATA)           \
    __asm("ldr    r3, =(1<<27);"    \
          "orr    r3, %[addr];"     \
          "mov    r2, %[wdata];"    \
          "strh   r2, [r3];"        \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOORB(ADDR,WDATA)           \
    __asm("ldr    r3, =(1<<27);"    \
          "orr    r3, %[addr];"     \
          "mov    r2, %[wdata];"    \
          "strb   r2, [r3];"        \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOXORW(ADDR,WDATA)          \
    __asm("ldr    r3, =(3<<26);"    \
          "orr    r3, %[addr];"     \
          "mov    r2, %[wdata];"    \
          "str    r2, [r3];"        \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOXORH(ADDR,WDATA)          \
    __asm("ldr    r3, =(3<<26);"    \
          "orr    r3, %[addr];"     \
          "mov    r2, %[wdata];"    \
          "strh   r2, [r3];"        \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOXORB(ADDR,WDATA)          \
    __asm("ldr    r3, =(3<<26);"    \
          "orr    r3, %[addr];"     \
          "mov    r2, %[wdata];"    \
          "strb   r2, [r3];"        \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");
```

Chapter 25
Direct Memory Access Multiplexer (DMAMUX)

25.1 Chip-specific DMAMUX information

25.1.1 DMAMUX request sources

DMAMUX0 is a DMA request mux that allows up to 63 DMA request signals to be mapped to any of the 8 DMA channels of DMA0. Because of the mux there is no hard correlation between any of the DMA request sources and a specific DMA channel. Some of the modules support asynchronous DMA operation as indicated in the following DMA source assignment table.

Table 25-1. DMAMUX0 source assignments

Source No	Source module	Source description	Async DMA capable
0	—	Channel disabled	—
1	FlexIO0	Shifter 0	Yes
2	FlexIO0	Shifter 1	Yes
3	FlexIO0	Shifter 2	Yes
4	FlexIO0	Shifter 3	Yes
5	FlexIO0	Shifter 4	Yes
6	FlexIO0	Shifter 5	Yes
7	FlexIO0	Shifter 6	Yes
8	FlexIO0	Shifter 7	Yes
9	I2C0	Transmit/Receive	No
10	I2C1	Transmit/Receive	No
11	—	—	—
12	—	—	—
13	—	—	—
14	—	—	—
15	LPUART0	Receive	Yes

Table continues on the next page...

Table 25-1. DMAMUX0 source assignments (continued)

Source No	Source module	Source description	Async DMA capable
16	LPUART0	Transmit	Yes
17	LPUART1	Receive	Yes
18	LPUART1	Transmit	Yes
19	LPUART2	Receive	Yes
20	LPUART2	Transmit	Yes
21	SPI0	Receive	No
22	SPI0	Transmit	No
23	SPI1	Receive	No
24	SPI1	Transmit	No
25	QSPI0	Receive	No
26	QSPI0	Transmit	No
27	TPM0	Channel 0	Yes
28	TPM0	Channel 1	Yes
29	TPM0	Channel 2	Yes
30	TPM0	Channel 3	Yes
31	TPM0	Channel 4	Yes
32	TPM0	Channel 5	Yes
33	—	—	—
34	—	—	—
35	TPM0	Overflow	Yes
36	TPM1	Channel 0	Yes
37	TPM1	Channel 1	Yes
38	TPM1	Overflow	Yes
39	TPM2	Channel 0	Yes
40	TPM2	Channel 1	Yes
41	TPM2	Overflow	Yes
42	TSI0	End of scan	Yes
43	EMVSIM0	Receive	No
44	EMVSIM0	Transmit	No
45	EMVSIM1	Receive	No
46	EMVSIM1	Transmit	No
47	PortA	Port Control module	Yes
48	PortB	Port Control module	Yes
49	PortC	Port Control module	Yes
50	PortD	Port Control Module	Yes
51	PortE	Port Control module	Yes
52	ADC0	conversion complete	Yes
53	—	—	—
54	DAC0	DAC0 buffer	No

Table continues on the next page...

Table 25-1. DMAMUX0 source assignments (continued)

Source No	Source module	Source description	Async DMA capable
55	LTC0	PKHA RAM read	No
56	CMP0	Comparison event	Yes
57	—	—	—
58	LTC0	Input FIFO	No
59	LTC0	Output FIFO	No
60	DMAMUX	Always enabled	No
61	DMAMUX	Always enabled	No
62	DMAMUX	Always enabled	No
63	DMAMUX	Always enabled	No

25.1.2 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 .

25.2 Introduction

25.2.1 Overview

The Direct Memory Access Multiplexer (DMAMUX) routes DMA sources, called slots, to any of the 8 DMA channels. This process is illustrated in the following figure.

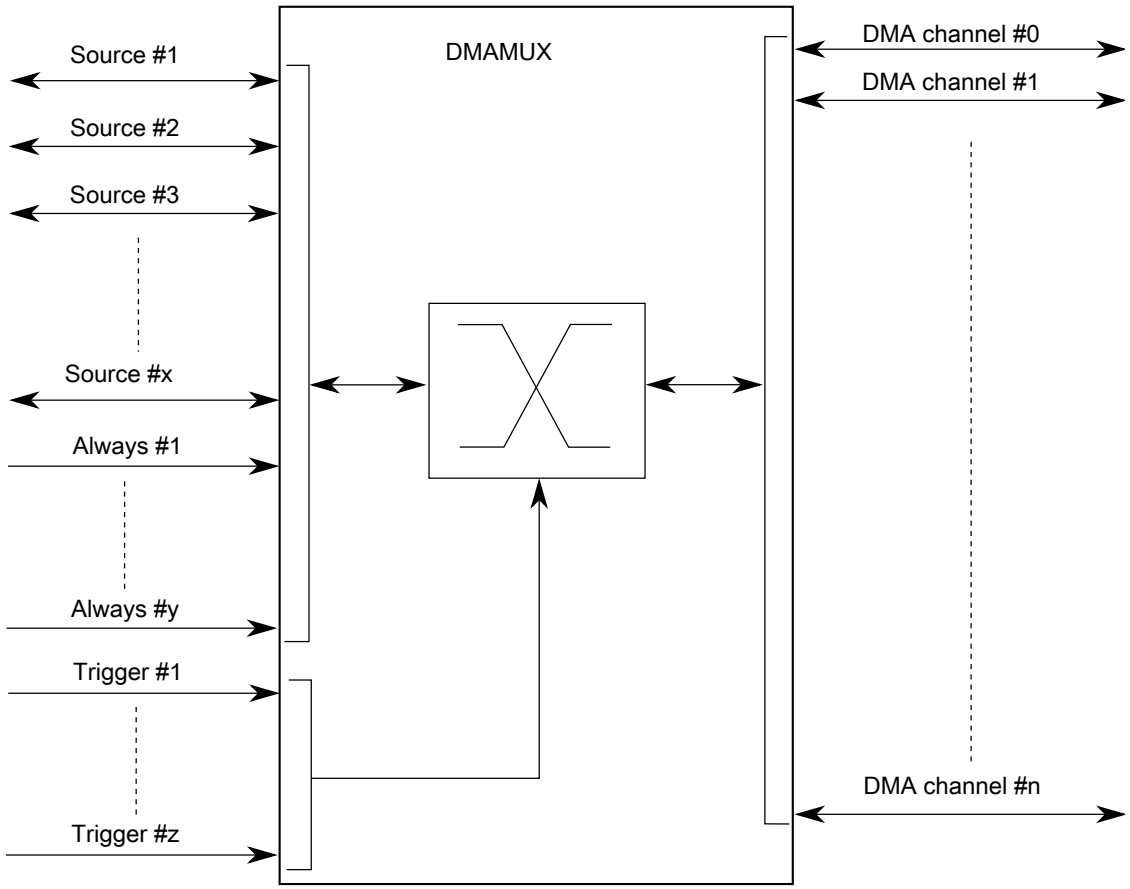


Figure 25-1. DMAMUX block diagram

25.2.2 Features

The DMAMUX module provides these features:

- Up to 59 peripheral slots and up to four always-on slots can be routed to 8 channels.
- 8 independently selectable DMA channel routers.
  - The first four channels additionally provide a trigger functionality.
- Each channel router can be assigned to one of the possible peripheral DMA slots or to one of the always-on slots.

25.2.3 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.

25.3 External signal description

The DMAMUX has no external pins.

25.4 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	25.4.1/498
4002_1001	Channel Configuration register (DMAMUX_CHCFG1)	8	R/W	00h	25.4.1/498
4002_1002	Channel Configuration register (DMAMUX_CHCFG2)	8	R/W	00h	25.4.1/498
4002_1003	Channel Configuration register (DMAMUX_CHCFG3)	8	R/W	00h	25.4.1/498
4002_1004	Channel Configuration register (DMAMUX_CHCFG4)	8	R/W	00h	25.4.1/498
4002_1005	Channel Configuration register (DMAMUX_CHCFG5)	8	R/W	00h	25.4.1/498
4002_1006	Channel Configuration register (DMAMUX_CHCFG6)	8	R/W	00h	25.4.1/498
4002_1007	Channel Configuration register (DMAMUX_CHCFG7)	8	R/W	00h	25.4.1/498

25.4.1 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. This is true, even if a channel is disabled (ENBL==0).

Before changing the trigger or source settings, a DMA channel must be disabled via CHCFGn[ENBL].

Address: 4002_1000h base + 0h offset + (1d × i), where i=0d to 7d

Bit	7	6	5	4	3	2	1	0
Read Write	ENBL	TRIG	SOURCE					
Reset	0	0	0	0	0	0	0	0

DMAMUX_CHCFG*n* field descriptions

Field	Description
7 ENBL	DMA Channel Enable Enables the DMA channel. 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. 1 DMA channel is enabled
6 TRIG	DMA Channel Trigger Enable Enables the periodic trigger capability for the triggered DMA channel. 0 Triggering is disabled. If triggering is disabled and ENBL is set, the DMA Channel will simply route the specified source to the DMA channel. (Normal mode) 1 Triggering is enabled. If triggering is enabled and ENBL is set, the DMAMUX is in Periodic Trigger mode.
SOURCE	DMA Channel Source (Slot) Specifies which DMA source, if any, is routed to a particular DMA channel. See the chip-specific DMAMUX information for details about the peripherals and their slot numbers.

25.5 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

25.5.1 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.

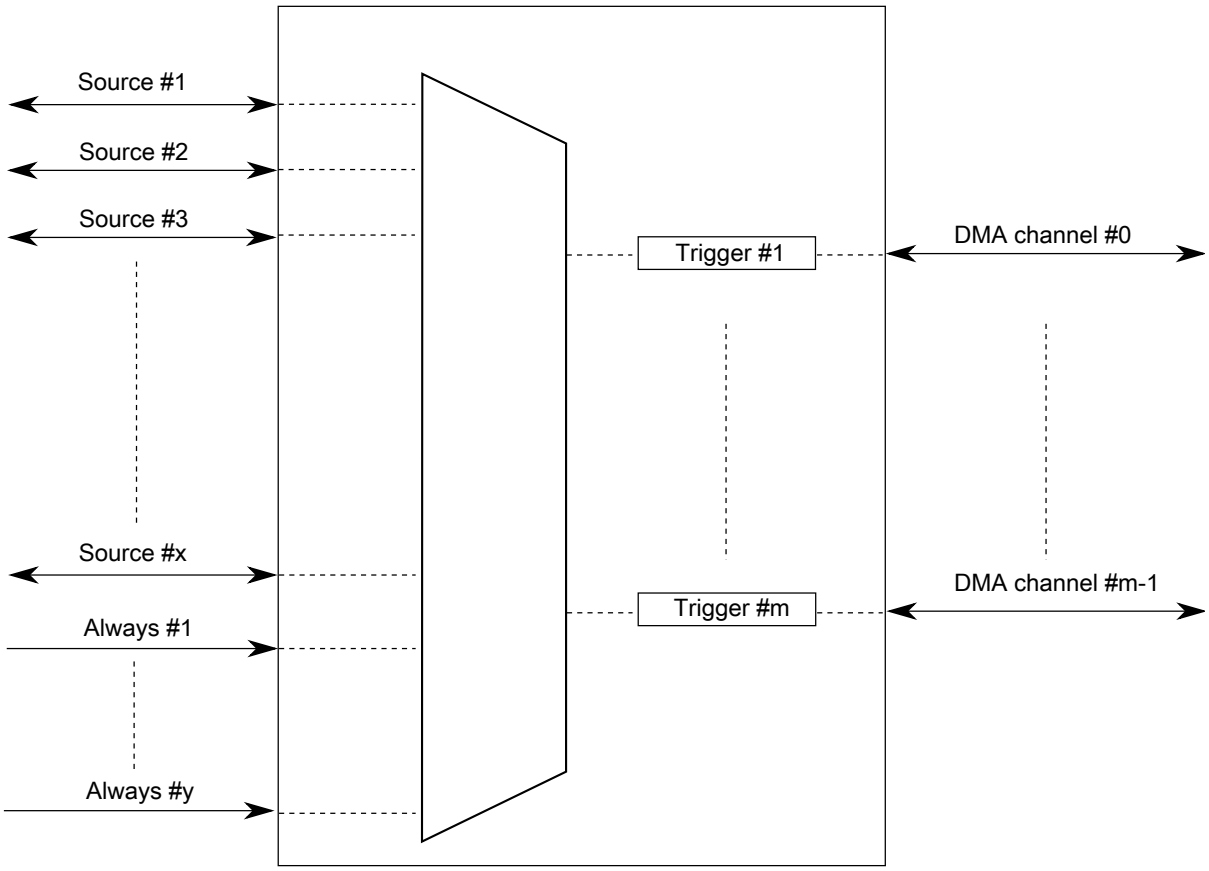


Figure 25-2. DMAMUX triggered channels

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.

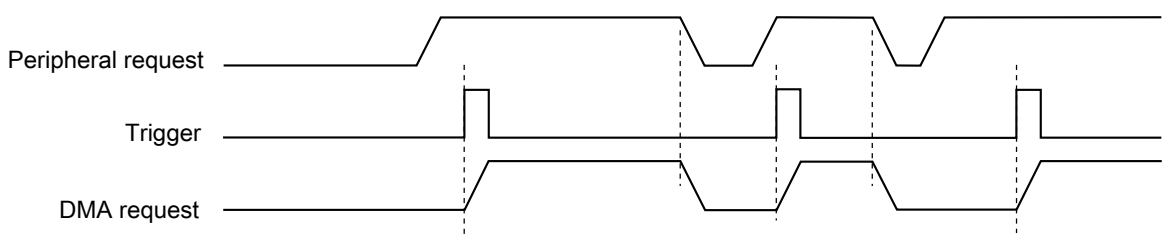


Figure 25-3. DMAMUX channel triggering: normal operation

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.

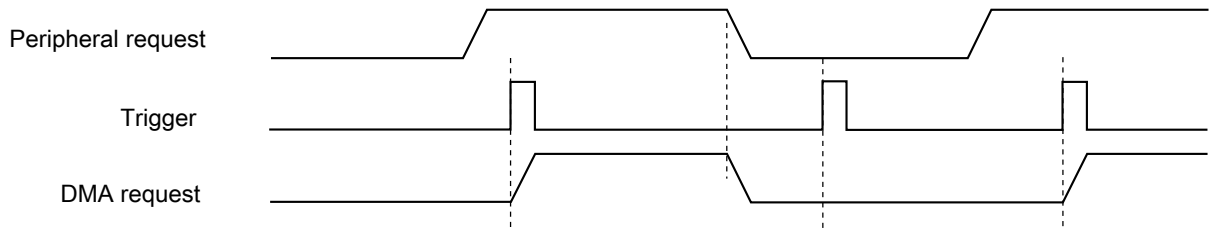


Figure 25-4. DMAMUX channel triggering: ignored trigger

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.

25.5.2 DMA channels with no triggering capability

The other channels of the DMAMUX provide the normal routing functionality as described in Modes of operation.

25.5.3 Always-enabled DMA sources

In addition to the peripherals that can be used as DMA sources, there are four 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.

25.6 Initialization/application information

This section provides instructions for initializing the DMA channel MUX.

25.6.1 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.

25.6.2 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.

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.

The following code example illustrates steps 1 and 4 above:

```
void DMAMUX_Init(uint8_t DMA_CH, uint8_t DMAMUX_SOURCE)
{
    DMAMUX_0.CHCFG[DMA_CH].B.SOURCE = DMAMUX_SOURCE;
    DMAMUX_0.CHCFG[DMA_CH].B.ENBL   = 1;
    DMAMUX_0.CHCFG[DMA_CH].B.TRIG   = 1;
}
```

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.
2. Configure channel 1 in the DMA, including enabling the channel.
3. Write 0x85 to CHCFG1.

The following code example illustrates steps 1 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
```

```
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);

In File main.c:
#include "registers.h"
:
:
*CHCFG1 = 0x00;
*CHCFG1 = 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.
3. Write 0x87 to CHCFG8. (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     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
```

```
In File main.c:
#include "registers.h"
:
:
*CHCFG8 = 0x00;
*CHCFG8 = 0x87;
```

Chapter 26
Enhanced Direct Memory Access (eDMA)

26.1 Introduction

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 address and destination address calculations
  - Data-movement operations
- Local memory containing transfer control descriptors for each of the 8 channels

26.1.1 eDMA system block diagram

Figure 26-1 illustrates the components of the eDMA system, including the eDMA module ("engine").

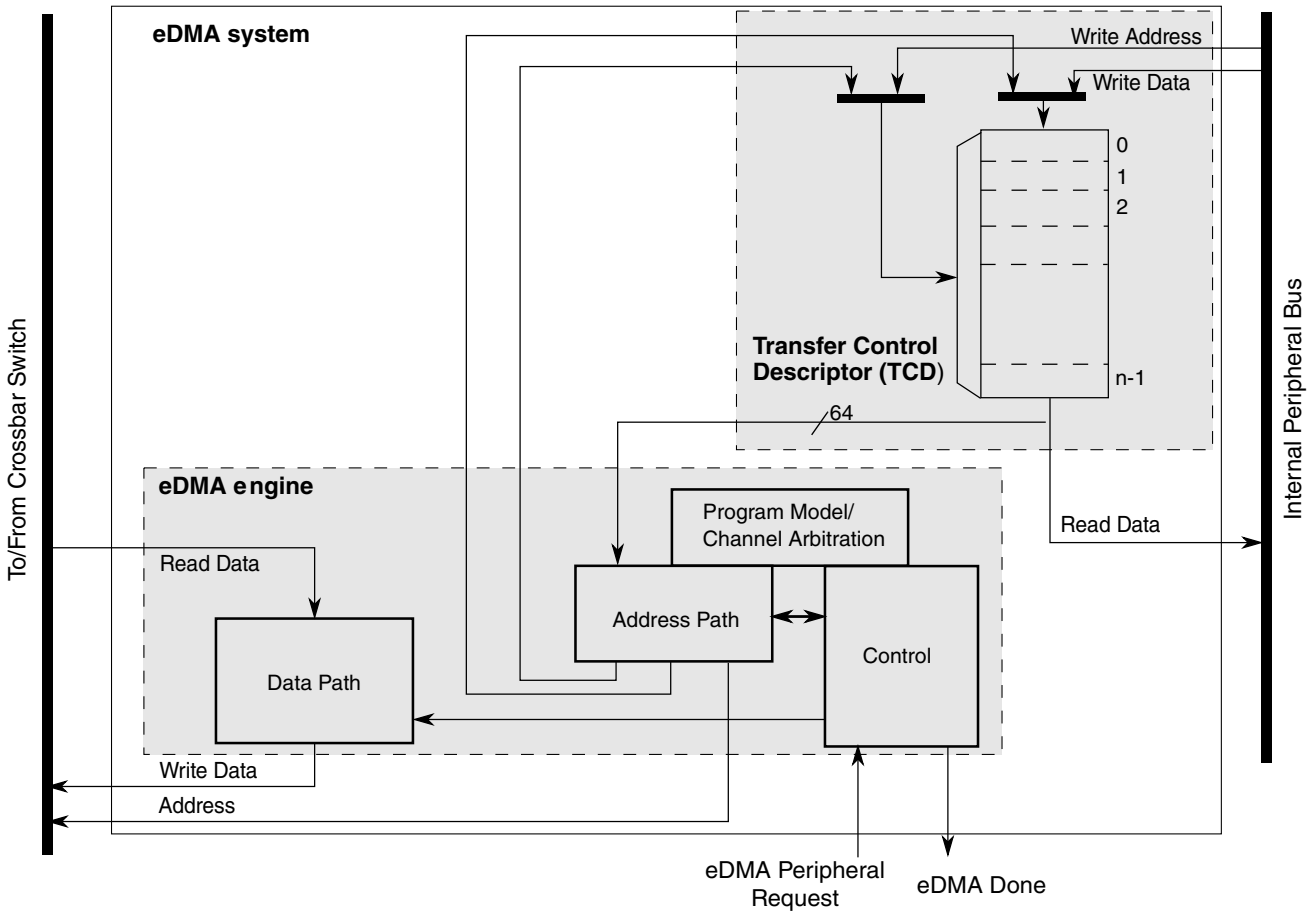


Figure 26-1. eDMA system block diagram

26.1.2 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 26-1. 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. 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

Table continues on the next page...

Table 26-1. eDMA engine submodules (continued)

Submodule	Function
	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 a data buffer and the necessary multiplex logic to support any required data alignment. The internal read data bus is the primary input, and the internal write data bus is the primary output. 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 26-2. 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.

26.1.3 Features

The eDMA is a highly programmable data-transfer engine optimized to minimize any required intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known and not defined within the transferred data itself. The eDMA module features:

- All data movement via dual-address transfers: read from source, write to destination
  - Programmable source and destination addresses and transfer size
  - Support for enhanced addressing modes

- 8-channel implementation that performs complex data transfers with minimal intervention from a host processor
  - 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 programmable interrupt requests
  - One interrupt per channel, which can be asserted at completion of major iteration count
  - Programmable error terminations per channel and logically summed together to form one error interrupt to the interrupt controller
- Programmable support for scatter/gather DMA processing
- Support for complex data structures

In the discussion of this module, *n* is used to reference the channel number.

26.2 Modes of operation

The eDMA operates in the following modes:

Table 26-3. 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.

Table continues on the next page...

Table 26-3. Modes of operation (continued)

Mode	Description
	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: • If CR[EDBG] is cleared, the DMA continues to operate. • 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.

26.3 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

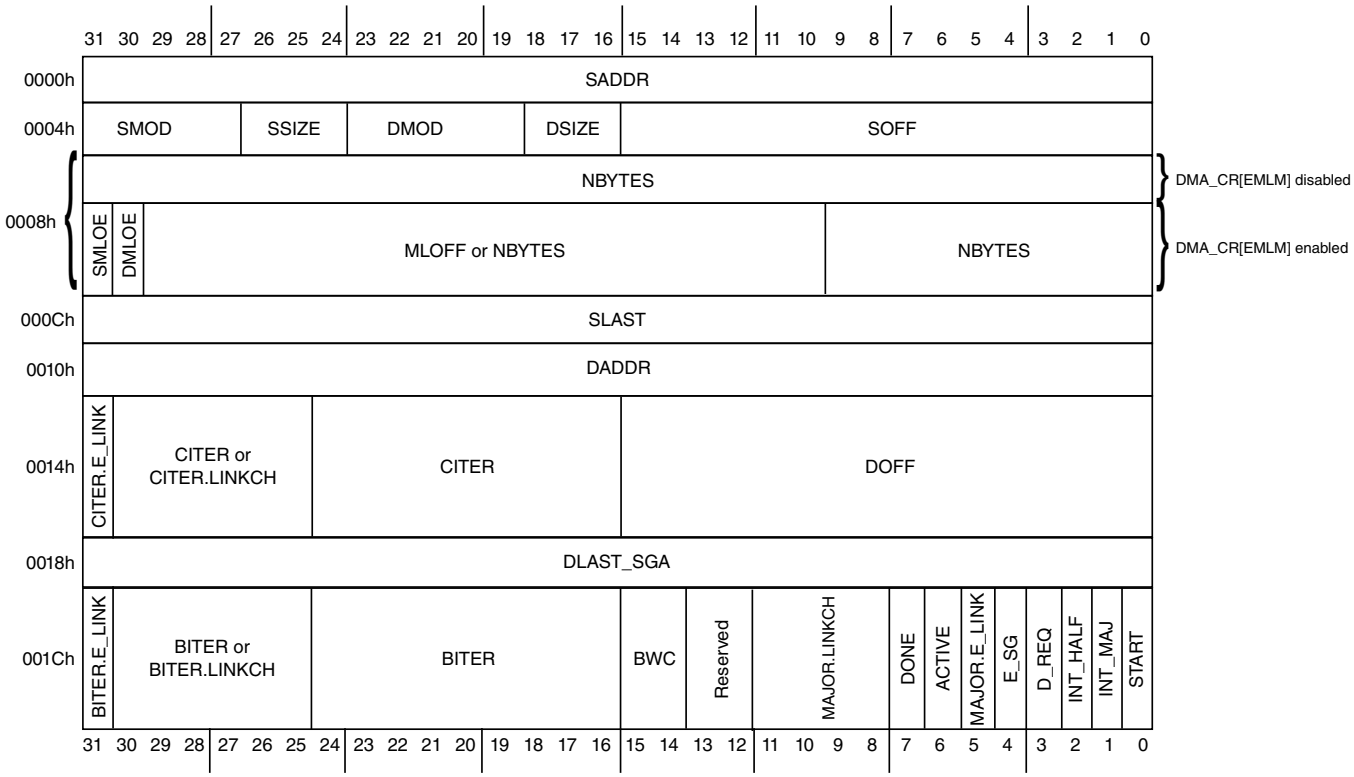
26.3.1 TCD memory

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 7. Each TCD*n* definition is presented as 11 registers of 16 or 32 bits.

26.3.2 TCD initialization

Prior to activating a channel, you must initialize its TCD with the appropriate transfer profile.

26.3.3 TCD structure



26.3.4 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	See section	26.3.1/519
4000_8004	Error Status Register (DMA_ES)	32	R	0000_0000h	26.3.2/522
4000_800C	Enable Request Register (DMA_ERQ)	32	R/W	0000_0000h	26.3.3/524
4000_8014	Enable Error Interrupt Register (DMA_EEI)	32	R/W	0000_0000h	26.3.4/526
4000_8018	Clear Enable Error Interrupt Register (DMA_CEEI)	8	W (always reads 0)	00h	26.3.5/527
4000_8019	Set Enable Error Interrupt Register (DMA_SEEI)	8	W (always reads 0)	00h	26.3.6/528

Table continues on the next page...

DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4000_801A	Clear Enable Request Register (DMA_CERQ)	8	W (always reads 0)	00h	26.3.7/529
4000_801B	Set Enable Request Register (DMA_SERQ)	8	W (always reads 0)	00h	26.3.8/530
4000_801C	Clear DONE Status Bit Register (DMA_CDNE)	8	W (always reads 0)	00h	26.3.9/531
4000_801D	Set START Bit Register (DMA_SSRT)	8	W (always reads 0)	00h	26.3.10/ 532
4000_801E	Clear Error Register (DMA_CERR)	8	W (always reads 0)	00h	26.3.11/ 533
4000_801F	Clear Interrupt Request Register (DMA_CINT)	8	W (always reads 0)	00h	26.3.12/ 534
4000_8024	Interrupt Request Register (DMA_INT)	32	R/W	0000_0000h	26.3.13/ 535
4000_802C	Error Register (DMA_ERR)	32	R/W	0000_0000h	26.3.14/ 536
4000_8034	Hardware Request Status Register (DMA_HRS)	32	R	0000_0000h	26.3.15/ 538
4000_8044	Enable Asynchronous Request in Stop Register (DMA_EARS)	32	R/W	0000_0000h	26.3.16/ 540
4000_8100	Channel n Priority Register (DMA_DCHPRI3)	8	R/W	See section	26.3.17/ 541
4000_8101	Channel n Priority Register (DMA_DCHPRI2)	8	R/W	See section	26.3.17/ 541
4000_8102	Channel n Priority Register (DMA_DCHPRI1)	8	R/W	See section	26.3.17/ 541
4000_8103	Channel n Priority Register (DMA_DCHPRI0)	8	R/W	See section	26.3.17/ 541
4000_8104	Channel n Priority Register (DMA_DCHPRI7)	8	R/W	See section	26.3.17/ 541
4000_8105	Channel n Priority Register (DMA_DCHPRI6)	8	R/W	See section	26.3.17/ 541
4000_8106	Channel n Priority Register (DMA_DCHPRI5)	8	R/W	See section	26.3.17/ 541
4000_8107	Channel n Priority Register (DMA_DCHPRI4)	8	R/W	See section	26.3.17/ 541
4000_9000	TCD Source Address (DMA_TCD0_SADDR)	32	R/W	Undefined	26.3.18/ 542

Table continues on the next page...

DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4000_9004	TCD Signed Source Address Offset (DMA_TCD0_SOFF)	16	R/W	Undefined	26.3.19/ 542
4000_9006	TCD Transfer Attributes (DMA_TCD0_ATTR)	16	R/W	Undefined	26.3.20/ 543
4000_9008	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD0_NBYTES_MLNO)	32	R/W	Undefined	26.3.21/ 544
4000_9008	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD0_NBYTES_MLOFFNO)	32	R/W	Undefined	26.3.22/ 544
4000_9008	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD0_NBYTES_MLOFFYES)	32	R/W	Undefined	26.3.23/ 546
4000_900C	TCD Last Source Address Adjustment (DMA_TCD0_SLAST)	32	R/W	Undefined	26.3.24/ 547
4000_9010	TCD Destination Address (DMA_TCD0_DADDR)	32	R/W	Undefined	26.3.25/ 547
4000_9014	TCD Signed Destination Address Offset (DMA_TCD0_DOFF)	16	R/W	Undefined	26.3.26/ 548
4000_9016	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD0_CITER_ELINKYES)	16	R/W	Undefined	26.3.27/ 548
4000_9016	DMA_TCD0_CITER_ELINKNO	16	R/W	Undefined	26.3.28/ 550
4000_9018	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD0_DLASTSGA)	32	R/W	Undefined	26.3.29/ 551
4000_901C	TCD Control and Status (DMA_TCD0_CSR)	16	R/W	Undefined	26.3.30/ 551
4000_901E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD0_BITER_ELINKYES)	16	R/W	Undefined	26.3.31/ 554
4000_901E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD0_BITER_ELINKNO)	16	R/W	Undefined	26.3.32/ 555
4000_9020	TCD Source Address (DMA_TCD1_SADDR)	32	R/W	Undefined	26.3.18/ 542
4000_9024	TCD Signed Source Address Offset (DMA_TCD1_SOFF)	16	R/W	Undefined	26.3.19/ 542
4000_9026	TCD Transfer Attributes (DMA_TCD1_ATTR)	16	R/W	Undefined	26.3.20/ 543
4000_9028	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD1_NBYTES_MLNO)	32	R/W	Undefined	26.3.21/ 544
4000_9028	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD1_NBYTES_MLOFFNO)	32	R/W	Undefined	26.3.22/ 544
4000_9028	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD1_NBYTES_MLOFFYES)	32	R/W	Undefined	26.3.23/ 546
4000_902C	TCD Last Source Address Adjustment (DMA_TCD1_SLAST)	32	R/W	Undefined	26.3.24/ 547

Table continues on the next page...

DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4000_9030	TCD Destination Address (DMA_TCD1_DADDR)	32	R/W	Undefined	26.3.25/ 547
4000_9034	TCD Signed Destination Address Offset (DMA_TCD1_DOFF)	16	R/W	Undefined	26.3.26/ 548
4000_9036	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD1_CITER_ELINKYES)	16	R/W	Undefined	26.3.27/ 548
4000_9036	DMA_TCD1_CITER_ELINKNO	16	R/W	Undefined	26.3.28/ 550
4000_9038	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD1_DLASTSGA)	32	R/W	Undefined	26.3.29/ 551
4000_903C	TCD Control and Status (DMA_TCD1_CSR)	16	R/W	Undefined	26.3.30/ 551
4000_903E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD1_BITER_ELINKYES)	16	R/W	Undefined	26.3.31/ 554
4000_903E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD1_BITER_ELINKNO)	16	R/W	Undefined	26.3.32/ 555
4000_9040	TCD Source Address (DMA_TCD2_SADDR)	32	R/W	Undefined	26.3.18/ 542
4000_9044	TCD Signed Source Address Offset (DMA_TCD2_SOFF)	16	R/W	Undefined	26.3.19/ 542
4000_9046	TCD Transfer Attributes (DMA_TCD2_ATTR)	16	R/W	Undefined	26.3.20/ 543
4000_9048	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD2_NBYTES_MLNO)	32	R/W	Undefined	26.3.21/ 544
4000_9048	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD2_NBYTES_MLOFFNO)	32	R/W	Undefined	26.3.22/ 544
4000_9048	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD2_NBYTES_MLOFFYES)	32	R/W	Undefined	26.3.23/ 546
4000_904C	TCD Last Source Address Adjustment (DMA_TCD2_SLAST)	32	R/W	Undefined	26.3.24/ 547
4000_9050	TCD Destination Address (DMA_TCD2_DADDR)	32	R/W	Undefined	26.3.25/ 547
4000_9054	TCD Signed Destination Address Offset (DMA_TCD2_DOFF)	16	R/W	Undefined	26.3.26/ 548
4000_9056	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD2_CITER_ELINKYES)	16	R/W	Undefined	26.3.27/ 548
4000_9056	DMA_TCD2_CITER_ELINKNO	16	R/W	Undefined	26.3.28/ 550
4000_9058	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD2_DLASTSGA)	32	R/W	Undefined	26.3.29/ 551
4000_905C	TCD Control and Status (DMA_TCD2_CSR)	16	R/W	Undefined	26.3.30/ 551

Table continues on the next page...

DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4000_905E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD2_BITER_ELINKYES)	16	R/W	Undefined	26.3.31/ 554
4000_905E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD2_BITER_ELINKNO)	16	R/W	Undefined	26.3.32/ 555
4000_9060	TCD Source Address (DMA_TCD3_SADDR)	32	R/W	Undefined	26.3.18/ 542
4000_9064	TCD Signed Source Address Offset (DMA_TCD3_SOFF)	16	R/W	Undefined	26.3.19/ 542
4000_9066	TCD Transfer Attributes (DMA_TCD3_ATTR)	16	R/W	Undefined	26.3.20/ 543
4000_9068	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD3_NBYTES_MLNO)	32	R/W	Undefined	26.3.21/ 544
4000_9068	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD3_NBYTES_MLOFFNO)	32	R/W	Undefined	26.3.22/ 544
4000_9068	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD3_NBYTES_MLOFFYES)	32	R/W	Undefined	26.3.23/ 546
4000_906C	TCD Last Source Address Adjustment (DMA_TCD3_SLAST)	32	R/W	Undefined	26.3.24/ 547
4000_9070	TCD Destination Address (DMA_TCD3_DADDR)	32	R/W	Undefined	26.3.25/ 547
4000_9074	TCD Signed Destination Address Offset (DMA_TCD3_DOFF)	16	R/W	Undefined	26.3.26/ 548
4000_9076	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD3_CITER_ELINKYES)	16	R/W	Undefined	26.3.27/ 548
4000_9076	DMA_TCD3_CITER_ELINKNO	16	R/W	Undefined	26.3.28/ 550
4000_9078	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD3_DLASTSGA)	32	R/W	Undefined	26.3.29/ 551
4000_907C	TCD Control and Status (DMA_TCD3_CSR)	16	R/W	Undefined	26.3.30/ 551
4000_907E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD3_BITER_ELINKYES)	16	R/W	Undefined	26.3.31/ 554
4000_907E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD3_BITER_ELINKNO)	16	R/W	Undefined	26.3.32/ 555
4000_9080	TCD Source Address (DMA_TCD4_SADDR)	32	R/W	Undefined	26.3.18/ 542
4000_9084	TCD Signed Source Address Offset (DMA_TCD4_SOFF)	16	R/W	Undefined	26.3.19/ 542
4000_9086	TCD Transfer Attributes (DMA_TCD4_ATTR)	16	R/W	Undefined	26.3.20/ 543
4000_9088	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD4_NBYTES_MLNO)	32	R/W	Undefined	26.3.21/ 544

Table continues on the next page...

DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4000_9088	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD4_NBYTES_MLOFFNO)	32	R/W	Undefined	26.3.22/ 544
4000_9088	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD4_NBYTES_MLOFFYES)	32	R/W	Undefined	26.3.23/ 546
4000_908C	TCD Last Source Address Adjustment (DMA_TCD4_SLAST)	32	R/W	Undefined	26.3.24/ 547
4000_9090	TCD Destination Address (DMA_TCD4_DADDR)	32	R/W	Undefined	26.3.25/ 547
4000_9094	TCD Signed Destination Address Offset (DMA_TCD4_DOFF)	16	R/W	Undefined	26.3.26/ 548
4000_9096	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD4_CITER_ELINKYES)	16	R/W	Undefined	26.3.27/ 548
4000_9096	DMA_TCD4_CITER_ELINKNO	16	R/W	Undefined	26.3.28/ 550
4000_9098	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD4_DLASTSGA)	32	R/W	Undefined	26.3.29/ 551
4000_909C	TCD Control and Status (DMA_TCD4_CSR)	16	R/W	Undefined	26.3.30/ 551
4000_909E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD4_BITER_ELINKYES)	16	R/W	Undefined	26.3.31/ 554
4000_909E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD4_BITER_ELINKNO)	16	R/W	Undefined	26.3.32/ 555
4000_90A0	TCD Source Address (DMA_TCD5_SADDR)	32	R/W	Undefined	26.3.18/ 542
4000_90A4	TCD Signed Source Address Offset (DMA_TCD5_SOFF)	16	R/W	Undefined	26.3.19/ 542
4000_90A6	TCD Transfer Attributes (DMA_TCD5_ATTR)	16	R/W	Undefined	26.3.20/ 543
4000_90A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD5_NBYTES_MLNO)	32	R/W	Undefined	26.3.21/ 544
4000_90A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD5_NBYTES_MLOFFNO)	32	R/W	Undefined	26.3.22/ 544
4000_90A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD5_NBYTES_MLOFFYES)	32	R/W	Undefined	26.3.23/ 546
4000_90AC	TCD Last Source Address Adjustment (DMA_TCD5_SLAST)	32	R/W	Undefined	26.3.24/ 547
4000_90B0	TCD Destination Address (DMA_TCD5_DADDR)	32	R/W	Undefined	26.3.25/ 547
4000_90B4	TCD Signed Destination Address Offset (DMA_TCD5_DOFF)	16	R/W	Undefined	26.3.26/ 548
4000_90B6	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD5_CITER_ELINKYES)	16	R/W	Undefined	26.3.27/ 548

Table continues on the next page...

DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4000_90B6	DMA_TCD5_CITER_ELINKNO	16	R/W	Undefined	26.3.28/ 550
4000_90B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD5_DLASTSGA)	32	R/W	Undefined	26.3.29/ 551
4000_90BC	TCD Control and Status (DMA_TCD5_CSR)	16	R/W	Undefined	26.3.30/ 551
4000_90BE	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD5_BITER_ELINKYES)	16	R/W	Undefined	26.3.31/ 554
4000_90BE	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD5_BITER_ELINKNO)	16	R/W	Undefined	26.3.32/ 555
4000_90C0	TCD Source Address (DMA_TCD6_SADDR)	32	R/W	Undefined	26.3.18/ 542
4000_90C4	TCD Signed Source Address Offset (DMA_TCD6_SOFF)	16	R/W	Undefined	26.3.19/ 542
4000_90C6	TCD Transfer Attributes (DMA_TCD6_ATTR)	16	R/W	Undefined	26.3.20/ 543
4000_90C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD6_NBYTES_MLNO)	32	R/W	Undefined	26.3.21/ 544
4000_90C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD6_NBYTES_MLOFFNO)	32	R/W	Undefined	26.3.22/ 544
4000_90C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD6_NBYTES_MLOFFYES)	32	R/W	Undefined	26.3.23/ 546
4000_90CC	TCD Last Source Address Adjustment (DMA_TCD6_SLAST)	32	R/W	Undefined	26.3.24/ 547
4000_90D0	TCD Destination Address (DMA_TCD6_DADDR)	32	R/W	Undefined	26.3.25/ 547
4000_90D4	TCD Signed Destination Address Offset (DMA_TCD6_DOFF)	16	R/W	Undefined	26.3.26/ 548
4000_90D6	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD6_CITER_ELINKYES)	16	R/W	Undefined	26.3.27/ 548
4000_90D6	DMA_TCD6_CITER_ELINKNO	16	R/W	Undefined	26.3.28/ 550
4000_90D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD6_DLASTSGA)	32	R/W	Undefined	26.3.29/ 551
4000_90DC	TCD Control and Status (DMA_TCD6_CSR)	16	R/W	Undefined	26.3.30/ 551
4000_90DE	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD6_BITER_ELINKYES)	16	R/W	Undefined	26.3.31/ 554
4000_90DE	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD6_BITER_ELINKNO)	16	R/W	Undefined	26.3.32/ 555
4000_90E0	TCD Source Address (DMA_TCD7_SADDR)	32	R/W	Undefined	26.3.18/ 542

Table continues on the next page...

DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4000_90E4	TCD Signed Source Address Offset (DMA_TCD7_SOFF)	16	R/W	Undefined	26.3.19/ 542
4000_90E6	TCD Transfer Attributes (DMA_TCD7_ATTR)	16	R/W	Undefined	26.3.20/ 543
4000_90E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD7_NBYTES_MLNO)	32	R/W	Undefined	26.3.21/ 544
4000_90E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD7_NBYTES_MLOFFNO)	32	R/W	Undefined	26.3.22/ 544
4000_90E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD7_NBYTES_MLOFFYES)	32	R/W	Undefined	26.3.23/ 546
4000_90EC	TCD Last Source Address Adjustment (DMA_TCD7_SLAST)	32	R/W	Undefined	26.3.24/ 547
4000_90F0	TCD Destination Address (DMA_TCD7_DADDR)	32	R/W	Undefined	26.3.25/ 547
4000_90F4	TCD Signed Destination Address Offset (DMA_TCD7_DOFF)	16	R/W	Undefined	26.3.26/ 548
4000_90F6	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD7_CITER_ELINKYES)	16	R/W	Undefined	26.3.27/ 548
4000_90F6	DMA_TCD7_CITER_ELINKNO	16	R/W	Undefined	26.3.28/ 550
4000_90F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD7_DLASTSGA)	32	R/W	Undefined	26.3.29/ 551
4000_90FC	TCD Control and Status (DMA_TCD7_CSR)	16	R/W	Undefined	26.3.30/ 551
4000_90FE	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD7_BITER_ELINKYES)	16	R/W	Undefined	26.3.31/ 554
4000_90FE	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD7_BITER_ELINKNO)	16	R/W	Undefined	26.3.32/ 555

26.3.1 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

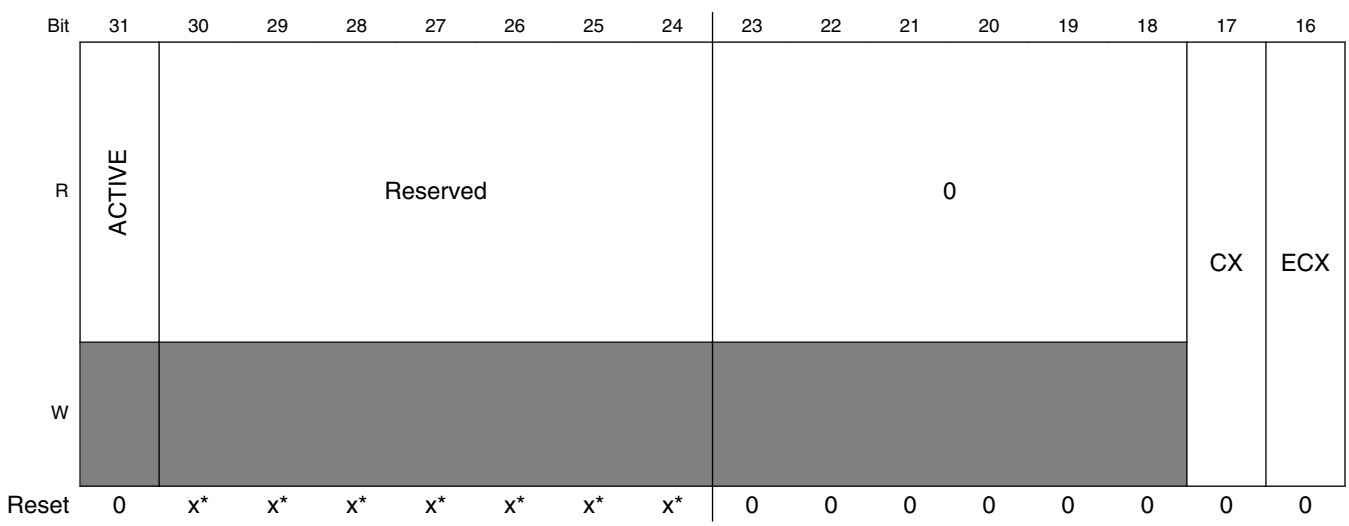
For correct operation, writes to the CR register must be performed only when the DMA channels are inactive; that is, when TCDn_CSR[ACTIVE] bits are cleared.

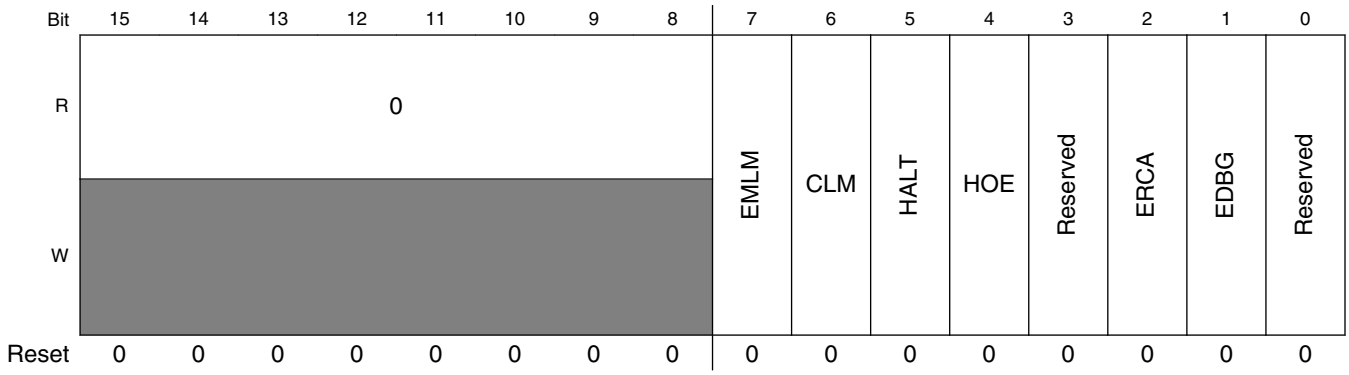
Minor loop offsets are address offset values added to the final source address (TCDn_SADDR) or destination address (TCDn_DADDR) upon minor loop completion. When minor loop offsets are enabled, the minor loop offset (MLOFF) is added to the final source address (TCDn_SADDR), to the final destination address (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





* Notes:
- x = Undefined at reset.

DMA_CR field descriptions

Field	Description
31 ACTIVE	DMA Active Status 0 eDMA is idle. 1 eDMA is executing a channel.
30–24 Reserved	This field is reserved. Reserved
23–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 CX	Cancel Transfer 0 Normal operation 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 ECX	Error Cancel Transfer 0 Normal operation 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 EMLM	Enable Minor Loop Mapping 0 Disabled. TCDn.word2 is defined as a 32-bit NBYTES field. 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 CLM	Continuous Link Mode

Table continues on the next page...

DMA_CR field descriptions (continued)

Field	Description
	NOTE: Do not use continuous link mode with a channel linking to itself if there is only one minor loop iteration per service request, e.g., if the channel's NBYTES value is the same as either the source or destination size. The same data transfer profile can be achieved by simply increasing the NBYTES value, which provides more efficient, faster processing. 0 A minor loop channel link made to itself goes through channel arbitration before being activated again. 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 HALT	Halt DMA Operations 0 Normal operation 1 Stall the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this bit is cleared.
4 HOE	Halt On Error 0 Normal operation 1 Any error causes the HALT bit to set. Subsequently, all service requests are ignored until the HALT bit is cleared.
3 Reserved	This field is reserved. Reserved
2 ERCA	Enable Round Robin Channel Arbitration 0 Fixed priority arbitration is used for channel selection . 1 Round robin arbitration is used for channel selection .
1 EDBG	Enable Debug 0 When in debug mode, the DMA continues to operate. 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 Reserved	This field is reserved. Reserved

26.3.2 Error Status Register (DMA_ES)

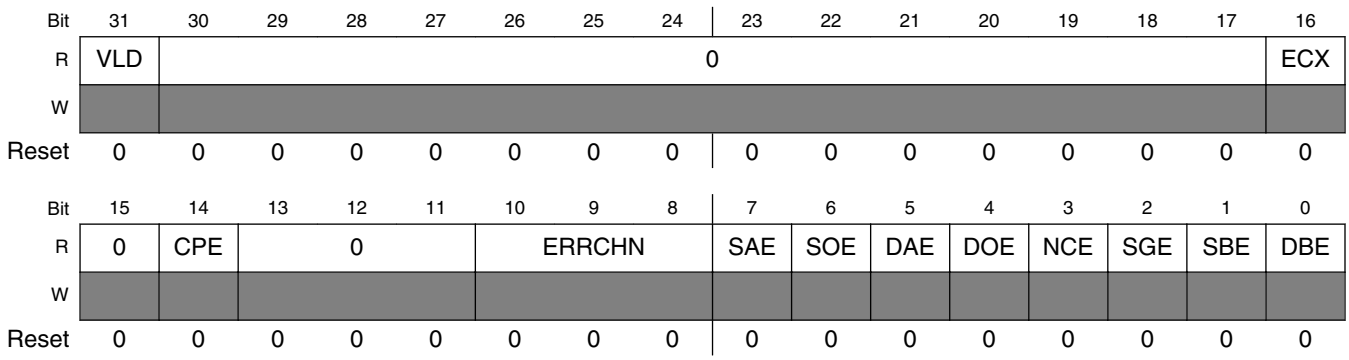
The ES provides information concerning the last recorded channel error. Channel errors can be caused by:

- 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

- A bit in the ES shows an uncorrectable error occurred while the device has ECC protection on the TCD SRAM
- A cancel transfer with error bit that will be set when a transfer is canceled via the corresponding cancel transfer control bit

See the Error Reporting and Handling section for more details.

Address: 4000_8000h base + 4h offset = 4000_8004h



DMA_ES field descriptions

Field	Description
31 VLD	Logical OR of all ERR status bits 0 No ERR bits are set. 1 At least one ERR bit is set indicating a valid error exists that has not been cleared.
30–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 ECX	Transfer Canceled 0 No canceled transfers 1 The last recorded entry was a canceled transfer by the error cancel transfer input
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 CPE	Channel Priority Error 0 No channel priority error 1 The last recorded error was a configuration error in the channel priorities . Channel priorities are not unique.
13–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–8 ERRCHN	Error Channel Number or Canceled Channel Number The channel number of the last recorded error, excluding CPE errors, or last recorded error canceled transfer.
7 SAE	Source Address Error 0 No source address configuration error. 1 The last recorded error was a configuration error detected in the TCDn_SADDR field. TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE].

Table continues on the next page...

DMA_ES field descriptions (continued)

Field	Description
6 SOE	Source Offset Error 0 No source offset configuration error 1 The last recorded error was a configuration error detected in the TCDn_SOFF field. TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE].
5 DAE	Destination Address Error 0 No destination address configuration error 1 The last recorded error was a configuration error detected in the TCDn_DADDR field. TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE].
4 DOE	Destination Offset Error 0 No destination offset configuration error 1 The last recorded error was a configuration error detected in the TCDn_DOFF field. TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE].
3 NCE	NBYTES/CITER Configuration Error 0 No NBYTES/CITER configuration error 1 The last recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields. • TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE], or • TCDn_CITER[CITER] is equal to zero, or • TCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK]
2 SGE	Scatter/Gather Configuration Error 0 No scatter/gather configuration error 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 SBE	Source Bus Error 0 No source bus error 1 The last recorded error was a bus error on a source read
0 DBE	Destination Bus Error 0 No destination bus error 1 The last recorded error was a bus error on a destination write

26.3.3 Enable Request Register (DMA_ERQ)

The ERQ register provides a bit map for the 8 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 registers. These 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	0								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
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 ERQ7	Enable DMA Request 7 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
6 ERQ6	Enable DMA Request 6 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
5 ERQ5	Enable DMA Request 5 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
4 ERQ4	Enable DMA Request 4 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
3 ERQ3	Enable DMA Request 3 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
2 ERQ2	Enable DMA Request 2 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
1 ERQ1	Enable DMA Request 1 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
0 ERQ0	Enable DMA Request 0

Table continues on the next page...

DMA_ERQ field descriptions (continued)

Field	Description
	0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

26.3.4 Enable Error Interrupt Register (DMA_EEI)

The EEI register provides a bit map for the 8 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. These registers are provided so that 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	0								EEI7	EEI6	EEI5	EEI4	EEI3	EEI2	EEI1	EEI0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_EEI field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 EEI7	Enable Error Interrupt 7 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
6 EEI6	Enable Error Interrupt 6 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
5 EEI5	Enable Error Interrupt 5 0 The error signal for corresponding channel does not generate an error interrupt 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
4 EEI4	Enable Error Interrupt 4 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
3 EEI3	Enable Error Interrupt 3 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
2 EEI2	Enable Error Interrupt 2 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
1 EEI1	Enable Error Interrupt 1 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
0 EEI0	Enable Error Interrupt 0 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

26.3.5 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 NOP	No Op enable

Table continues on the next page...

DMA_CEEI field descriptions (continued)

Field	Description
	0 Normal operation 1 No operation, ignore the other bits in this register
6 CAEE	Clear All Enable Error Interrupts 0 Clear only the EEI bit specified in the CEEI field 1 Clear all bits in EEI
5–3 Reserved	This field is reserved.
CEEI	Clear Enable Error Interrupt Clears the corresponding bit in EEI

26.3.6 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 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
6 SAEE	Sets All Enable Error Interrupts 0 Set only the EEI bit specified in the SEEI field. 1 Sets all bits in EEI
5–3 Reserved	This field is reserved.
SEEI	Set Enable Error Interrupt

Table continues on the next page...

DMA_SEEI field descriptions (continued)

Field	Description
	Sets the corresponding bit in EEI

26.3.7 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 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
6 CAER	Clear All Enable Requests 0 Clear only the ERQ bit specified in the CERQ field 1 Clear all bits in ERQ
5–3 Reserved	This field is reserved.
CERQ	Clear Enable Request Clears the corresponding bit in ERQ.

26.3.8 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 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
6 SAER	Set All Enable Requests 0 Set only the ERQ bit specified in the SERQ field 1 Set all bits in ERQ
5–3 Reserved	This field is reserved.
SERQ	Set Enable Request Sets the corresponding bit in ERQ.

26.3.9 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 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
6 CADN	Clears All DONE Bits 0 Clears only the TCDn_CSR[DONE] bit specified in the CDNE field 1 Clears all bits in TCDn_CSR[DONE]
5–3 Reserved	This field is reserved.
CDNE	Clear DONE Bit Clears the corresponding bit in TCDn_CSR[DONE]

26.3.10 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 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
6 SAST	Set All START Bits (activates all channels) 0 Set only the TCDn_CSR[START] bit specified in the SSRT field 1 Set all bits in TCDn_CSR[START]
5–3 Reserved	This field is reserved.
SSRT	Set START Bit Sets the corresponding bit in TCDn_CSR[START]

26.3.11 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 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
6 CAEI	Clear All Error Indicators 0 Clear only the ERR bit specified in the CERR field 1 Clear all bits in ERR
5–3 Reserved	This field is reserved.
CERR	Clear Error Indicator Clears the corresponding bit in ERR

26.3.12 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 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
6 CAIR	Clear All Interrupt Requests 0 Clear only the INT bit specified in the CINT field 1 Clear all bits in INT
5–3 Reserved	This field is reserved.
CINT	Clear Interrupt Request Clears the corresponding bit in INT

26.3.13 Interrupt Request Register (DMA_INT)

The INT register provides a bit map for the 8 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. 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	0								INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_INT field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 INT7	Interrupt Request 7 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
6 INT6	Interrupt Request 6 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
5 INT5	Interrupt Request 5

Table continues on the next page...

DMA_INT field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
4 INT4	Interrupt Request 4 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
3 INT3	Interrupt Request 3 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
2 INT2	Interrupt Request 2 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
1 INT1	Interrupt Request 1 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
0 INT0	Interrupt Request 0 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active

26.3.14 Error Register (DMA_ERR)

The ERR provides a bit map for the 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

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								ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ERR field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 ERR7	Error In Channel 7 0 An error in this channel has not occurred 1 An error in this channel has occurred
6 ERR6	Error In Channel 6 0 An error in this channel has not occurred 1 An error in this channel has occurred
5 ERR5	Error In Channel 5 0 An error in this channel has not occurred 1 An error in this channel has occurred
4 ERR4	Error In Channel 4 0 An error in this channel has not occurred 1 An error in this channel has occurred
3 ERR3	Error In Channel 3 0 An error in this channel has not occurred 1 An error in this channel has occurred
2 ERR2	Error In Channel 2 0 An error in this channel has not occurred 1 An error in this channel has occurred
1 ERR1	Error In Channel 1 0 An error in this channel has not occurred 1 An error in this channel has occurred
0 ERR0	Error In Channel 0 0 An error in this channel has not occurred 1 An error in this channel has occurred

26.3.15 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	0								HRS7	HRS6	HRS5	HRS4	HRS3	HRS2	HRS1	HRS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_HRS field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 HRS7	Hardware Request Status Channel 7 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 7 is not present 1 A hardware service request for channel 7 is present
6 HRS6	Hardware Request Status Channel 6 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 6 is not present 1 A hardware service request for channel 6 is present
5 HRS5	Hardware Request Status Channel 5

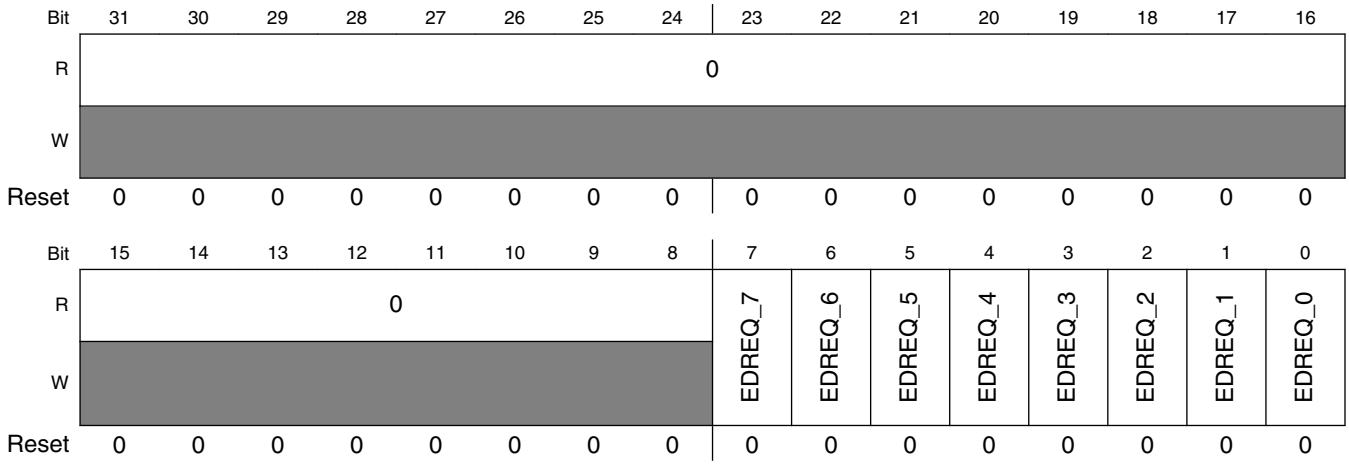
Table continues on the next page...

DMA_HRS field descriptions (continued)

Field	Description
	The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 5 is not present 1 A hardware service request for channel 5 is present
4 HRS4	Hardware Request Status Channel 4 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 4 is not present 1 A hardware service request for channel 4 is present
3 HRS3	Hardware Request Status Channel 3 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 3 is not present 1 A hardware service request for channel 3 is present
2 HRS2	Hardware Request Status Channel 2 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 2 is not present 1 A hardware service request for channel 2 is present
1 HRS1	Hardware Request Status Channel 1 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 1 is not present 1 A hardware service request for channel 1 is present
0 HRS0	Hardware Request Status Channel 0 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 0 is not present 1 A hardware service request for channel 0 is present

26.3.16 Enable Asynchronous Request in Stop Register (DMA_EARS)

Address: 4000_8000h base + 44h offset = 4000_8044h



DMA_EARS field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 EDREQ_7	Enable asynchronous DMA request in stop mode for channel 7 0 Disable asynchronous DMA request for channel 7. 1 Enable asynchronous DMA request for channel 7.
6 EDREQ_6	Enable asynchronous DMA request in stop mode for channel 6 0 Disable asynchronous DMA request for channel 6. 1 Enable asynchronous DMA request for channel 6.
5 EDREQ_5	Enable asynchronous DMA request in stop mode for channel 5 0 Disable asynchronous DMA request for channel 5. 1 Enable asynchronous DMA request for channel 5.
4 EDREQ_4	Enable asynchronous DMA request in stop mode for channel 4 0 Disable asynchronous DMA request for channel 4. 1 Enable asynchronous DMA request for channel 4.
3 EDREQ_3	Enable asynchronous DMA request in stop mode for channel 3. 0 Disable asynchronous DMA request for channel 3. 1 Enable asynchronous DMA request for channel 3.
2 EDREQ_2	Enable asynchronous DMA request in stop mode for channel 2. 0 Disable asynchronous DMA request for channel 2. 1 Enable asynchronous DMA request for channel 2.
1 EDREQ_1	Enable asynchronous DMA request in stop mode for channel 1.

Table continues on the next page...

DMA_EARS field descriptions (continued)

Field	Description
	0 Disable asynchronous DMA request for channel 1 1 Enable asynchronous DMA request for channel 1.
0 EDREQ_0	Enable asynchronous DMA request in stop mode for channel 0. 0 Disable asynchronous DMA request for channel 0. 1 Enable asynchronous DMA request for channel 0.

26.3.17 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 higher 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 7.

Address: 4000_8000h base + 100h offset + (1d × i), where i=0d to 7d

Bit	7	6	5	4	3	2	1	0
Read	ECP	DPA	0			CHPRI		
Write								
Reset	0	0	0	0	0	*	*	*

* Notes:
- CHPRI field: See bit field description.

DMA_DCHPRI*n* field descriptions

Field	Description
7 ECP	Enable Channel Preemption. 0 Channel n cannot be suspended by a higher priority channel's service request. 1 Channel n can be temporarily suspended by the service request of a higher priority channel.
6 DPA	Disable Preempt Ability. 0 Channel n can suspend a lower priority channel. 1 Channel n cannot suspend any channel, regardless of channel priority.
5–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CHPRI	Channel n Arbitration Priority Channel priority when fixed-priority arbitration is enabled **NOTE:** Reset value for the channel priority field, CHPRI, is equal to the corresponding channel number for each priority register, that is, DCHPRI7[CHPRI] = 0b111.

26.3.18 TCD Source Address (DMA_TCD*n*_SADDR)

Address: 4000_8000h base + 1000h offset + (32d × 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 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
SADDR	Source Address Memory address pointing to the source data.

26.3.19 TCD Signed Source Address Offset (DMA_TCD*n*_SOFF)

Address: 4000_8000h base + 1004h offset + (32d × i), where i=0d to 7d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read 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
SOFF	Source address signed offset Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

26.3.20 TCD Transfer Attributes (DMA_TCD*n*_ATTR)

Address: 4000_8000h base + 1006h offset + (32d × i), where i=0d to 7d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read 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 SMOD	Source Address Modulo 0 Source address modulo feature is disabled ≠0 This value defines a specific address range specified to be the value after SADDR + SOFF calculation is performed on the original register value. Setting 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 SSIZE	Source data transfer size **NOTE:** Using a Reserved value causes a configuration error. 000 8-bit 001 16-bit 010 32-bit 011 Reserved 100 16-byte 101 32-byte 110 Reserved 111 Reserved
7–3 DMOD	Destination Address Modulo See the SMOD definition
DSIZE	Destination data transfer size See the SSIZE definition

26.3.21 TCD Minor Byte Count (Minor Loop Mapping 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 the definition of TCD word 2.

Address: 4000_8000h base + 1008h offset + (32d × 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 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
NBYTES	Minor Byte Transfer Count 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. It can, however, 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. **NOTE:** An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer.

26.3.22 TCD Signed Minor Loop Offset (Minor Loop Mapping 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 7d

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 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR 1 The minor loop offset is applied to the SADDR
30 DMLOE	Destination Minor Loop Offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR 1 The minor loop offset is applied to the DADDR
NBYTES	Minor Byte Transfer Count 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. It can, however, 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.

26.3.23 TCD Signed Minor Loop Offset (Minor Loop Mapping 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 7d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SMLOE	DMLOE	MLOFF													
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 W	MLOFF						NBYTES									
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_MLOFFYES field descriptions

Field	Description
31 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR 1 The minor loop offset is applied to the SADDR
30 DMLOE	Destination Minor Loop Offset enable 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 1 The minor loop offset is applied to the DADDR
29–10 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.
NBYTES	Minor Byte Transfer Count 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. It can, however, 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.

26.3.24 TCD Last Source Address Adjustment (DMA_TCD*n*_SLAST)

Address: 4000_8000h base + 100Ch offset + (32d × 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 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
SLAST	Last Source Address Adjustment 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. This register uses two's complement notation; the overflow bit is discarded.

26.3.25 TCD Destination Address (DMA_TCD*n*_DADDR)

Address: 4000_8000h base + 1010h offset + (32d × 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 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
DADDR	Destination Address Memory address pointing to the destination data.

26.3.26 TCD Signed Destination Address Offset (DMA_TCD*n*_DOFF)

Address: 4000_8000h base + 1014h offset + (32d × i), where i=0d to 7d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read 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
DOFF	Destination Address Signed Offset Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed.

26.3.27 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 7d

Bit	15	14	13	12	11	10	9	8
Read Write	ELINK	0			LINKCH			CITER
Reset	x*	x*	x*	x*	x*	x*	x*	x*

Bit	7	6	5	4	3	2	1	0
Read 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 ELINK	Enable channel-to-channel linking on minor-loop complete 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. 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. **NOTE:** This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
14–12 Reserved	This field is reserved.
11–9 LINKCH	Minor Loop Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request to the channel defined by this field by setting that channel's TCDn_CSR[START] bit.
CITER	Current Major Iteration Count 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, for example, 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. **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. **NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

26.3.28 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 7d

Bit	15	14	13	12	11	10	9	8
Read Write	ELINK	CITER						
Reset	x*	x*	x*	x*	x*	x*	x*	x*

Bit	7	6	5	4	3	2	1	0
Read 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 ELINK	Enable channel-to-channel linking on minor-loop complete 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. 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. **NOTE:** This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
CITER	Current Major Iteration Count 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, for example, 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. **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. **NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

26.3.29 TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD*n*_DLASTSGA)

Address: 4000_8000h base + 1018h offset + (32d × 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 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
DLASTSGA	Destination last address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather). If (TCDn_CSR[ESG] = 0) then: • 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. • This field uses two's complement notation for the final destination address adjustment. Otherwise: • 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, otherwise a configuration error is reported.

26.3.30 TCD Control and Status (DMA_TCD*n*_CSR)

Address: 4000_8000h base + 101Ch offset + (32d × i), where i=0d to 7d

Bit	15	14	13	12	11	10	9	8
Read	BWC					MAJORLINKCH		
Write			0					
Reset	x*	x*	x*	x*	x*	x*	x*	x*

Bit	7	6	5	4	3	2	1	0
Read	DONE	ACTIVE	MAJORELINK	ESG	DREQ	INTHALF	INTMAJOR	START
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

DMA_TCD*n*_CSR field descriptions

Field	Description
15–14 BWC	Bandwidth Control Throttles the amount of bus bandwidth consumed by the eDMA. Generally, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the crossbar switch. 00 No eDMA engine stalls. 01 Reserved 10 eDMA engine stalls for 4 cycles after each R/W. 11 eDMA engine stalls for 8 cycles after each R/W.
13–11 Reserved	This field is reserved.
10–8 MAJORLINKCH	Major Loop Link Channel Number If (MAJORELINK = 0) then: • No channel-to-channel linking, or chaining, is performed after the major loop counter is exhausted. Otherwise: • After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] bit.
7 DONE	Channel Done 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. **NOTE:** This bit must be cleared to write the MAJORELINK or ESG bits.
6 ACTIVE	Channel Active This flag signals the channel is currently in execution. It is set when channel service begins, and is cleared by the eDMA as the minor loop completes or when any error condition is detected.
5 MAJORELINK	Enable channel-to-channel linking on major loop complete 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. **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. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
4 ESG	Enable Scatter/Gather Processing 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. **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. 0 The current channel's TCD is normal format. 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.

Table continues on the next page...

DMA_TCD*n*_CSR field descriptions (continued)

Field	Description
3 DREQ	Disable Request If this flag is set, the eDMA hardware automatically clears the corresponding ERQ bit when the current major iteration count reaches zero. 0 The channel's ERQ bit is not affected. 1 The channel's ERQ bit is cleared when the major loop is complete.
2 INTHALF	Enable an interrupt when major counter is half complete. 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, also known as ping-pong, schemes or other types of data movement where the processor needs an early indication of the transfer's progress. **NOTE:** If BITER = 1, do not use INTHALF. Use INTMAJOR instead. 0 The half-point interrupt is disabled. 1 The half-point interrupt is enabled.
1 INTMAJOR	Enable an interrupt when major iteration count completes. 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. 0 The end-of-major loop interrupt is disabled. 1 The end-of-major loop interrupt is enabled.
0 START	Channel Start If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution. 0 The channel is not explicitly started. 1 The channel is explicitly started via a software initiated service request.

26.3.31 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 7d

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 ELINK	Enables channel-to-channel linking on minor loop complete 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. **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 are reloaded into the CITER field. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
14–12 Reserved	This field is reserved.
11–9 LINKCH	Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] bit. **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 are reloaded into the CITER field.
BITER	Starting major iteration count 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 are 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.

26.3.32 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 7d

Bit	15	14	13	12	11	10	9	8
Read Write	ELINK	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_ELINKNO field descriptions

Field	Description
15 ELINK	Enables channel-to-channel linking on minor loop complete 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. **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 are reloaded into the CITER field. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
BITER	Starting Major Iteration Count 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. **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

Table continues on the next page...

DMA_TCD*n*_BITER_ELINKNO field descriptions (continued)

Field	Description
	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.

26.4 Functional description

The operation of the eDMA is described in the following subsections.

26.4.1 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:

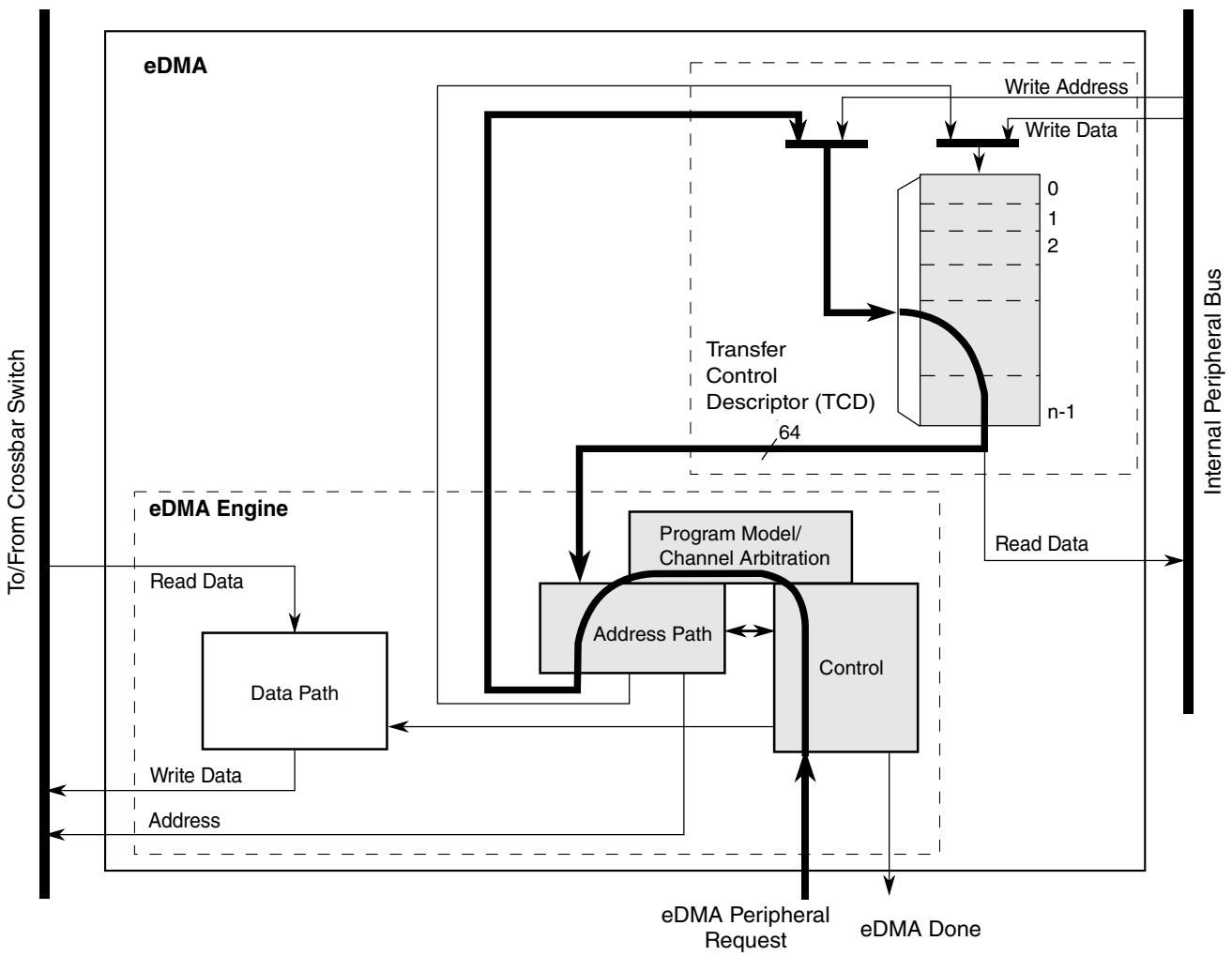


Figure 26-2. eDMA operation, part 1

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:

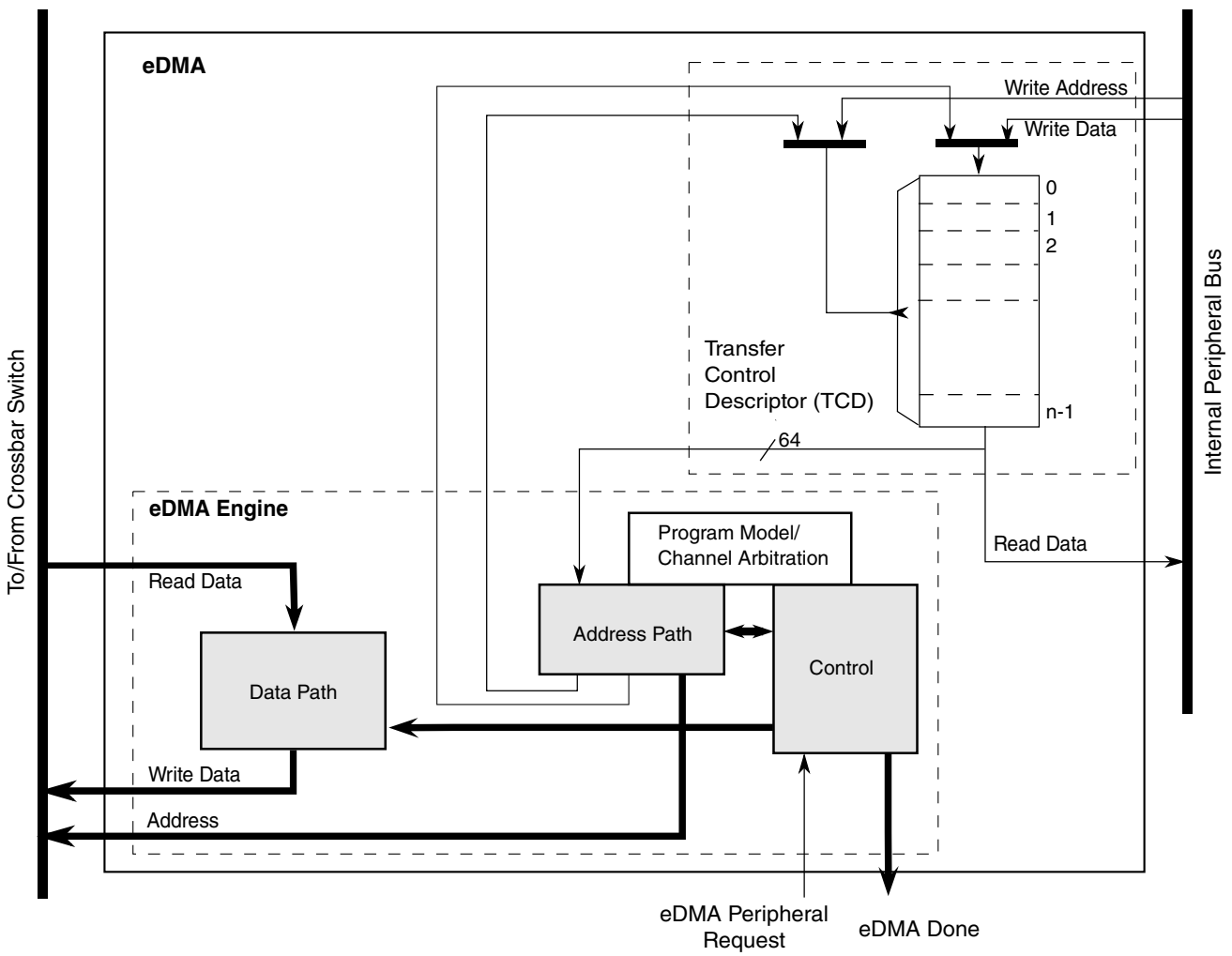


Figure 26-3. eDMA operation, part 2

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, for example, SADDR, DADDR, CITER. If the major iteration count is exhausted, additional operations are performed. These include the final address adjustments and reloading of the BITER field into the CITER. 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.

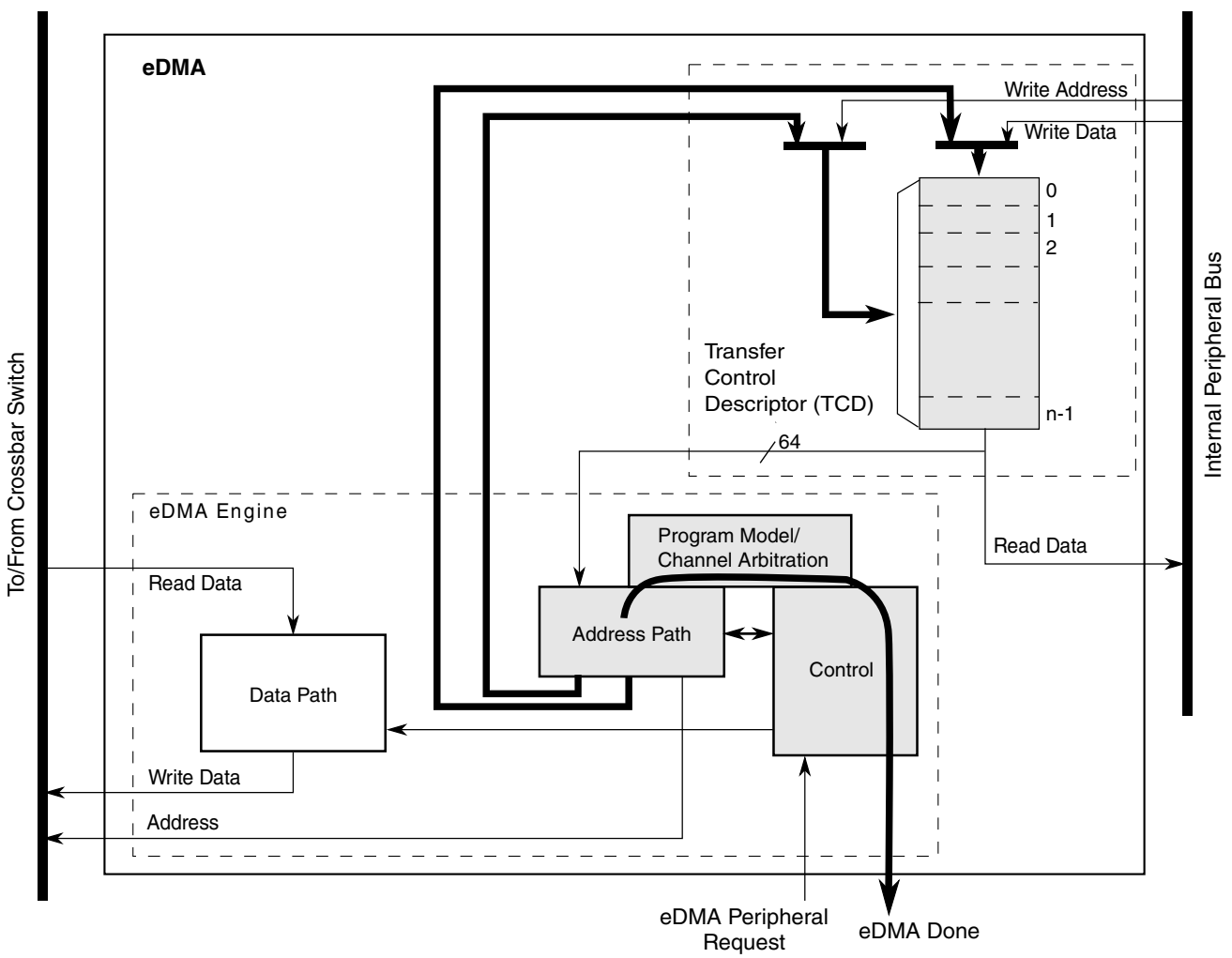


Figure 26-4. eDMA operation, part 3

26.4.2 Fault 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.

NOTE

When two channels have the same priority, a channel priority error exists and will be reported in the Error Status register. However, the channel number will not be reported in the Error Status register. When all of the channel priorities within a group are not unique, the channel number selected by arbitration is undetermined.

To aid in Channel Priority Error (CPE) debug, set the Halt On Error bit in the DMA's Control Register. If all of the channel priorities within a group are not unique, the DMA will be halted after the CPE error is recorded. The DMA will remain halted and will not process any channel service requests. Once all of the channel priorities are set to unique numbers, the DMA may be enabled again by clearing the Halt bit.

- 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.

NOTE

The cancel transfer request allows the user to stop a large data transfer in the event the full data transfer is no longer needed. The cancel transfer bit does not abort the channel. It simply stops the transferring of data and then retires the channel through its normal shutdown sequence. The application

software must handle the context of the cancel. If an interrupt is desired (or not), then the interrupt should be enabled (or disabled) before the cancel request. The application software must clean up the transfer control descriptor since the full transfer did not occur.

The occurrence of any error causes the eDMA engine to stop normal processing of the active channel immediately (it goes to its error processing states and the transaction to the system bus still has pipeline effect), and the appropriate channel 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.

26.4.3 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.

26.4.4 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.

26.4.4.1 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

NOTE

All architectures will not meet the assumptions listed above. See the SRAM configuration section for more information.

This table compares peak transfer rates based on different possible system speeds. Specific chips/devices may not support all system speeds listed.

Table 26-4. eDMA peak transfer rates (Mbytes/sec)

System Speed, Width	Internal SRAM-to-Internal SRAM	32 bit internal peripheral bus-to-Internal SRAM	Internal SRAM-to-32 bit internal peripheral bus
66.7 MHz, 32 bit	133.3	66.7	53.3
83.3 MHz, 32 bit	166.7	83.3	66.7
100.0 MHz, 32 bit	200.0	100.0	80.0
133.3 MHz, 32 bit	266.7	133.3	106.7
150.0 MHz, 32 bit	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.

26.4.4.2 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 26-5. 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.
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.

Table continues on the next page...

Table 26-5. Hardware service request process (continued)

Cycle		Description
With internal peripheral bus read and internal SRAM write	With SRAM read and internal peripheral bus 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 26-6. 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 26-7. Peak request formula operands

Operand	Description
PEAKreq	Peak request rate

Table continues on the next page...

Table 26-7. Peak request formula operands (continued)

Operand	Description
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)

26.4.4.3 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.

26.5 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

26.5.1 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 ERQH and ERQL registers.
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 26-8. 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).

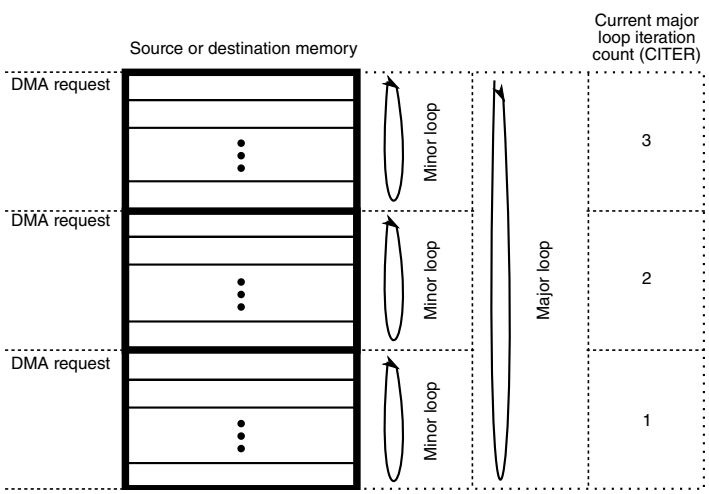


Figure 26-5. Example of multiple loop iterations

The following figure lists the memory array terms and how the TCD settings interrelate.

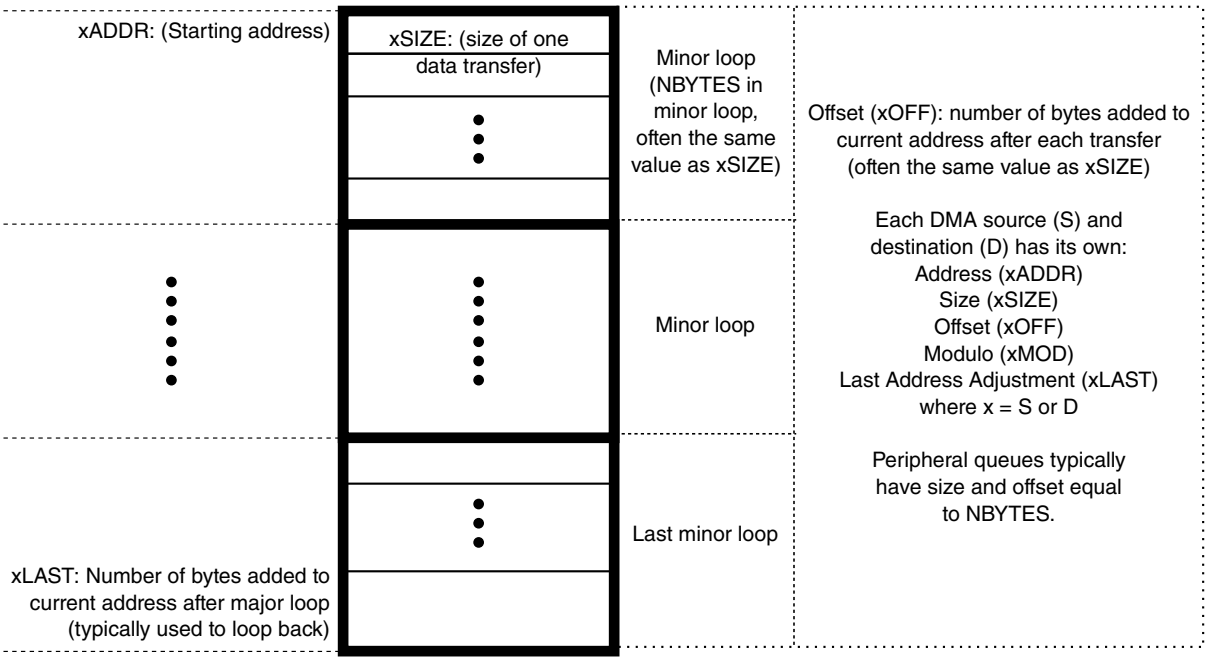


Figure 26-6. Memory array terms

26.5.2 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.

26.5.3 Arbitration mode considerations

This section discusses arbitration considerations for the eDMA.

26.5.3.1 Fixed channel arbitration

In this mode, the channel service request from the highest priority channel is selected to execute.

26.5.3.2 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.

26.5.4 Performing DMA transfers

This section presents examples on how to perform DMA transfers with the eDMA.

26.5.4.1 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.

26.5.4.2 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.

26.5.4.3 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 26-9. Modulo example

Transfer Number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C
5	0x12345670
6	0x12345674

26.5.5 Monitoring transfer descriptor status

This section discusses how to monitor eDMA status.

26.5.5.1 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.

26.5.5.2 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.

26.5.5.3 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.

26.5.6 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 26-10. 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

26.5.7 Dynamic programming

This section provides recommended methods to change the programming model during channel execution.

26.5.7.1 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.

26.5.7.2 Dynamic channel linking

Dynamic channel linking is the process of setting the TCD.major.e_link bit during channel execution (see the diagram in TCD structure). 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.

26.5.7.3 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.

26.5.7.3.1 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).

26.5.7.3.2 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 27
External Watchdog Monitor (EWM)

27.1 Chip-specific EWM information

27.1.1 EWM clocks

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

Table 27-1. EWM clock connections

Module clock	Chip clock
Low Power Clock	1 kHz LPO Clock

27.1.2 EWM low-power modes

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

Table 27-2. EWM low-power modes

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

27.1.3 $\overline{\text{EWM\_OUT}}$ pin state in low power modes

During Wait, Stop and Power Down modes the $\overline{\text{EWM\_OUT}}$ pin preserves 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.

27.2 Introduction

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.

27.2.1 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 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.

27.2.2 Modes of Operation

This section describes the module's operating modes.

27.2.2.1 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.

27.2.2.2 Wait Mode

The EWM module treats the stop and wait modes as the same. EWM functionality remains the same in both of these modes.

27.2.2.3 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.

27.2.3 Block Diagram

This figure shows the EWM block diagram.

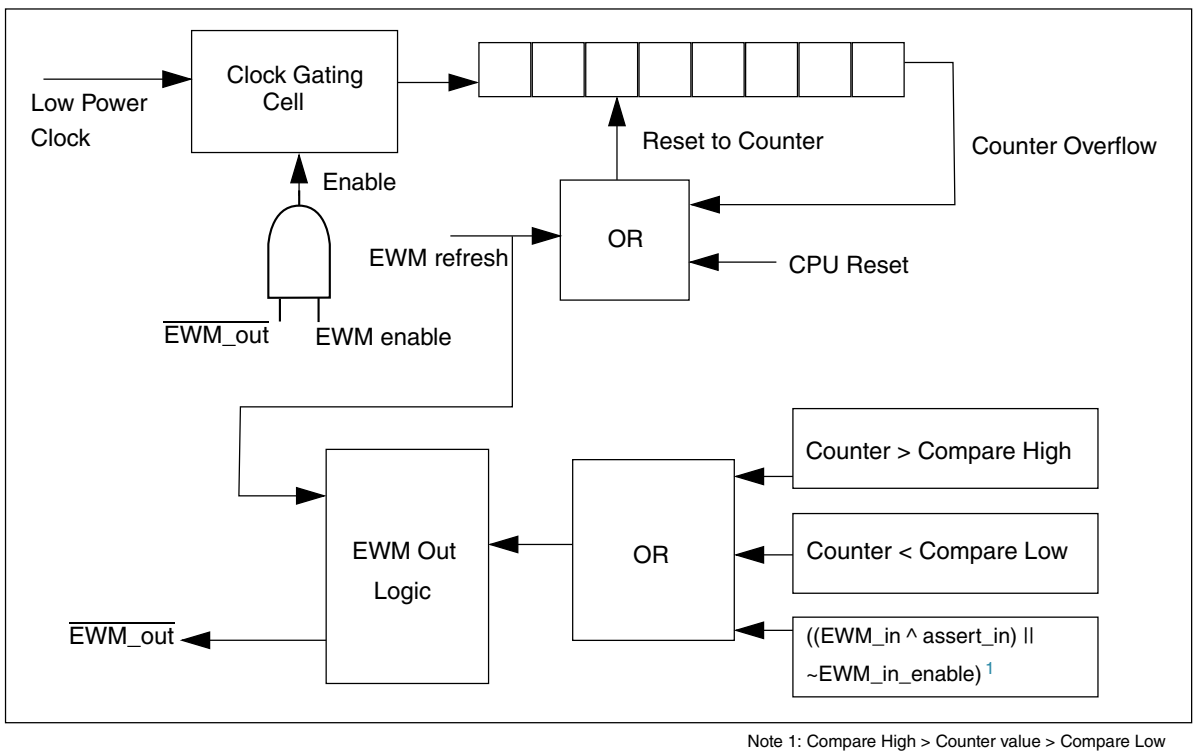


Figure 27-1. EWM Block Diagram

27.3 EWM Signal Descriptions

The EWM has two external signals, as shown in the following table.

Table 27-3. 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

27.4 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	27.4.1/585
4006_1001	Service Register (EWM_SERV)	8	W (always reads 0)	00h	27.4.2/586
4006_1002	Compare Low Register (EWM_CMPL)	8	R/W	00h	27.4.3/586
4006_1003	Compare High Register (EWM_CMPH)	8	R/W	FFh	27.4.4/587
4006_1005	Clock Prescaler Register (EWM_CLKPRESCALER)	8	R/W	00h	27.4.5/587

27.4.1 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 INTEN	Interrupt Enable. 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 INEN	Input Enable. This bit when set, enables the EWM_in port.
1 ASSIN	EWM_in's Assertion State Select.

Table continues on the next page...

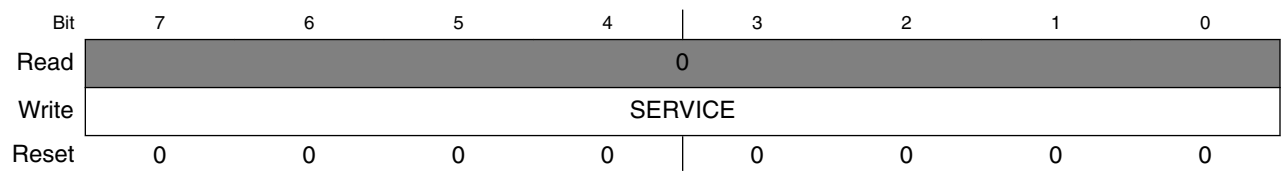
EWM_CTRL field descriptions (continued)

Field	Description
	Default assert state of the EWM_in signal is logic zero. Setting ASSIN bit inverts the assert state to a logic one.
0 EWMEN	EWM enable. 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.

27.4.2 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



EWM_SERV field descriptions

Field	Description
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. • The first or second data byte is not written correctly. • 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*.

27.4.3 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 Write	COMPAREL							
Reset	0	0	0	0	0	0	0	0

EWM_CMPL field descriptions

Field	Description
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.

27.4.4 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 Write	COMPAREH							
Reset	1	1	1	1	1	1	1	1

EWM_CMPH field descriptions

Field	Description
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.

27.4.5 Clock Prescaler Register (EWM_CLKPRESCALER)

This CLKPRESCALER register is reset to 0x00 after a CPU reset.

NOTE

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

NOTE

Write the required prescaler value before enabling the EWM.

NOTE

The implementation of this register is chip-specific. See the Chip Configuration details.

Address: 4006_1000h base + 5h offset = 4006_1005h

Bit	7	6	5	4	3	2	1	0
Read Write	CLK_DIV							
Reset	0	0	0	0	0	0	0	0

EWM_CLKPRESCALER field descriptions

Field	Description
CLK_DIV	Selected low power clock source for running the EWM counter can be prescaled as below. • Prescaled clock frequency = low power clock source frequency / (1 + CLK_DIV)

27.5 Functional Description

The following sections describe functional details of the EWM module.

27.5.1 The $\overline{\text{EWM\_out}}$ Signal

The $\overline{\text{EWM\_out}}$ is a digital output signal used to gate an external circuit (application specific) that controls critical safety functions. For example, the $\overline{\text{EWM\_out}}$ could be connected to the high voltage transistors circuits that control an AC motor in a large appliance.

The $\overline{\text{EWM\_out}}$ 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 $\overline{\text{EWM\_out}}$ 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 $\overline{\text{EWM\_out}}$ pin)

On a normal reset, the $\overline{\text{EWM\_out}}$ is asserted. To deassert the $\overline{\text{EWM\_out}}$, set EWMEN bit in the CTRL register to enable the EWM.

If the $\overline{\text{EWM\_out}}$ signal shares its pad with a digital I/O pin, on reset this actual pad defers to being an input signal. It takes the $\overline{\text{EWM\_out}}$ output condition only after you enable the EWM by the EWMEN bit in the CTRL register.

When the $\overline{\text{EWM\_out}}$ pin is asserted, it can only be deasserted by forcing a MCU reset.

Note

$\overline{\text{EWM\_out}}$ pad must be in pull down state when EWM functionality is used and when EWM is under Reset.

27.5.2 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.

27.5.3 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.

27.5.4 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, $\overline{\text{EWM\_out}}$ is asserted.

It is illegal to program CMPL and CMPH with same value. In this case, as soon as counter reaches (CMPL + 1), $\overline{\text{EWM\_out}}$ is asserted.

27.5.5 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 27-4. 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 $\overline{\text{EWM\_out}}$ pin remains in the deasserted state. **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 $\overline{\text{EWM\_out}}$ pin (irrespective of the EWM_in pin). The $\overline{\text{EWM\_out}}$ 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 $\overline{\text{EWM\_out}}$ pin (irrespective of the EWM_in pin). The $\overline{\text{EWM\_out}}$ pin is expected to gate critical safety circuits.

Any illegal service on EWM has no effect on $\overline{\text{EWM\_out}}$.

27.5.6 EWM Interrupt

When $\overline{\text{EWM\_out}}$ 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 $\overline{\text{EWM\_out}}$. The $\overline{\text{EWM\_out}}$ signal can be deasserted only by forcing a system reset.

27.5.7 Counter clock prescaler

The EWM counter clock source can be prescaled by a clock divider, by programming CLKPRESCALER[CLK_DIV]. This divided clock is used to run the EWM counter.

NOTE

The divided clock used to run the EWM counter must be no more than half the frequency of the bus clock.

Chapter 28
Watchdog Timer (WDOG)

28.1 Chip-specific WDOG information

28.1.1 WDOG clocks

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

Table 28-1. 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

28.1.2 WDOG low-power modes

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

Table 28-2. WDOG low-power modes

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

28.2 Introduction

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.

28.3 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.

28.4 Functional overview

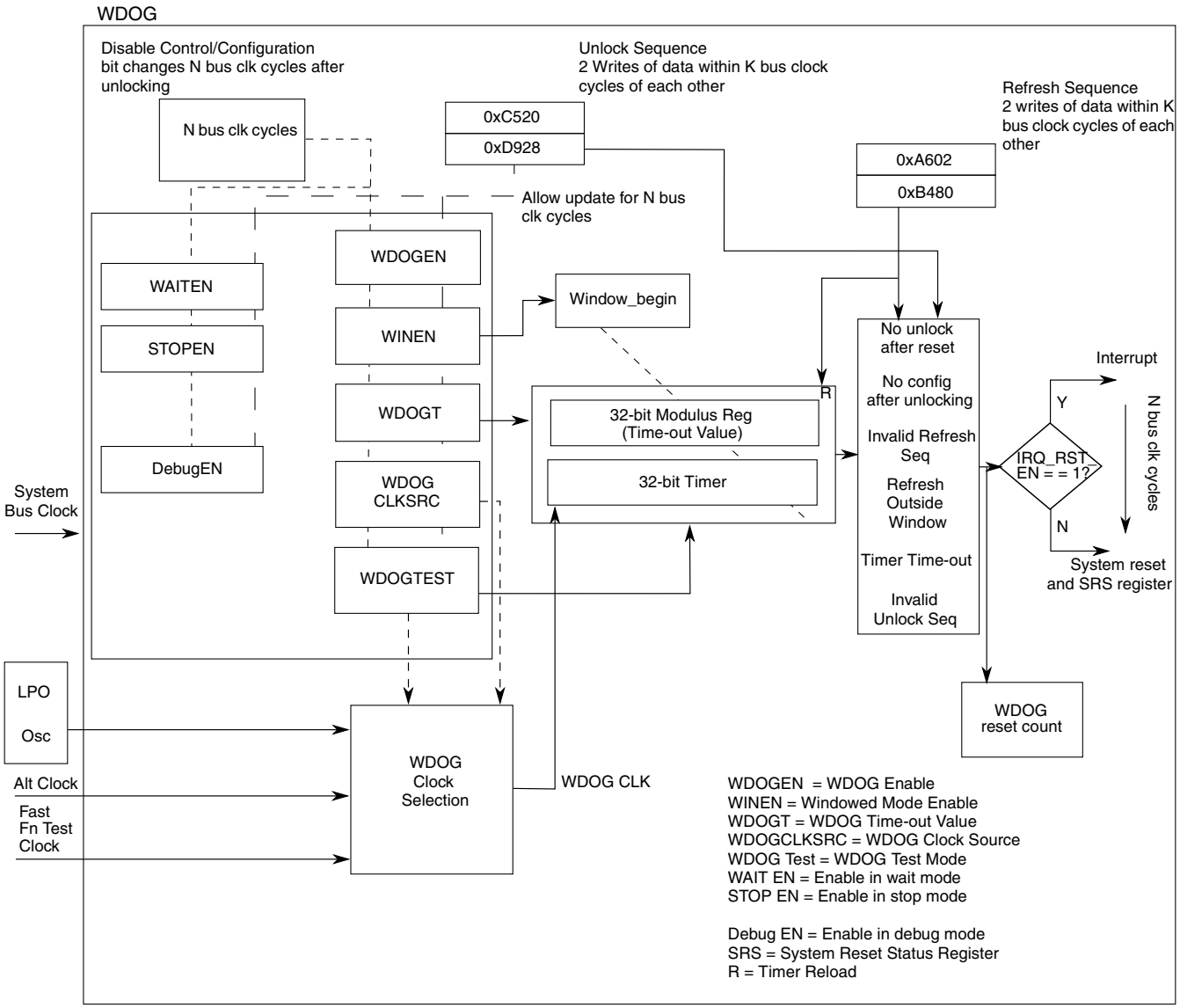


Figure 28-1. WDOG operation

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.

28.4.1 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.

28.4.2 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.

28.4.3 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.

28.4.4 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.

28.4.5 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.

28.4.6 Low-power modes of operation

The low-power modes of operation of the watchdog are described in the following table:

Table 28-3. Low-power modes of operation

Mode	Behavior
Wait	If the WDOG is enabled (WAIT_EN = 1), it can run on bus clock or low-power oscillator clock (CLK_SRC = x) to generate interrupt (IRQ_RST_EN=1) followed by a reset on time-out. After reset the WDOG reset counter increments by one.
Stop	Where the bus clock is gated, the WDOG can run only on low-power oscillator clock (CLK_SRC=0) if it is enabled in stop (STOP_EN=1). In this case, the WDOG runs to time-out twice, and then generates a reset from its backup circuitry. Therefore, if you program the watchdog to time-out after 100 ms and then enter such a stop mode, the reset will occur after 200 ms. Also, in this case, no interrupt will be generated irrespective of the value of IRQ_RST_EN bit. After WDOG reset, the WDOG reset counter will also not increment.
Power-Down	The watchdog is • static in LLS mode • powered off in VLLSx mode

28.4.7 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 debug 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 this 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 this mode, the WDOG timer restarts and the WDOG has to be unlocked and configured within WCT.

28.5 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.

28.5.1 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.

28.5.2 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:

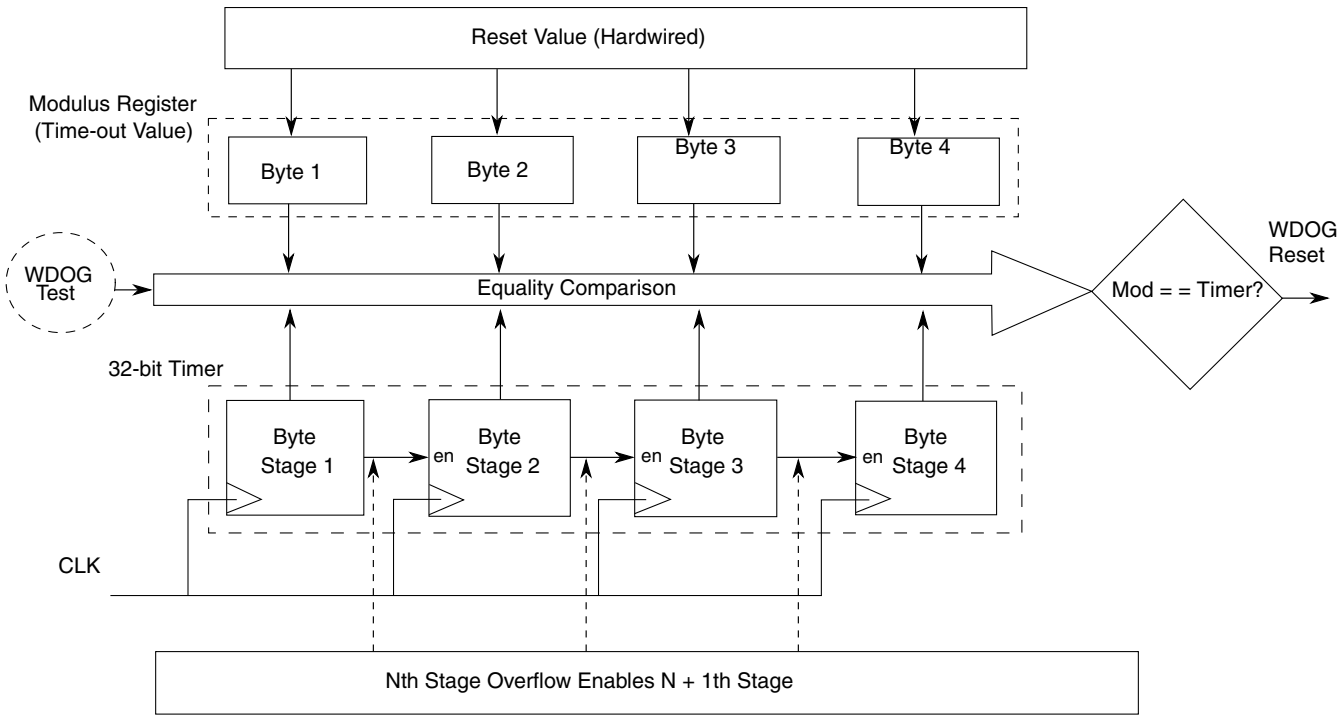


Figure 28-2. Watchdog timer byte splitting

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.

28.6 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.

28.7 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.

28.8 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	28.8.1/605
4005_2002	Watchdog Status and Control Register Low (WDOG_STCTRLL)	16	R/W	0001h	28.8.2/606
4005_2004	Watchdog Time-out Value Register High (WDOG_TOVALH)	16	R/W	004Ch	28.8.3/607
4005_2006	Watchdog Time-out Value Register Low (WDOG_TOVALL)	16	R/W	4B4Ch	28.8.4/607
4005_2008	Watchdog Window Register High (WDOG_WINH)	16	R/W	0000h	28.8.5/608
4005_200A	Watchdog Window Register Low (WDOG_WINL)	16	R/W	0010h	28.8.6/608
4005_200C	Watchdog Refresh register (WDOG_REFRESH)	16	R/W	B480h	28.8.7/609
4005_200E	Watchdog Unlock register (WDOG_UNLOCK)	16	R/W	D928h	28.8.8/609
4005_2010	Watchdog Timer Output Register High (WDOG_TMROUTH)	16	R/W	0000h	28.8.9/609
4005_2012	Watchdog Timer Output Register Low (WDOG_TMROUTL)	16	R/W	0000h	28.8.10/ 610
4005_2014	Watchdog Reset Count register (WDOG_RSTCNT)	16	R/W	0000h	28.8.11/ 610
4005_2016	Watchdog Prescaler register (WDOG_PRESC)	16	R/W	0400h	28.8.12/ 610

28.8.1 Watchdog Status and Control Register High (WDOG_STCTRLH)

Address: 4005_2000h base + 0h offset = 4005_2000h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	DISTESTWDOG	BYTESEL[1:0]		TESTSEL	TESTWDOG	0	Reserved	WAITEN	STOPEN	DBGEN	ALLOWUPDATE	WINEN	IRQRSTEN	CLKSRC	WDOGEN
Write																
Reset	0	0	0	0	0	0	0	1	1	1	0	1	0	0	1	1

WDOG_STCTRLH field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 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. 0 WDOG functional test mode is not disabled. 1 WDOG functional test mode is disabled permanently until reset.
13–12 BYTESEL[1:0]	This 2-bit field selects the byte to be tested when the watchdog is in the byte test mode. 00 Byte 0 selected 01 Byte 1 selected 10 Byte 2 selected 11 Byte 3 selected
11 TESTSEL	Effective only if TESTWDOG is set. Selects the test to be run on the watchdog timer. 0 Quick test. The timer runs in normal operation. You can load a small time-out value to do a quick test. 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 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 Reserved	This field is reserved.
7 WAITEN	Enables or disables WDOG in Wait mode. 0 WDOG is disabled in CPU Wait mode. 1 WDOG is enabled in CPU Wait mode.
6 STOPEN	Enables or disables WDOG in Stop mode.

Table continues on the next page...

WDOG_STCTRLH field descriptions (continued)

Field	Description
	0 WDOG is disabled in CPU Stop mode. 1 WDOG is enabled in CPU Stop mode.
5 DBGEN	Enables or disables WDOG in Debug mode. 0 WDOG is disabled in CPU Debug mode. 1 WDOG is enabled in CPU Debug mode.
4 ALLOWUPDATE	Enables updates to watchdog write-once registers, after the reset-triggered initial configuration window (WCT) closes, through unlock sequence. 0 No further updates allowed to WDOG write-once registers. 1 WDOG write-once registers can be unlocked for updating.
3 WINEN	Enables Windowing mode. 0 Windowing mode is disabled. 1 Windowing mode is enabled.
2 IRQRSTEN	Used to enable the debug breadcrumbs feature. A change in this bit is updated immediately, as opposed to updating after WCT. 0 WDOG time-out generates reset only. 1 WDOG time-out initially generates an interrupt. After WCT, it generates a reset.
1 CLKSRC	Selects clock source for the WDOG timer and other internal timing operations. 0 WDOG clock sourced from LPO . 1 WDOG clock sourced from alternate clock source.
0 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. 0 WDOG is disabled. 1 WDOG is enabled.

28.8.2 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 Write	INTFLG	Reserved						
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	1

WDOG_STCTRLL field descriptions

Field	Description
15 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.
Reserved	This field is reserved. **NOTE:** Do not modify this field value.

28.8.3 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 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
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.

28.8.4 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 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
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.

28.8.5 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 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
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.

28.8.6 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 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
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.

28.8.7 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 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
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.

28.8.8 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 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
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.

28.8.9 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 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
TIMEROUTHIGH	Shows the value of the upper 16 bits of the watchdog timer.

28.8.10 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 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
TIMEROUTLOW	Shows the value of the lower 16 bits of the watchdog timer.

28.8.11 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 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
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.

28.8.12 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–8 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.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

28.9 Watchdog operation with 8-bit access

28.9.1 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.

28.9.2 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 28-4. 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.

28.10 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 29
Multipurpose Clock Generator (MCG)

29.1 Chip-specific MCG information

29.1.1 MCG oscillator clock input options

The MCG has multiple oscillator input clock sources. Within the context of the MCG these are all referred to as the external reference clock and selection is determined by MCG_C7[OSCSEL] bitfield. The following table shows the chip-specific clock assignments for this bitfield.

Table 29-1. MCG oscillator reference options

MCG_C7[OSCSEL]	MCG defined selection	Chip clock
00	OSCCLK0 - System Oscillator	OSCCLK - System oscillator output. Derived from external crystal circuit or directly from EXTAL.
01	OSC2/RTC Oscillator	RTC 32kHz oscillator output. RTC clock is derived from external crystal circuit associated with RTC.
10	OSCCLK1 - Oscillator	IRC48MCLK. Derived from internal 48 MHz oscillator.
11	Reserved	—

See Clock Distribution for more details on these clocks.

29.2 Introduction

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 an FLL or PLL output clock, or a reference clock (internal or external) 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.

29.2.1 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.
- Control signals for the MCG external reference low power oscillator clock generators are provided:
    - HGO, RANGE, EREFS
- 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 be used as a source for the FLL only.
    - 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 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

This figure presents the block diagram of the MCG module.

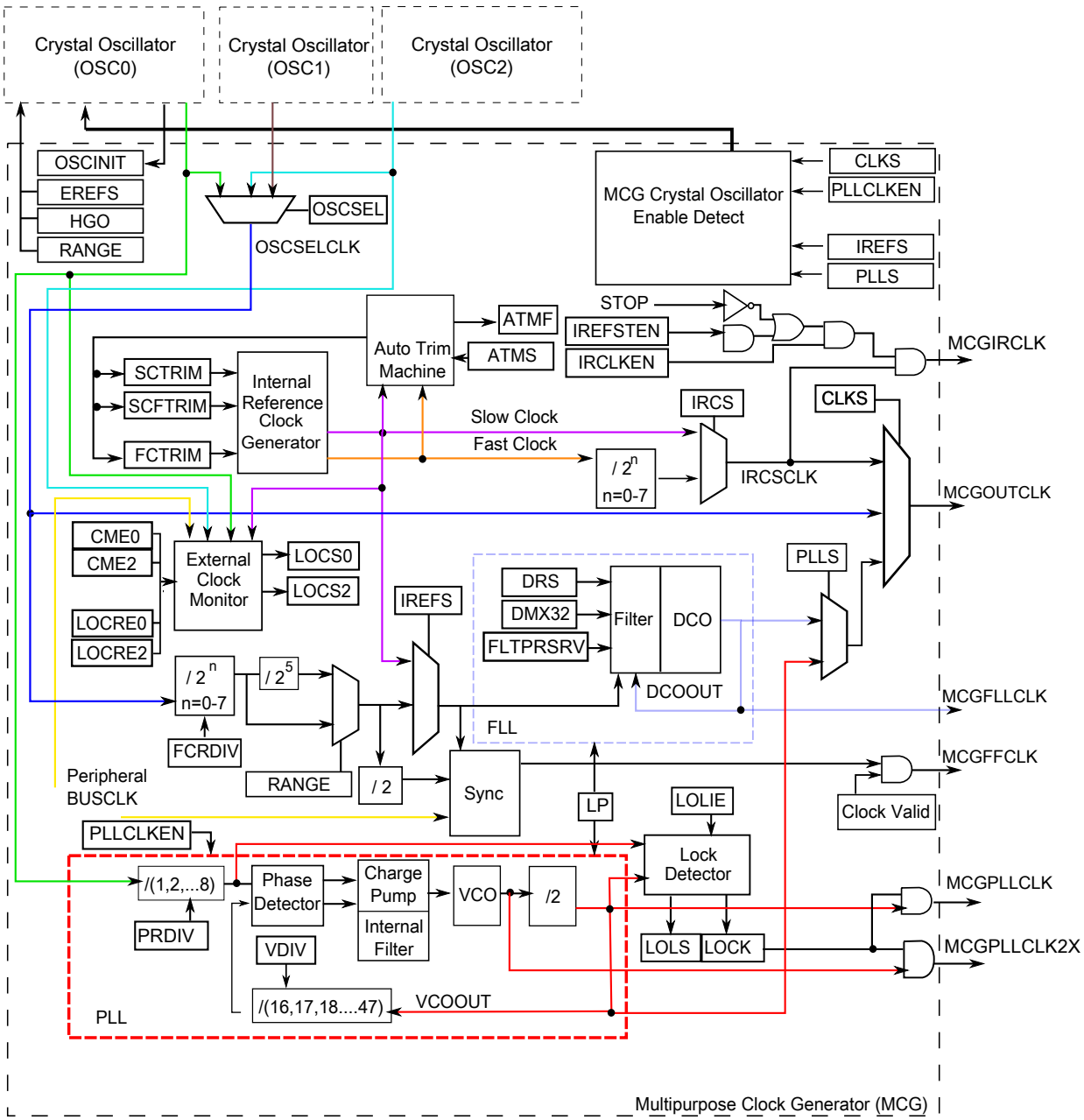


Figure 29-1. Multipurpose Clock Generator (MCG) block diagram

29.2.2 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.

29.3 External Signal Description

There are no MCG signals that connect off chip.

29.4 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	29.4.1/620
4006_4001	MCG Control 2 Register (MCG_C2)	8	R/W	80h	29.4.2/621
4006_4002	MCG Control 3 Register (MCG_C3)	8	R/W	Undefined	29.4.3/622
4006_4003	MCG Control 4 Register (MCG_C4)	8	R/W	Undefined	29.4.4/623
4006_4004	MCG Control 5 Register (MCG_C5)	8	R/W	00h	29.4.5/624
4006_4005	MCG Control 6 Register (MCG_C6)	8	R/W	00h	29.4.6/625
4006_4006	MCG Status Register (MCG_S)	8	R	10h	29.4.7/627
4006_4008	MCG Status and Control Register (MCG_SC)	8	R/W	02h	29.4.8/628
4006_400A	MCG Auto Trim Compare Value High Register (MCG_ATCVH)	8	R/W	00h	29.4.9/630
4006_400B	MCG Auto Trim Compare Value Low Register (MCG_ATCVL)	8	R/W	00h	29.4.10/ 630
4006_400C	MCG Control 7 Register (MCG_C7)	8	R/W	00h	29.4.11/ 630
4006_400D	MCG Control 8 Register (MCG_C8)	8	R/W	80h	29.4.12/ 631

29.4.1 MCG Control 1 Register (MCG_C1)

Address: 4006_4000h base + 0h offset = 4006_4000h

Bit	7	6	5	4	3	2	1	0
Read Write	CLKS		FRDIV			IREFS	IRCLKEN	IREFSTEN
Reset	0	0	0	0	0	1	0	0

MCG_C1 field descriptions

Field	Description
7–6 CLKS	Clock Source Select Selects the clock source for MCGOUTCLK . 00 Encoding 0 — Output of FLL or PLL is selected (depends on PLLS control bit). 01 Encoding 1 — Internal reference clock is selected. 10 Encoding 2 — External reference clock is selected. 11 Encoding 3 — Reserved.
5–3 FRDIV	FLL External Reference Divider 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). 000 If RANGE = 0 or OSCSEL=1 , Divide Factor is 1; for all other RANGE values, Divide Factor is 32. 001 If RANGE = 0 or OSCSEL=1 , Divide Factor is 2; for all other RANGE values, Divide Factor is 64. 010 If RANGE = 0 or OSCSEL=1 , Divide Factor is 4; for all other RANGE values, Divide Factor is 128. 011 If RANGE = 0 or OSCSEL=1 , Divide Factor is 8; for all other RANGE values, Divide Factor is 256. 100 If RANGE = 0 or OSCSEL=1 , Divide Factor is 16; for all other RANGE values, Divide Factor is 512. 101 If RANGE = 0 or OSCSEL=1 , Divide Factor is 32; for all other RANGE values, Divide Factor is 1024. 110 If RANGE = 0 or OSCSEL=1 , Divide Factor is 64; for all other RANGE values, Divide Factor is 1280 . 111 If RANGE = 0 or OSCSEL=1 , Divide Factor is 128; for all other RANGE values, Divide Factor is 1536 .
2 IREFS	Internal Reference Select Selects the reference clock source for the FLL. 0 External reference clock is selected. 1 The slow internal reference clock is selected.
1 IRCLKEN	Internal Reference Clock Enable Enables the internal reference clock for use as MCGIRCLK. 0 MCGIRCLK inactive. 1 MCGIRCLK active.
0 IREFSTEN	Internal Reference Stop Enable Controls whether or not the internal reference clock remains enabled when the MCG enters Stop mode.

Table continues on the next page...

MCG_C1 field descriptions (continued)

Field	Description
	0 Internal reference clock is disabled in Stop mode. 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.

29.4.2 MCG Control 2 Register (MCG_C2)

Address: 4006_4000h base + 1h offset = 4006_4001h

Bit	7	6	5	4	3	2	1	0
Read Write	LOCRE0	FCFTRIM	RANGE		HGO	EREFS	LP	IRCS
Reset	1	0	0	0	0	0	0	0

MCG_C2 field descriptions

Field	Description
7 LOCRE0	Loss of Clock Reset Enable 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. 0 Interrupt request is generated on a loss of OSC0 external reference clock. 1 Generate a reset request on a loss of OSC0 external reference clock.
6 FCFTRIM	Fast Internal Reference Clock Fine Trim FCFTRIM controls the smallest adjustment of the fast internal reference clock frequency. Setting FCFTRIM increases the period and clearing FCFTRIM decreases the period by the smallest amount possible. If an FCFTRIM 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.
5–4 RANGE	Frequency Range Select 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. 00 Encoding 0 — Low frequency range selected for the crystal oscillator . 01 Encoding 1 — High frequency range selected for the crystal oscillator . 1X Encoding 2 — Very high frequency range selected for the crystal oscillator .
3 HGO	High Gain Oscillator Select Controls the crystal oscillator mode of operation. See the Oscillator (OSC) chapter for more details. 0 Configure crystal oscillator for low-power operation. 1 Configure crystal oscillator for high-gain operation.
2 EREFS	External Reference Select Selects the source for the external reference clock. See the Oscillator (OSC) chapter for more details. 0 External reference clock requested. 1 Oscillator requested.

Table continues on the next page...

MCG_C2 field descriptions (continued)

Field	Description
1 LP	Low Power Select 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. 0 FLL or PLL is not disabled in bypass modes. 1 FLL or PLL is disabled in bypass modes (lower power)
0 IRCS	Internal Reference Clock Select Selects between the fast or slow internal reference clock source. 0 Slow internal reference clock selected. 1 Fast internal reference clock selected.

29.4.3 MCG Control 3 Register (MCG_C3)

Address: 4006_4000h base + 2h offset = 4006_4002h

Bit	7	6	5	4	3	2	1	0
Read Write	SCTRIM							
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

MCG_C3 field descriptions

Field	Description
SCTRIM	Slow Internal Reference Clock Trim Setting 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. 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. 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.

29.4.4 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 Write	DMX32	DRST_DRS			FCTRIM			SCFTRIM
Reset	0	0	0	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.
- x = Undefined at reset.

MCG_C4 field descriptions

Field	Description
7 DMX32	DCO Maximum Frequency with 32.768 kHz Reference The DMX32 bit controls whether the DCO frequency range is narrowed to its maximum frequency with a 32.768 kHz reference. The following table identifies settings for the DCO frequency range. **NOTE:** The system clocks derived from this source should not exceed their specified maximums.

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

0 DCO has a default range of 25%.
1 DCO is fine-tuned for maximum frequency with 32.768 kHz reference.

6–5 DRST_DRS	DCO Range Select 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. 00 Encoding 0 — Low range (reset default).

Table continues on the next page...

MCG_C4 field descriptions (continued)

Field	Description
	01 Encoding 1 — Mid range. 10 Encoding 2 — Mid-high range. 11 Encoding 3 — High range.
4–1 FCTRIM	Fast Internal Reference Clock Trim Setting 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. 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 SCFTRIM	Slow Internal Reference Clock Fine Trim 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. 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 .

29.4.5 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	PLLCLKEN	PLLSTEN	0		PRDIV		
Write								
Reset	0	0	0	0	0	0	0	0

MCG_C5 field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 PLLCLKEN	PLL Clock Enable Enables PLL independent of PLLS and enables the PLL clock for use as MCGPLLCLK. (PRDIV needs to be programmed to the correct divider to generate a PLL reference clock in a valid reference range prior to setting the PLLCLKEN bit). Setting PLLCLKEN will enable the external oscillator if not already enabled. Whenever the PLL is being enabled by means of the PLLCLKEN 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. 0 MCGPLLCLK is inactive. 1 MCGPLLCLK is active.
5 PLLSTEN	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=1). All other power modes, PLLSTEN bit has no affect and does not enable the PLL Clock to run if it is written to 1. 0 MCGPLLCLK and MCGPLLCLK2X are disabled in any of the Stop modes. 1 MCGPLLCLK and MCGPLLCLK2X are enabled if system is in Normal Stop mode.
4–3 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
PRDIV	PLL External Reference Divider Selects the amount to divide down the external reference clock for the PLL0. The resulting frequency must be in the range of 8 MHz to 16 MHz. After the PLL0 is enabled (by setting either PLLCLKEN0 or PLLS), the PRDIV0 value must not be changed when LOCK0 is zero.

Table 29-2. PLL External Reference Divide Factor

PRDIV	Divide Factor
000	1
001	2
010	3
011	4
100	5
101	6
110	7
111	8

29.4.6 MCG Control 6 Register (MCG_C6)

Address: 4006_4000h base + 5h offset = 4006_4005h

Bit	7	6	5	4	3	2	1	0
Read Write	LOLIE0	PLLS	CME0	VDIV				
Reset	0	0	0	0	0	0	0	0

MCG_C6 field descriptions

Field	Description
7 LOLIE0	Loss of Lock Interrrupt Enable Determines if an interrupt request is made following a loss of lock indication. This bit only has an effect when LOLS 0 is set.

Table continues on the next page...

MCG_C6 field descriptions (continued)

Field	Description
	0 No interrupt request is generated on loss of lock. 1 Generate an interrupt request on loss of lock.
6 PLLS	PLL Select 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. 0 FLL is selected. 1 PLL is selected (PRDIV 0 need to be programmed to the correct divider to generate a PLL reference clock in the range of 8–16 MHz prior to setting the PLLS bit).
5 CME0	Clock Monitor Enable 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. 0 External clock monitor is disabled for OSC0. 1 External clock monitor is enabled for OSC0.
VDIV	VCO Divider Selects the amount to divide the VCO output of the PLL. The VDIV bits establish the multiplication factor (M) applied to the reference clock frequency. After the PLL is enabled (by setting either PLLCLKEN or PLLS), the VDIV value must not be changed when LOCK is zero.

Table 29-3. PLL VCO Divide Factor

VDIV	Multiply Factor		VDIV	Multiply Factor		VDIV	Multiply Factor		VDIV	Multiply Factor
00000	16		01000	24		10000	32		11000	40
00001	17		01001	25		10001	33		11001	41
00010	18		01010	26		10010	34		11010	42
00011	19		01011	27		10011	35		11011	43
00100	20		01100	28		10100	36		11100	44
00101	21		01101	29		10101	37		11101	45
00110	22		01110	30		10110	38		11110	46
00111	23		01111	31		10111	39		11111	47

29.4.7 MCG Status Register (MCG_S)

Address: 4006_4000h base + 6h offset = 4006_4006h

Bit	7	6	5	4	3	2	1	0
Read	LOLS0	LOCK0	PLLST	IREFST	CLKST		OSCINIT0	IRCST
Write								
Reset	0	0	0	1	0	0	0	0

MCG_S field descriptions

Field	Description
7 LOLS0	Loss of Lock Status 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. 0 PLL has not lost lock since LOLS 0 was last cleared. 1 PLL has lost lock since LOLS 0 was last cleared.
6 LOCK0	Lock Status This bit indicates whether the PLL has acquired lock. Lock detection is disabled when not operating in either PBE or PEE mode unless PLLCLKEN=1 and the MCG is not configured in BLPI or BLPE mode. While the PLL clock is locking to the desired frequency, MCGPLLCLK and MCGPLLCLK2X will be gated off until the LOCK bit gets asserted. If the lock status bit is set, changing the value of the PRDIV[2:0] bits in the C5 register or the VDIV[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 LOCK bit to clear until 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 LOCK bit is cleared, the MCGPLLCLK and MCGPLLCLK2X will be gated off until the LOCK bit is asserted again. 0 PLL is currently unlocked. 1 PLL is currently locked.
5 PLLST	PLL Select Status 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. 0 Source of PLLS clock is FLL clock. 1 Source of PLLS clock is PLL output clock.
4 IREFST	Internal Reference Status 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. 0 Source of FLL reference clock is the external reference clock. 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 CLKST	Clock Mode Status 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. 00 Encoding 0 — Output of the FLL is selected (reset default). 01 Encoding 1 — Internal reference clock is selected. 10 Encoding 2 — External reference clock is selected. 11 Encoding 3 — Output of the PLL is selected.
1 OSCINIT0	OSC Initialization 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 IRCST	Internal Reference Clock Status 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 . 0 Source of internal reference clock is the slow clock (32 kHz IRC). 1 Source of internal reference clock is the fast clock (4 MHz IRC).

29.4.8 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			w1c					w1c
Reset	0	0	0	0	0	0	1	0

MCG_SC field descriptions

Field	Description
7 ATME	Automatic Trim Machine Enable Enables the Auto Trim Machine to start automatically trimming the selected Internal Reference Clock. **NOTE:** ATME deasserts after the Auto Trim Machine has completed trimming all trim bits of the IRCS clock selected by the ATMS bit. Writing to C1, C3, C4, and SC registers or entering Stop mode aborts the auto trim operation and clears this bit. 0 Auto Trim Machine disabled. 1 Auto Trim Machine enabled.

Table continues on the next page...

MCG_SC field descriptions (continued)

Field	Description
6 ATMS	Automatic Trim Machine Select Selects the IRCS clock for Auto Trim Test. 0 32 kHz Internal Reference Clock selected. 1 4 MHz Internal Reference Clock selected.
5 ATMF	Automatic Trim Machine Fail Flag 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. 0 Automatic Trim Machine completed normally. 1 Automatic Trim Machine failed.
4 FLTPRSRV	FLL Filter Preserve Enable 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.) 0 FLL filter and FLL frequency will reset on changes to currect clock mode. 1 Fll filter and FLL frequency retain their previous values during new clock mode change.
3–1 FCRDIV	Fast Clock Internal Reference Divider 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). 000 Divide Factor is 1 001 Divide Factor is 2. 010 Divide Factor is 4. 011 Divide Factor is 8. 100 Divide Factor is 16 101 Divide Factor is 32 110 Divide Factor is 64 111 Divide Factor is 128.
0 LOCS0	OSC0 Loss of Clock Status 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. 0 Loss of OSC0 has not occurred. 1 Loss of OSC0 has occurred.

29.4.9 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 Write	ATCVH							
Reset	0	0	0	0	0	0	0	0

MCG_ATCVH field descriptions

Field	Description
ATCVH	ATM Compare Value High Values are used by Auto Trim Machine to compare and adjust Internal Reference trim values during ATM SAR conversion.

29.4.10 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 Write	ATCVL							
Reset	0	0	0	0	0	0	0	0

MCG_ATCVL field descriptions

Field	Description
ATCVL	ATM Compare Value Low Values are used by Auto Trim Machine to compare and adjust Internal Reference trim values during ATM SAR conversion.

29.4.11 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–2 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
OSCSEL	MCG OSC Clock Select Selects the MCG FLL external reference clock **NOTE:** The OSCSEL field can't be changed during MCG modes (like PBE), when external clock is serving as the clock source for MCG. 00 Selects Oscillator (OSCCLK0). 01 Selects 32 kHz RTC Oscillator. 10 Selects Oscillator (OSCCLK1). 11 RESERVED

29.4.12 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								w1c
Reset	1	0	0	0	0	0	0	0

MCG_C8 field descriptions

Field	Description
7 LOCRE1	Loss of Clock Reset Enable 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. 0 Interrupt request is generated on a loss of RTC external reference clock. 1 Generate a reset request on a loss of RTC external reference clock
6 LOLRE	PLL Loss of Lock Reset Enable Determines if an interrupt or a reset request is made following a PLL loss of lock. 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. 1 Generate a reset request on a PLL loss of lock indication.
5 CME1	Clock Monitor Enable1 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

Table continues on the next page...

MCG_C8 field descriptions (continued)

Field	Description
	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. 0 External clock monitor is disabled for RTC clock. 1 External clock monitor is enabled for RTC clock.
4–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 LOCS1	RTC Loss of Clock Status This bit indicates when a loss of clock has occurred. This bit is cleared by writing a logic 1 to it when set. 0 Loss of RTC has not occur. 1 Loss of RTC has occur

29.5 Functional description

29.5.1 MCG mode state diagram

The nine states of the MCG are shown in the following figure and are described in Table 29-4. The arrows indicate the permitted MCG mode transitions.

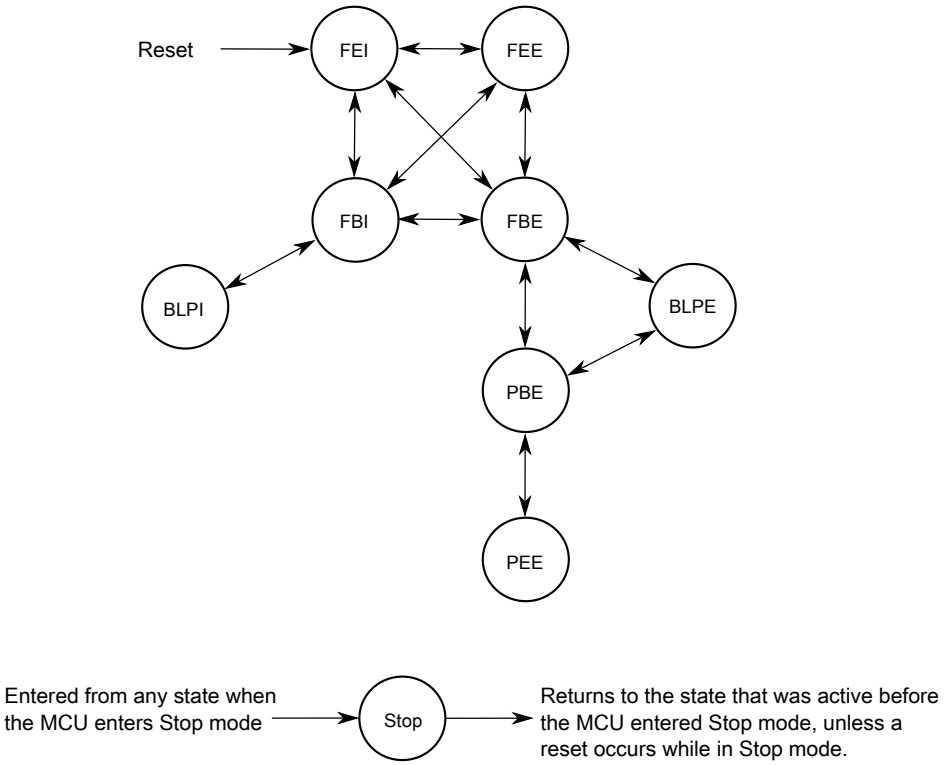


Figure 29-2. MCG mode state diagram

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.

29.5.1.1 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 29-2.

Table 29-4. 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: • 00 is written to C1[CLKS]. • 1 is written to C1[IREFS]. • 0 is written to C6[PLLS]. 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 .
FLL Engaged External (FEE)	FLL engaged external (FEE) mode is entered when all the following conditions occur: • 00 is written to C1[CLKS]. • 0 is written to C1[IREFS]. • C1[FRDIV] must be written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz • 0 is written to C6[PLLS]. 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[RANGE]. 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: • 01 is written to C1[CLKS]. • 1 is written to C1[IREFS]. • 0 is written to C6[PLLS] • 0 is written to C2[LP]. 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: • 10 is written to C1[CLKS]. • 0 is written to C1[IREFS]. • C1[FRDIV] must be written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz. • 0 is written to C6[PLLS]. • 0 is written to C2[LP].

Table continues on the next page...

Table 29-4. MCG modes of operation (continued)

Mode	Description
	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 FBE mode, the PLL is disabled in a low-power state unless C5[PLLCLKEN] is set .
PLL Engaged External (PEE)	PLL Engaged External (PEE) mode is entered when all the following conditions occur: • 00 is written to C1[CLKS]. • 0 is written to C1[IREFS]. • 1 is written to C6[PLLS]. 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: • 10 is written to C1[CLKS]. • 0 is written to C1[IREFS]. • 1 is written to C6[PLLS]. • 0 is written to C2[LP]. 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: • 01 is written to C1[CLKS]. • 1 is written to C1[IREFS]. • 0 is written to C6[PLLS]. • 1 is written to C2[LP]. In BLPI mode, MCGOUTCLK is derived from the internal reference clock. The FLL is disabled and PLL is disabled even if C5[PLLCLKEN] is set to 1.
Bypassed Low Power External (BLPE) [1]	Bypassed Low Power External (BLPE) mode is entered when all the following conditions occur: • 10 is written to C1[CLKS]. • 0 is written to C1[IREFS]. • 1 is written to C2[LP]. In BLPE mode, MCGOUTCLK is derived from the OSCSEL external reference clock. The FLL is disabled and PLL is disabled even if the C5[PLLCLKEN] is set to 1.

Table continues on the next page...

Table 29-4. 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: MCGPLLCLK is active in Normal Stop mode when PLLSTEN=1 MCGIRCLK is active in Normal Stop mode when all the following conditions become true: • C1[IRCLKEN] = 1 • C1[IREFSTEN] = 1 **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) **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[LOCK] bit will be cleared without setting S[LOLS]. • When entering Normal Stop mode from PEE mode and if C5[PLLSTEN]=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[LOCK] bit will clear without setting S[LOLS]. If C5[PLLSTEN]=1, the S[LOCK] bit will not get cleared and on exit the MCG will continue to run in PEE mode.

1. **Caution:** 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.

29.5.1.2 MCG mode switching

C1[IREFS] can be changed at any time, but the actual switch to the newly selected reference clocks is shown by S[IREFST]. When switching between engaged internal and engaged external modes, the FLL will begin locking again after the switch is completed.

C1[CLKS] can also be changed at any time, but the actual switch to the newly selected clock is shown by S[CLKST]. 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 C4[DRST_DRS] write bits are changed while in FLL engaged internal (FEI) or FLL engaged external (FEE) mode, the MCGOUTCLK switches to the new selected DCO

range within three clocks of the selected DCO clock. After switching to the new DCO (indicated by the updated C4[DRST_DRS] read bits), the FLL remains unlocked for several reference cycles. The FLL lock time is provided in the device data sheet as $t_{fll\_acquire}$.

29.5.2 Low-power bit usage

C2[LP] is provided to allow the FLL or PLL to be disabled and thus conserve power when these systems are not being used. C4[DRST_DRS] can not be written while C2[LP] 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 0 to C2[LP].

29.5.3 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.

29.5.3.1 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 C4[FCTRIM]:C2[FCFTRIM] 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]:C2[FCFTRIM] (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.

29.5.4 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.

On detecting 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.

29.5.5 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.

29.5.6 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.

29.5.7 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.

$$\text{Expected Count Value} = (\text{Fe}/\text{Fr}) * 21 * (128)$$

29.6 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.

29.6.1 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.

29.6.1.1 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 29-2). 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 C1[IREFS] bit to switch to the external reference, and leave C1[CLKS] at 2'b00 so that the output of the FLL is selected as the system clock source.

- If entering FBE, clear C1[IREFS] to switch to the external reference and change C1[CLKS] 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 C1[IRCLKEN]. 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[EREFS] 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 S[IREFST] is cleared before moving on.
   - If in FBE mode, check to make sure S[IREFST] 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] to 1 for a IRCS clock derived from the 4 MHz IRC source.

29.6.2 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.

29.6.3 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[EREFS], 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 16 MHz crystal, C5[PRDIV0] must be set to 3'b000 (divide-by-1) or 3'b001 (divide-by-2) to divide the external reference down to the required frequency between 8 and 16 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 29-5. MCGOUTCLK Frequency Calculation Options

Clock Mode	$f_{MCGOUTCLK}$[1]	Note
FEI (FLL engaged internal)	$f_{int} \times F$	Typical $f_{MCGOUTCLK}$ = 21 MHz immediately after reset.
FEE (FLL engaged external)	$(f_{ext} / FLL\_R) \times F$	f_{ext} / FLL_R must be in the range of 31.25 kHz to 39.0625 kHz
FBE (FLL bypassed external)	OSCCLK	OSCCLK / FLL_R must be in the range of 31.25 kHz to 39.0625 kHz

Table continues on the next page...

Table 29-5. MCGOUTCLK Frequency Calculation Options (continued)

Clock Mode	$f_{MCGOUTCLK}$[1]	Note
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 8 – 16 MHz
PBE (PLL bypassed external)	OSCCLK	OSCCLK / PLL_R must be in the range of 8 – 16 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, F is the FLL factor selected by C4[DRST_DRS] and C4[DMX32] bits , PLL_R is the reference divider selected by C5[PRDIV0] bits, and M is the multiplier selected by C6[VDIV0] bits.

This section will include several mode switching examples, using an 16 MHz external crystal. If using an external clock source less than 8 MHz, the MCG must not be configured for any of the PLL modes (PEE and PBE).

29.6.3.1 Example 1: Moving from FEI to PEE Mode with OSC0 as the source for the external crystal clock: External Crystal = 16 MHz, MCGOUTCLK frequency = 120 MHz

In this example, the MCG will move through the proper operational modes from FEI to PEE to achieve 120 MHz MCGOUTCLK frequency from 16 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[RANGE] set to 2'b10 because the frequency of 16 MHz is within the high frequency range.
      - C2[HGO] set to 1 to configure the crystal oscillator for high gain operation.
      - C2[EREFS] set to 1, because a crystal is being used.
   b. C1 = 0xA0
      - C1[CLKS] set to 2'b10 to select external reference clock as system clock source

- C1[FRDIV] set to 3'b100, or divide-by-512 because 8 MHz / 512 = 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[PRDIV] set to 3'b001, or divide-by-2 resulting in a pll reference frequency of 16MHz/2 = 8 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 = 0x4E

- 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 16 MHz/ 2 = 8 MHz. In BLPE mode, changing the C6[PLLS] bit only prepares the MCG for PLL usage in PBE mode.
- C6[VDIV] set to 5'b01110, or multiply-by-30 because 8 MHz reference * 30 = 240 MHz. This is the MCGPLL0CLK2X frequency, which is the frequency of the VCO. This is divided by 2 to achieve the MCGPLL0CLK frequency of 120 MHz which is later used as the MCGOUTCLK frequency. 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 = 0x20

- 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 PRDIV of divide-by-2, and C6[VDIV] of multiply-by-30, MCGOUTCLK = [(16 MHz / 2) * 30] / 2 = 120 MHz.

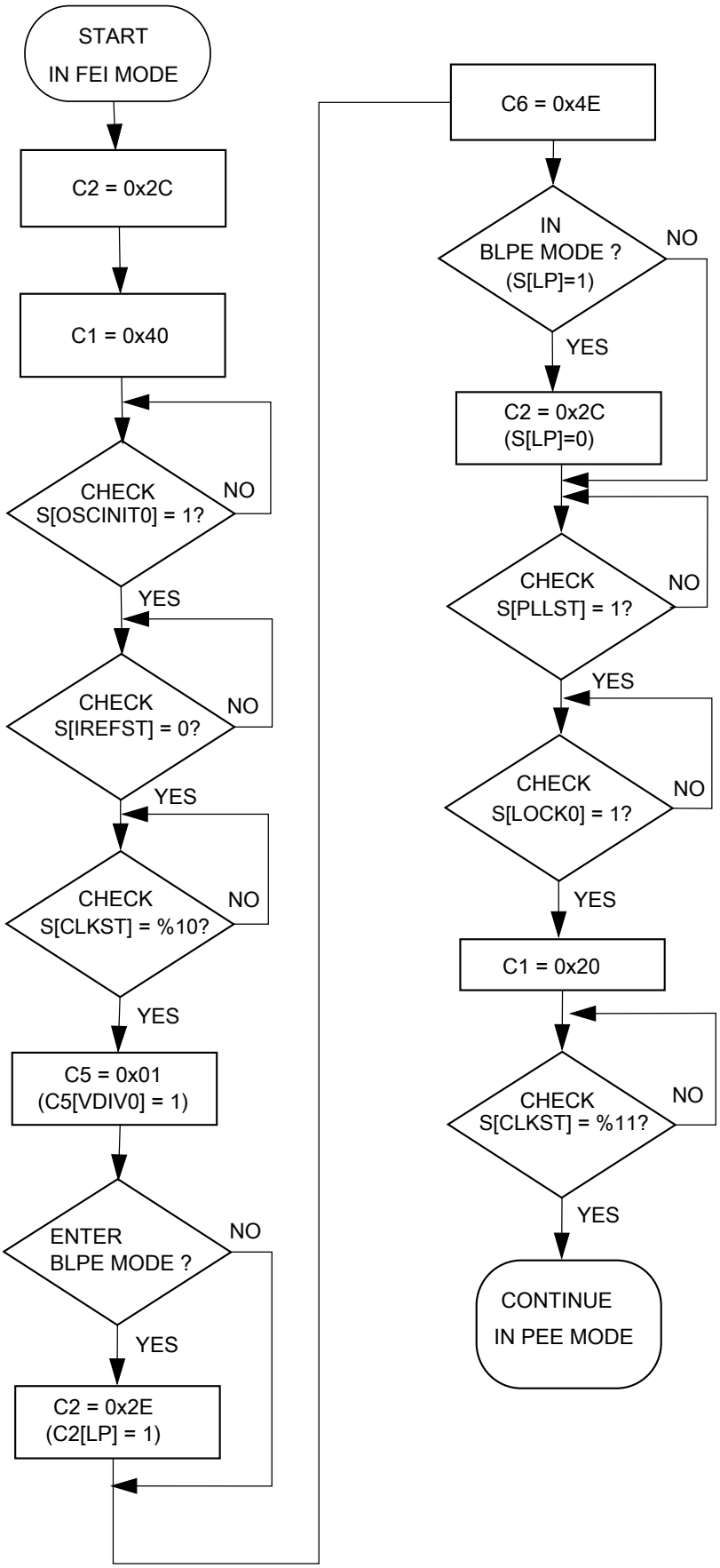


Figure 29-3. Flowchart of FEI to PEE mode transition using an 16 MHz crystal

29.6.3.2 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 16 MHz crystal configured for a 120 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 = 0x40
      - 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'b100, the FLL divider is set to 512, resulting in a reference frequency of 16 MHz / 512 = 31.25 kHz. If C1[FRDIV] was not previously set to 3'b100 (necessary to achieve required 31.25–39.06 kHz FLL reference frequency with an 16 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[VDIV] 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 = 0x64
      - 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 = 0x22

- C2[LP] is 1
- C2[RANGE], C2[HGO], C2[EREFS], 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.

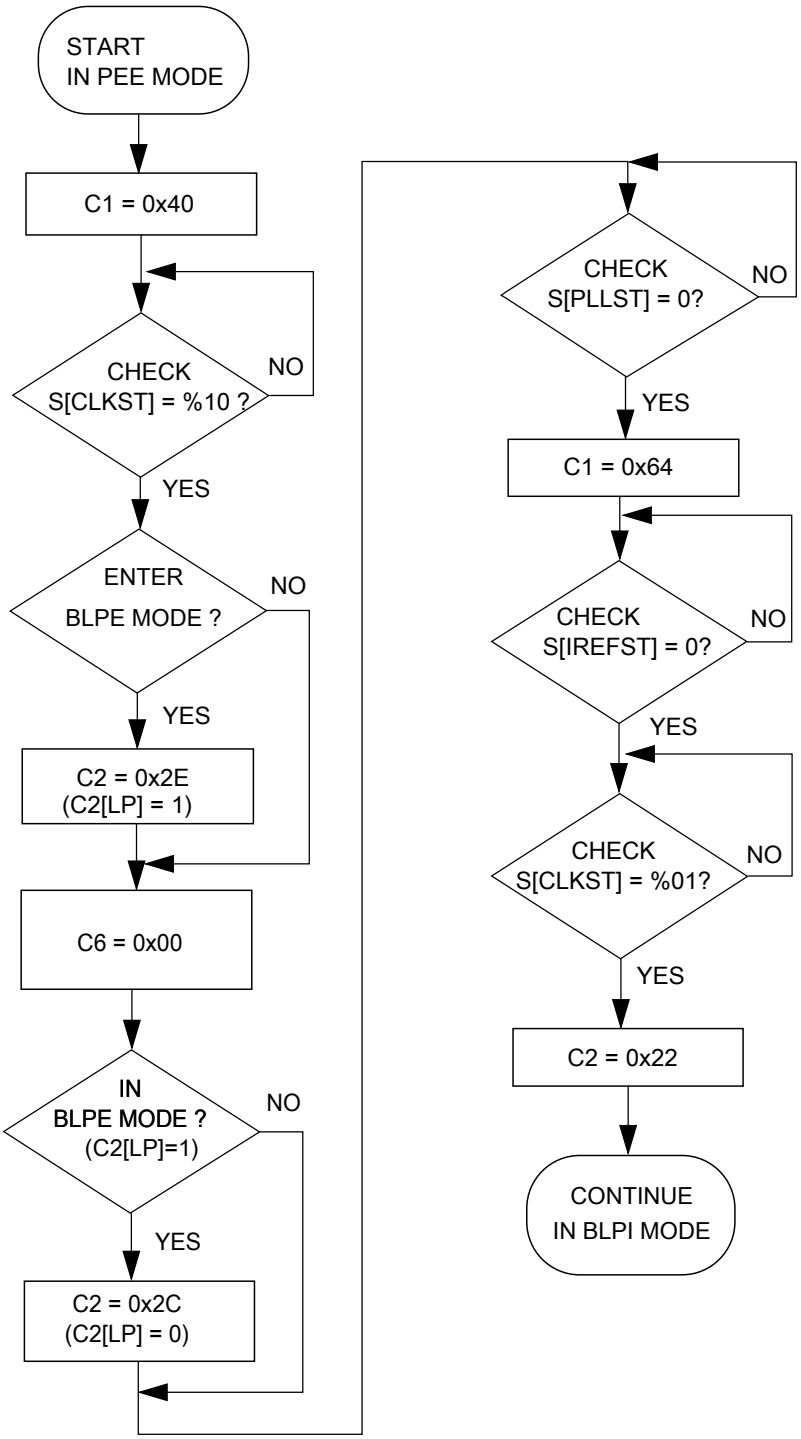


Figure 29-4. Flowchart of PEE to BLPI mode transition using an 16 MHz crystal

Chapter 30
Oscillator (OSC)

30.1 Chip-specific OSC information

30.1.1 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.

30.2 Introduction

The OSC module is a crystal oscillator. The module, in conjunction with an external crystal or resonator, generates a reference clock for the MCU.

30.3 Features and Modes

Key features of the module are listed here.

- 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.

30.4 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 . The OSCCLK can only work in run mode. OSCERCLK and 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 details for the external reference clock source in this MCU.

The figure found here shows the block diagram of the OSC module.

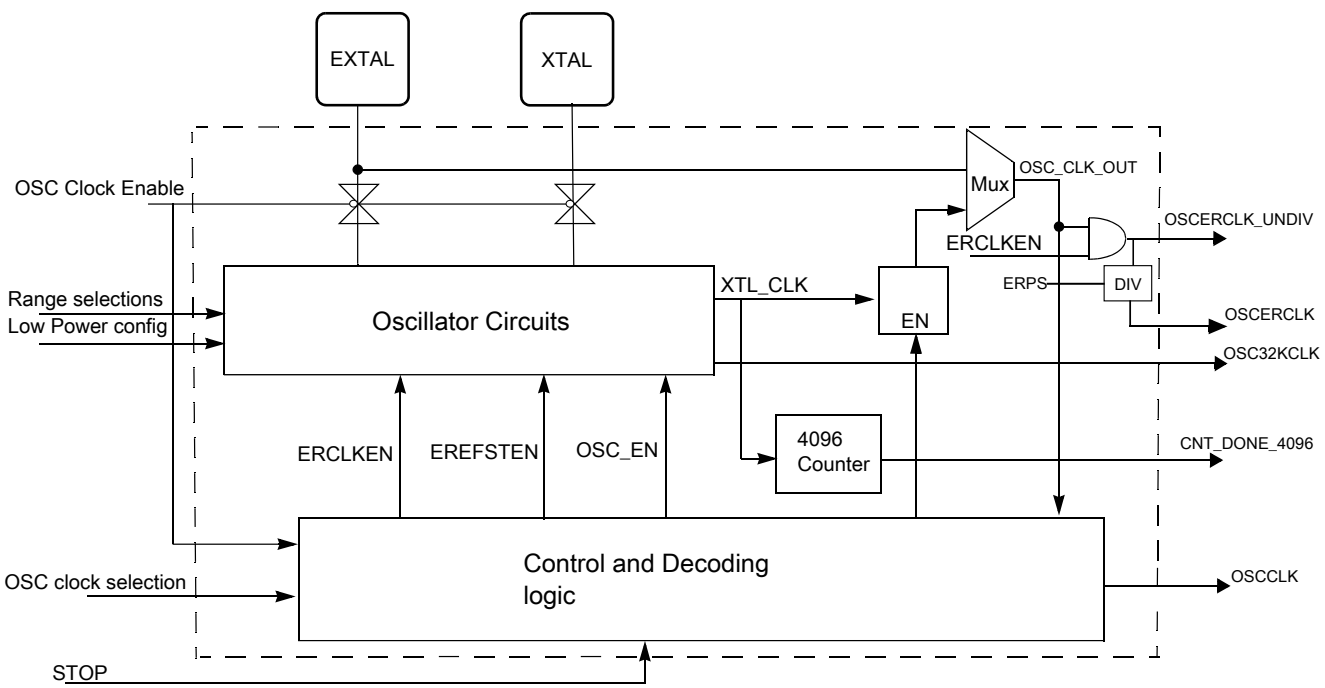


Figure 30-1. OSC Module Block Diagram

30.5 OSC Signal Descriptions

The table found here shows the user-accessible signals available for the OSC module.

Refer to signal multiplexing information for this MCU for more details.

Table 30-1. OSC Signal Descriptions

Signal	Description	I/O
EXTAL	External clock/Oscillator input	I
XTAL	Oscillator output	O

30.6 External Crystal / Resonator Connections

The connections for a crystal/resonator frequency reference are shown in the figures found here.

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 30-2. External Caystal/Resonator Connections

Oscillator Mode	Connections
Low-frequency (32 kHz), low-power	Connection 1[1]
Low-frequency (32 kHz), high-gain	Connection 2/Connection 3[2]
High-frequency (3~32 MHz), low-power	Connection 3[1]
High-frequency (3~32 MHz), high-gain	Connection 3

1. 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.
2. When the load capacitors (Cx, Cy) are greater than 30 pF, use Connection 3.

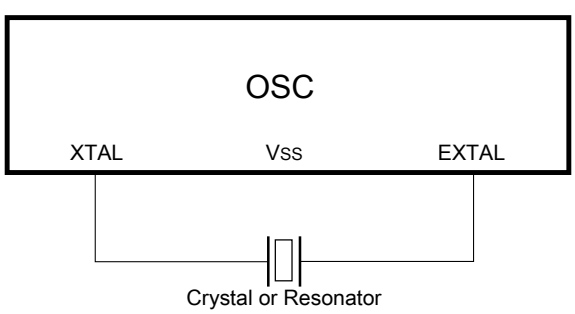


Figure 30-2. Crystal/Ceramic Resonator Connections - Connection 1

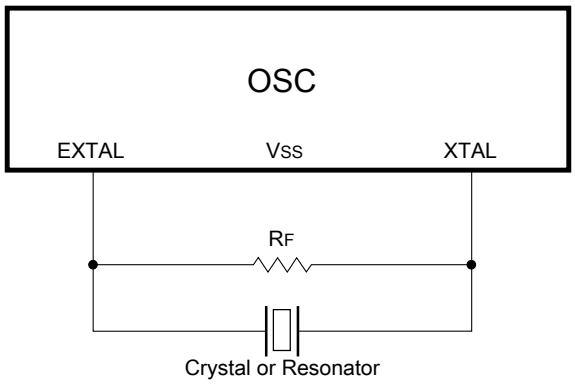


Figure 30-3. Crystal/Ceramic Resonator Connections - Connection 2

NOTE

Connection 1 and Connection 2 should use internal capacitors as the load of the oscillator by configuring the CR[SCxP] bits.

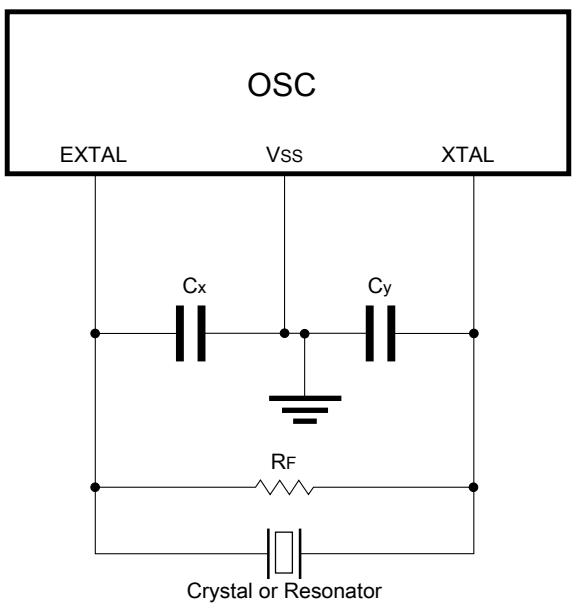


Figure 30-4. Crystal/Ceramic Resonator Connections - Connection 3

30.7 External Clock Connections

In external clock mode, the pins can be connected as shown in the figure found here.

NOTE

XTAL can be used as a GPIO when the GPIO alternate function is configured for it.

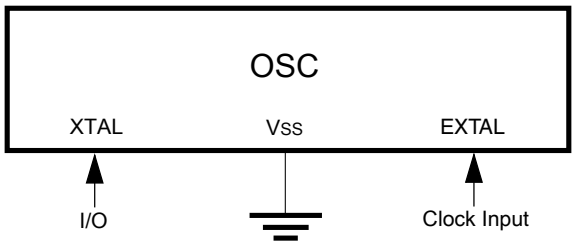


Figure 30-5. External Clock Connections

30.8 Memory Map/Register Definitions

Some oscillator module register bits are typically incorporated into other peripherals such as MCG or SIM.

30.8.1 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	30.8.1.1/ 655
4006_5002	OSC_DIV (OSC_OSC_DIV)	8	R/W	00h	30.8.1.2/ 657

30.8.1.1 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 ERCLKEN	External Reference Enable Enables external reference clock (OSCERCLK) . 0 External reference clock is inactive. 1 External reference clock is enabled.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 EREFSTEN	External Reference Stop Enable Controls whether or not the external reference clock (OSCERCLK) remains enabled when MCU enters Stop mode. 0 External reference clock is disabled in Stop mode. 1 External reference clock stays enabled in Stop mode if ERCLKEN is set before entering Stop mode.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 SC2P	Oscillator 2 pF Capacitor Load Configure Configures the oscillator load. 0 Disable the selection. 1 Add 2 pF capacitor to the oscillator load.
2 SC4P	Oscillator 4 pF Capacitor Load Configure Configures the oscillator load. 0 Disable the selection. 1 Add 4 pF capacitor to the oscillator load.
1 SC8P	Oscillator 8 pF Capacitor Load Configure Configures the oscillator load. 0 Disable the selection. 1 Add 8 pF capacitor to the oscillator load.
0 SC16P	Oscillator 16 pF Capacitor Load Configure Configures the oscillator load. 0 Disable the selection. 1 Add 16 pF capacitor to the oscillator load.

30.8.1.2 OSC_DIV (OSC_OSC_DIV)

OSC Clock divider register.

Address: 4006_5000h base + 2h offset = 4006_5002h

Bit	7	6	5	4	3	2	1	0
Read	ERPS		0	0	0	0	0	0
Write								
Reset	0	0	0	0	0	0	0	0

OSC_OSC_DIV field descriptions

Field	Description
7–6 ERPS	ERCLK prescaler. These two bits are used to divide the ERCLK output. The un-divided ERCLK output is not affected by these two bits. 00 The divisor ratio is 1. 01 The divisor ratio is 2. 10 The divisor ratio is 4. 11 The divisor ratio is 8.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

30.9 Functional Description

Functional details of the module can be found here.

30.9.1 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.

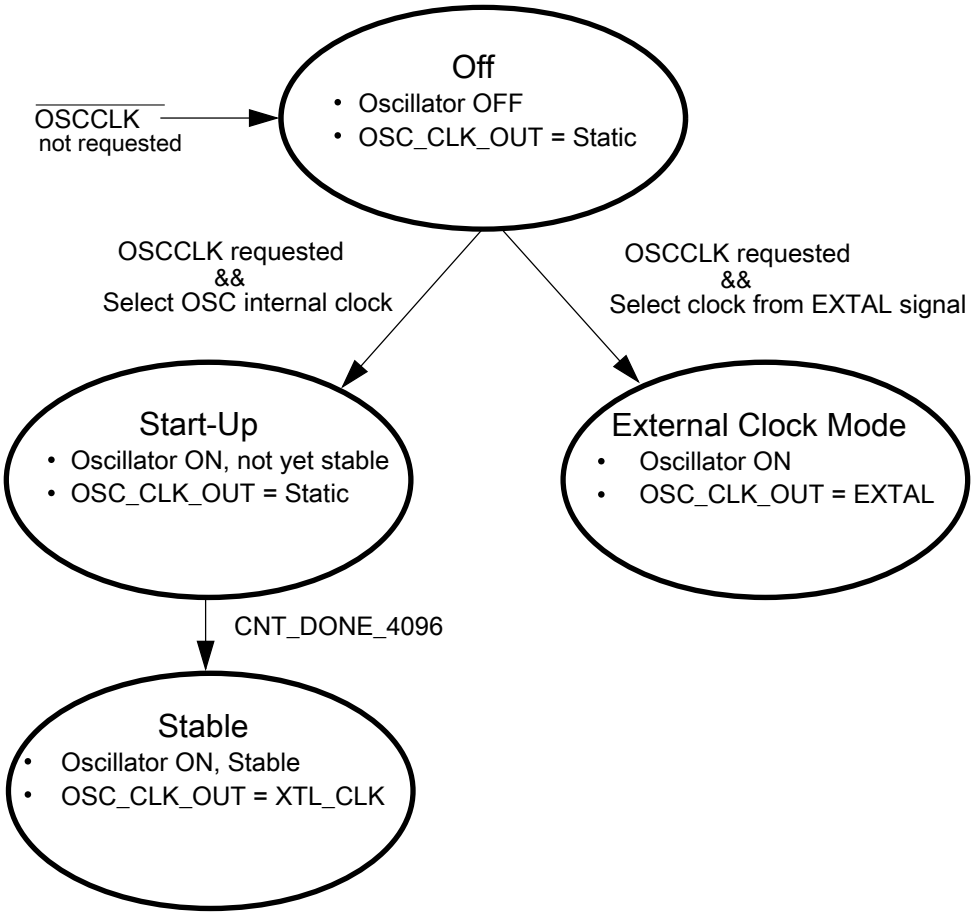


Figure 30-6. OSC Module state diagram

NOTE

XTL_CLK is the clock generated internally from OSC circuits.

30.9.1.1 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 details. 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.

30.9.1.2 Oscillator startup

The OSC enters startup 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.

30.9.1.3 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.

30.9.1.4 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, see the chip configuration details. 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.

30.9.2 OSC module modes

The OSC is a pierce-type oscillator that supports external crystals or resonators operating over the frequency ranges shown in Table 30-3. 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 30-3. Oscillator modes

Mode	Frequency Range
Low-frequency, high-gain	f_{osc_lo} (32.768 kHz) up to f_{osc_lo} (39.0625 kHz)

Table continues on the next page...

Table 30-3. Oscillator modes (continued)

Mode	Frequency Range
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, see the chip's Power Management details.

30.9.2.1 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.

30.9.2.2 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.

30.9.2.3 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.

30.9.2.4 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, 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.

30.9.3 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 timeout is used to guarantee output clock stability.

30.9.4 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.

30.10 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.

30.11 Low power modes operation

When the MCU enters Stop modes, the OSC is functional depending on CR[ERCLKEN] and CR[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 CR[ERCLKEN] and CR[EREFSTEN] 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.

30.12 Interrupts

The OSC module does not generate any interrupts.

Chapter 31
RTC Oscillator (OSC32K)

31.1 Introduction

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.

31.1.1 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 .

31.1.2 Block Diagram

The following is the block diagram of the RTC oscillator.

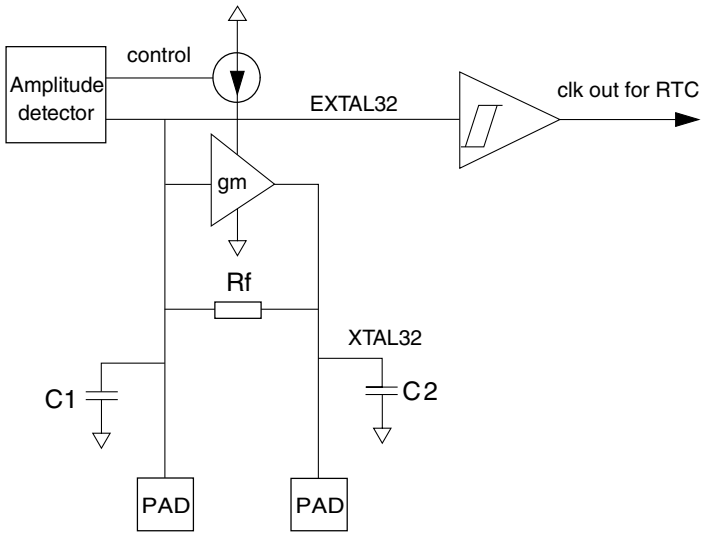


Figure 31-1. RTC Oscillator Block Diagram

31.2 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 31-1. RTC Signal Descriptions

Signal	Description	I/O
EXTAL32	Oscillator Input	I
XTAL32	Oscillator Output	O

31.2.1 EXTAL32 — Oscillator Input

This signal is the analog input of the RTC oscillator.

31.2.2 XTAL32 — Oscillator Output

This signal is the analog output of the RTC oscillator module.

31.3 External Crystal Connections

The connections with a crystal is shown in the following figure. External load capacitors and feedback resistor are not required.

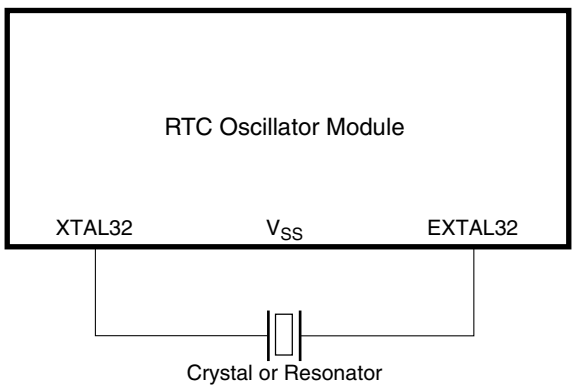


Figure 31-2. Crystal Connections

31.4 Memory Map/Register Descriptions

RTC oscillator control bits are part of the RTC registers. Refer to RTC Control Register (RTC_CR) , or RTC_GP_DATA_REG in the chip-specific information section, for more details.

31.5 Functional Description

As shown in Figure 31-1, 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.

31.6 Reset Overview

There is no reset state associated with the RTC oscillator.

31.7 Interrupts

The RTC oscillator does not generate any interrupts.

Chapter 32
Micro Trace Buffer (MTB)

32.1 Introduction

Microcontrollers using the Cortex-M0+ processor core include support for a CoreSight Micro Trace Buffer to provide program trace capabilities.

The proper name for this function is the CoreSight Micro Trace Buffer for the Cortex-M0+ Processor; in this document, it is simply abbreviated as the MTB.

The simple program trace function creates instruction address change-of-flow data packets in a user-defined region of the system RAM. Accordingly, the system RAM controller manages requests from two sources:

- AMBA-AHB reads and writes from the system bus
- program trace packet writes from the processor

As part of the MTB functionality, there is a DWT (Data Watchpoint and Trace) module that allows the user to define watchpoint addresses, or optionally, an address and data value, that when triggered, can be used to start or stop the program trace recording.

This document details the functionality of both the MTB_RAM and MTB_DWT capabilities.

32.1.1 Overview

A generic block diagram of the processor core and platform for this class of ultra low-end microcontrollers is shown as follows:

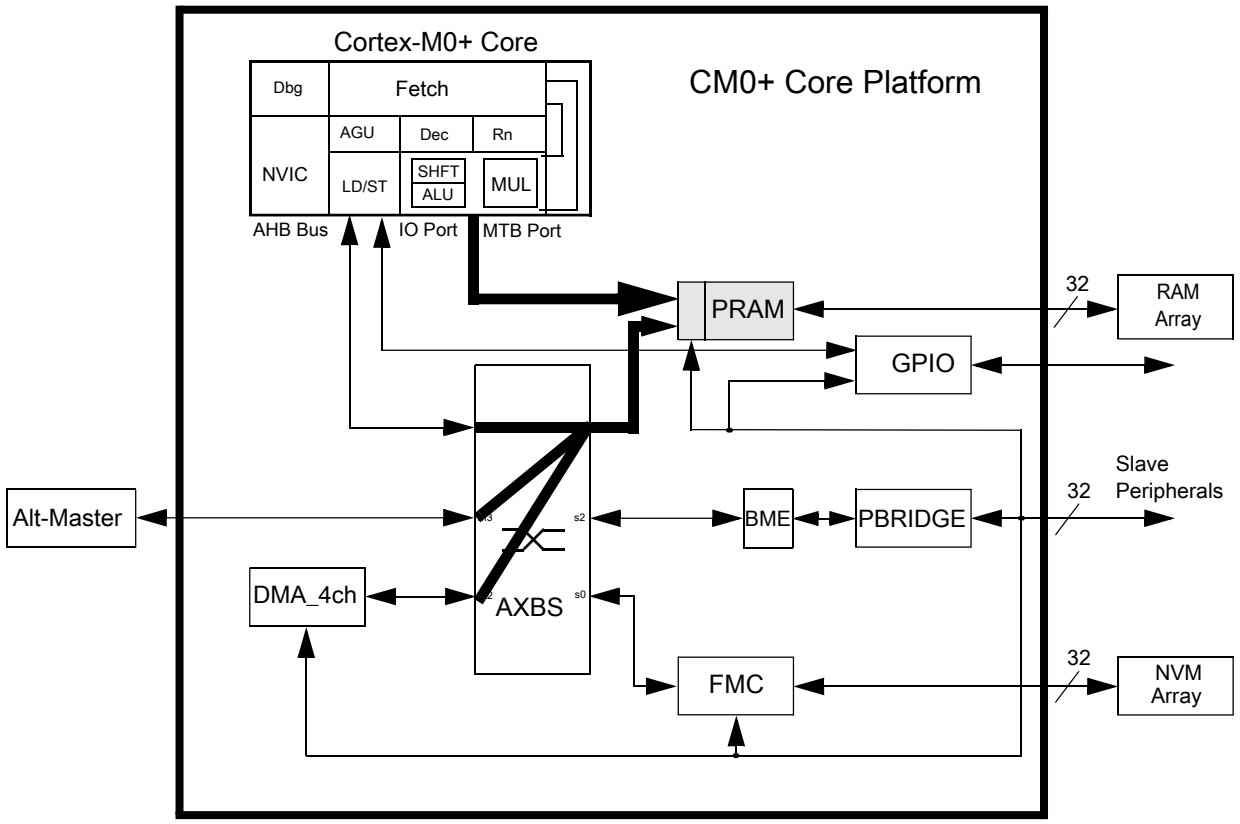


Figure 32-1. Generic Cortex-M0+ core platform block diagram

As shown in the block diagram, the platform RAM (PRAM) controller connects to two input buses:

- the crossbar slave port for system bus accesses
- a "private execution MTB port" from the core

The logical paths from the crossbar master input ports to the PRAM controller are highlighted along with the private execution trace port from the processor core. The private MTB port signals the instruction address information needed for the 64-bit program trace packets written into the system RAM. The PRAM controller output interfaces to the attached RAM array. In this document, the PRAM controller is the MTB_RAM controller.

The following information is taken from the ARM CoreSight Micro Trace Buffer documentation.

"The execution trace packet consists of a pair of 32-bit words that the MTB generates when it detects the processor PC value changes non-sequentially. A non-sequential PC change can occur during branch instructions or during exception entry.

The processor can cause a trace packet to be generated for any instruction.

The following figure shows how the execution trace information is stored in memory as a sequence of packets.

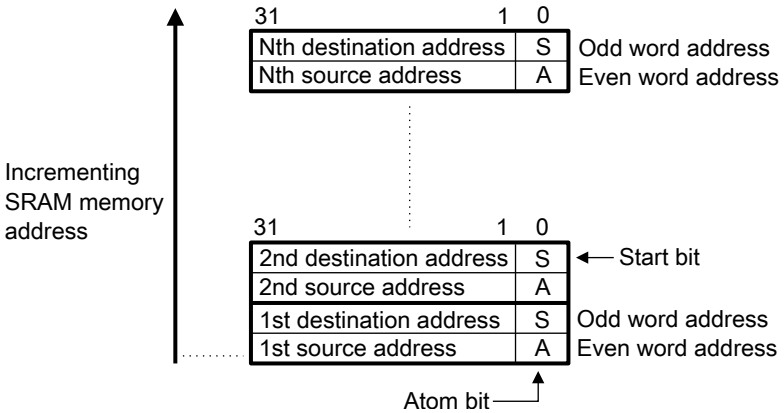


Figure 32-2. MTB execution trace storage format

The first, lower addressed, word contains the source of the branch, the address it branched from. The value stored only records bits[31:1] of the source address, because Thumb instructions are at least halfword aligned. The least significant bit of the value is the A-bit. The A-bit indicates the atomic state of the processor at the time of the branch, and can differentiate whether the branch originated from an instruction in a program, an exception, or a PC update in Debug state. When it is zero the branch originated from an instruction, when it is one the branch originated from an exception or PC update in Debug state. This word is always stored at an even word location.

The second, higher addressed word contains the destination of the branch, the address it branched to. The value stored only records bits[31:1] of the branch address. The least significant bit of the value is the S-bit. The S-bit indicates where the trace started. An S-bit value of 1 indicates where the first packet after the trace started and a value of 0 is used for other packets. Because it is possible to start and stop tracing multiple times in a trace session, the memory might contain several packets with the S-bit set to 1. This word is always stored in the next higher word in memory, an odd word address.

When the A-bit is set to 1, the source address field contains the architecturally-preferred return address for the exception. For example, if an exception was caused by an SVC instruction, then the source address field contains the address of the following instruction. This is different from the case where the A-bit is set to 0. In this case, the source address contains the address of the branch instruction.

For an exception return operation, two packets are generated:

- The first packet has the:
  - Source address field set to the address of the instruction that causes the exception return, BX or POP.

  - Destination address field set to bits[31:1] of the EXC_RETURN value. See the ARM v6-M Architecture Reference Manual.
  - The A-bit set to 0.

- The second packet has the:
  - Source address field set to bits[31:1] of the EXC_RETURN value.
  - Destination address field set to the address of the instruction where execution commences.
  - A-bit set to 1."

Given the recorded change-of-flow trace packets in system RAM and the memory image of the application, a debugger can read out the data and create an instruction-by-instruction program trace. In keeping with the low area and power implementation cost design targets, the MTB trace format is less efficient than other CoreSight trace modules, for example, the ETM (Embedded Trace Macrocell). Since each branch packet is 8 bytes in size, a 1 KB block of system RAM can contain 128 branches. Using the Dhrystone 2.1 benchmark's dynamic runtime as an example, this corresponds to about 875 instructions per KB of trace RAM, or with a zero wait state memory, this corresponds to approximately 1600 processor cycles per KB. This metric is obviously very sensitive to the runtime characteristics of the user code.

The MTB_DWT function (not shown in the core platform block diagram) monitors the processor address and data buses so that configurable watchpoints can be detected to trigger the appropriate response in the MTB recording.

32.1.2 Features

The key features of the MTB_RAM and MTB_DWT include:

- Memory controller for system RAM and Micro Trace Buffer for program trace packets
- Read/write capabilities for system RAM accesses, write-only for program trace packets
- Supports zero wait state response to system bus accesses when no trace data is being written
- Can buffer two AHB address phases and one data write for system RAM accesses
- Supports 64-bit program trace packets including source and destination instruction addresses
- Program trace information in RAM available to MCU's application code or external debugger
- Program trace watchpoint configuration accessible by MCU's application code or debugger
- Location and size of RAM trace buffer is configured by software

- Two DWT comparators (addresses or address + data) provide programmable start/ stop recording
- CoreSight compliant debug functionality

32.1.3 Modes of operation

The MTB_RAM and MTB_DWT functions do not support any special modes of operation. The MTB_RAM controller, as a memory-mapped device located on the platform's slave AHB system bus, responds strictly on the basis of memory addresses for accesses to its attached RAM array. The MTB private execution bus provides program trace packet write information to the RAM controller. Both the MTB_RAM and MTB_DWT modules are memory-mapped, so their programming models can be accessed.

All functionality associated with the MTB_RAM and MTB_DWT modules resides in the core platform's clock domain; this includes its connections with the RAM array.

32.2 External signal description

The MTB_RAM and MTB_DWT modules do not directly support any external interfaces.

The internal interface includes a standard AHB bus with a 32-bit datapath width from the appropriate crossbar slave port plus the private execution trace bus from the processor core. The signals in the private execution trace bus are detailed in the following table taken from the ARM CoreSight Micro Trace Buffer documentation. The signal direction is defined as viewed by the MTB_RAM controller.

Table 32-1. Private execution trace port from the core to MTB_RAM

Signal	Direction	Description
LOCKUP	Input	Indicates the processor is in the Lockup state. This signal is driven LOW for cycles when the processor is executing normally and driven HIGH for every cycle the processor is waiting in the Lockup state. This signal is valid on every cycle.
IAESEQ	Input	Indicates the next instruction address in execute, IAEX, is sequential, that is non-branching.
IAEXEN	Input	IAEX register enable.
IAEX[30:0]	Input	Registered address of the instruction in the execution stage, shifted right by one bit, that is, PC >> 1.
ATOMIC	Input	Indicates the processor is performing non-instruction related activities.
EDBGRQ	Output	Request for the processor to enter the Debug state, if enabled, and halt.

In addition, there are two signals formed by the MTB_DWT module and driven to the MTB_RAM controller: TSTART (trace start) and TSTOP (trace stop). These signals can be configured using the trace watchpoints to define programmable addresses and data values to affect the program trace recording state.

32.3 Memory map and register definition

The MTB_RAM and MTB_DWT modules each support a sparsely-populated 4 KB address space for their programming models. For each address space, there are a variety of control and configurable registers near the base address, followed by a large unused address space and finally a set of CoreSight registers to support dynamic determination of the debug configuration for the device.

Accesses to the programming model follow standard ARM conventions. Taken from the ARM CoreSight Micro Trace Buffer documentation, these are:

- Do not attempt to access reserved or unused address locations. Attempting to access these locations can result in UNPREDICTABLE behavior.
- The behavior of the MTB is UNPREDICTABLE if the registers with UNKNOWN reset values are not programmed prior to enabling trace.
- Unless otherwise stated in the accompanying text:
  - Do not modify reserved register bits
  - Ignore reserved register bits on reads
  - All register bits are reset to a logic 0 by a system or power-on reset
  - Use only word size, 32-bit, transactions to access all registers

32.3.1 MTB_RAM Memory Map

MTB memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F000_0000	MTB Position Register (MTB_POSITION)	32	R/W	Undefined	32.3.1.1/ 674
F000_0004	MTB Master Register (MTB_MASTER)	32	R/W	See section	32.3.1.2/ 675
F000_0008	MTB Flow Register (MTB_FLOW)	32	R/W	Undefined	32.3.1.3/ 677
F000_000C	MTB Base Register (MTB_BASE)	32	R	Undefined	32.3.1.4/ 679
F000_0F00	Integration Mode Control Register (MTB_MODECTRL)	32	R	0000_0000h	32.3.1.5/ 679

Table continues on the next page...

MTB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F000_0FA0	Claim TAG Set Register (MTB_TAGSET)	32	R	0000_0000h	32.3.1.6/ 680
F000_0FA4	Claim TAG Clear Register (MTB_TAGCLEAR)	32	R	0000_0000h	32.3.1.7/ 680
F000_0FB0	Lock Access Register (MTB_LOCKACCESS)	32	R	0000_0000h	32.3.1.8/ 681
F000_0FB4	Lock Status Register (MTB_LOCKSTAT)	32	R	0000_0000h	32.3.1.9/ 681
F000_0FB8	Authentication Status Register (MTB_AUTHSTAT)	32	R	0000_0000h	32.3.1.10/ 681
F000_0FBC	Device Architecture Register (MTB_DEVICEARCH)	32	R	4770_0A31h	32.3.1.11/ 682
F000_0FC8	Device Configuration Register (MTB_DEVICECFG)	32	R	0000_0000h	32.3.1.12/ 683
F000_0FCC	Device Type Identifier Register (MTB_DEVICETYPID)	32	R	0000_0031h	32.3.1.13/ 683
F000_0FD0	Peripheral ID Register (MTB_PERIPHID4)	32	R	See section	32.3.1.14/ 684
F000_0FD4	Peripheral ID Register (MTB_PERIPHID5)	32	R	See section	32.3.1.14/ 684
F000_0FD8	Peripheral ID Register (MTB_PERIPHID6)	32	R	See section	32.3.1.14/ 684
F000_0FDC	Peripheral ID Register (MTB_PERIPHID7)	32	R	See section	32.3.1.14/ 684
F000_0FE0	Peripheral ID Register (MTB_PERIPHID0)	32	R	See section	32.3.1.14/ 684
F000_0FE4	Peripheral ID Register (MTB_PERIPHID1)	32	R	See section	32.3.1.14/ 684
F000_0FE8	Peripheral ID Register (MTB_PERIPHID2)	32	R	See section	32.3.1.14/ 684
F000_0FEC	Peripheral ID Register (MTB_PERIPHID3)	32	R	See section	32.3.1.14/ 684
F000_0FF0	Component ID Register (MTB_COMPID0)	32	R	See section	32.3.1.15/ 684
F000_0FF4	Component ID Register (MTB_COMPID1)	32	R	See section	32.3.1.15/ 684
F000_0FF8	Component ID Register (MTB_COMPID2)	32	R	See section	32.3.1.15/ 684
F000_0FFC	Component ID Register (MTB_COMPID3)	32	R	See section	32.3.1.15/ 684

32.3.1.1 MTB Position Register (MTB_POSITION)

The MTB_POSITION register contains the Trace Write Address Pointer and Wrap fields. This register can be modified by the explicit programming model writes. It is also automatically updated by the MTB hardware when trace packets are being recorded.

The base address of the system RAM in the memory map dictates special consideration for the placement of the MTB. Consider the following guidelines:

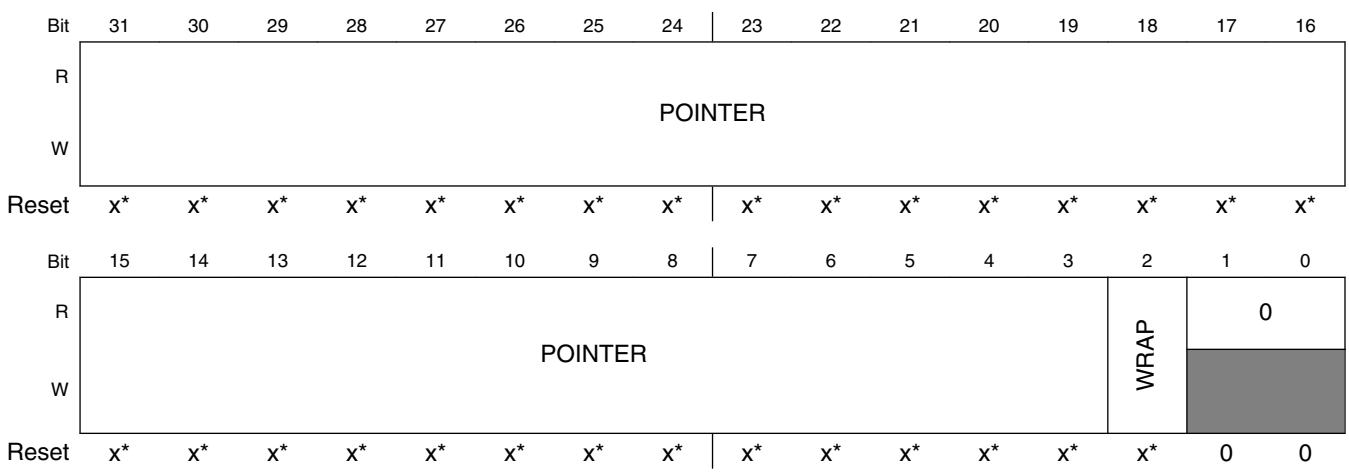
For the standard configuration where the size of the MTB is ≤ 25% of the total RAM capacity, it is recommended the MTB be based at the address defined by the MTB_BASE register. The read-only MTB_BASE register is defined by the expression (0x2000_0000 - (RAM_Size/4)). For this configuration, the MTB_POSITION register is initialized to MTB_BASE & 0x0001_FFF8.

If the size of the MTB is more than 25% but less than or equal to 50% of the total RAM capacity, it is recommended the MTB be based at address 0x2000_0000. In this configuration, the MTB_POSITION register is initialized to (0x2000_0000 & 0x0001_FFF8) = 0x0000_00000.

Following these two suggested placements provides a full-featured circular memory buffer containing program trace packets.

In the unlikely event an even larger trace buffer is required, a write-once capacity of 75% of the total RAM capacity can be based at address 0x2000_0000. The MTB_POSITION register is initialized to (0x2000_0000 & 0x0001_FFF8) = 0x0000_0000. However, this configuration cannot support operation as a circular queue and instead requires the use of the MTB_FLOW[WATERMARK] capability to automatically disable tracing or halting the processor as the number of packet writes approach the buffer capacity. See the MTB_FLOW register description for more details.

Address: F000_0000h base + 0h offset = F000_0000h



* Notes:
- x = Undefined at reset.

MTB_POSITION field descriptions

Field	Description

31–3
POINTER
	Trace Packet Address Pointer[28:0]
	
	Because a packet consists of two words, the POINTER field is the address of the first word of a packet. This field contains bits[31:3] of the RAM address where the next trace packet is written. Therefore, it points to an unused location and is automatically incremented.
	
	A debug agent can calculate the system memory map address for the current location in the MTB using the following "generic" equation:
	
	Given mtb_size = 1 << (MTB_MASTER[MASK] + 4),
	
	systemAddress = MTB_BASE + (((MTB_POSITION & 0xFFFF_FFF8) + (mtb_size - (MTB_BASE & (mtb_size-1)))) & 0x0001_FFF8);
	
	For this device, a simpler expression also applies. See the following pseudo-code:
	
	if ((MTB_POSITION >> 16) == 0x3) systemAddress = (0x7FFF << 18) + (MTB_POSITION & 0x3_FFF8); else systemAddress = (0x800 << 18) + (MTB_POSITION & 0x3_FFF8);
	
	NOTE: The size of the RAM is parameterized and the most significant bits of the POINTER field are RAZ/WI.
	
	For these devices, POSITION[31:18] == POSITION[POINTER[28:15]] are RAZ/WI. Therefore, the active bits in this field are POSITION[17:3] == POSITION[POINTER[14:0]].

2 WRAP	WRAP This field is set to 1 automatically when the POINTER value wraps as determined by the MTB_MASTER[MASK] field in the MASTER Trace Control Register. A debug agent might use the WRAP field to determine whether the trace information above and below the pointer address is valid.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

32.3.1.2 MTB Master Register (MTB_MASTER)

The MTB_MASTER register contains the main program trace enable plus other trace controls. This register can be modified by the explicit programming model writes. MTB_MASTER[EN] and MTB_MASTER[HALTREQ] fields are also automatically updated by the MTB hardware.

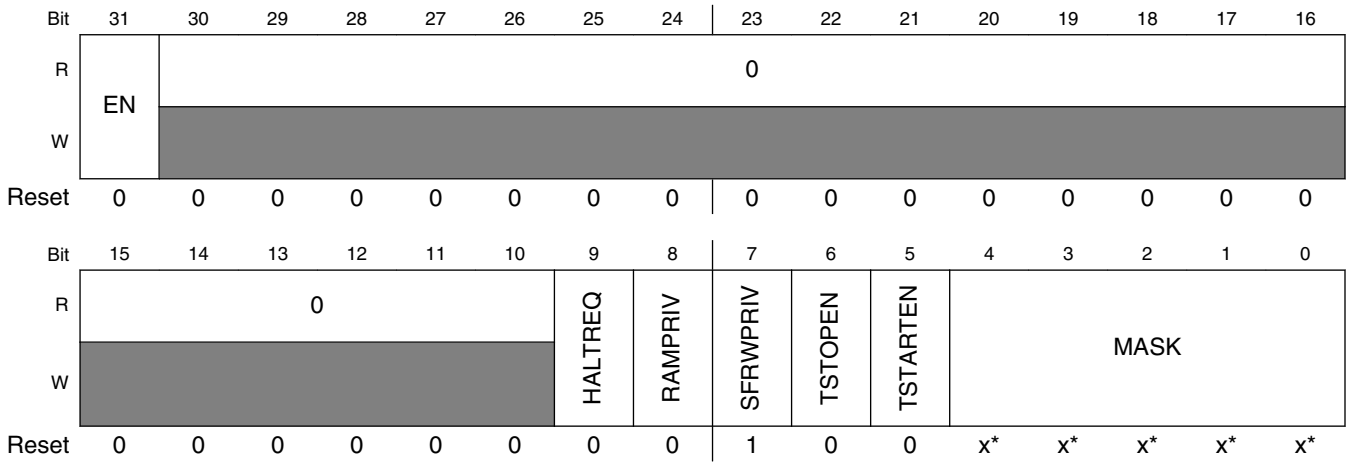
Before MTB_MASTER[EN] or MTB_MASTER[TSTARTEN] are set to 1, the software must initialize the MTB_POSITION and MTB_FLOW registers.

If MTB_FLOW[WATERMARK] is used to stop tracing or to halt the processor, MTB_MASTER[MASK] must still be set to a value that prevents MTB_POSITION[POINTER] from wrapping before it reaches the MTB_FLOW[WATERMARK] value.

NOTE

The format of this mask field is different than MTBDWT_MASKn[MASK].

Address: F000_0000h base + 4h offset = F000_0004h



* Notes:
- x = Undefined at reset.

MTB_MASTER field descriptions

Field	Description
31 EN	Main Trace Enable When this field is 1, trace data is written into the RAM memory location addressed by MTB_POSITION[POINTER]. The MTB_POSITION[POINTER] value auto increments after the trace data packet is written. EN can be automatically set to 0 using the MTB_FLOW[WATERMARK] field and the MTB_FLOW[AUTOSTOP] bit. EN is automatically set to 1 if TSTARTEN is 1 and the TSTART signal is HIGH. EN is automatically set to 0 if TSTOPEN is 1 and the TSTOP signal is HIGH. **NOTE:** If EN is set to 0 because MTB_FLOW[WATERMARK] is set, then it is not automatically set to 1 if TSTARTEN is 1 and the TSTART input is HIGH. In this case, tracing can only be restarted if MTB_FLOW[WATERMARK] or MTB_POSITION[POINTER] value is changed by software.
30–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 HALTREQ	Halt Request This field is connected to the halt request signal of the trace logic, EDBGRQ. When HALTREQ is set to 1, the EDBFGRQ is asserted if DBGEN (invasive debug enable, one of the debug authentication interface signals) is also HIGH. HALTREQ can be automatically set to 1 using MTB_FLOW[WATERMARK].
8 RAMPRIV	RAM Privilege If this field is 0, then user or privileged AHB read and write accesses to the RAM are permitted. If this field is 1, then only privileged AHB read and write accesses to the RAM are permitted and user accesses are RAZ/WI. The HPROT[1] signal determines if an access is a user or privileged mode reference.

Table continues on the next page...

MTB_MASTER field descriptions (continued)

Field	Description
7 SFRWPRIV	Special Function Register Write Privilege If this field is 0, then user or privileged AHB read and write accesses to the MTB_RAM Special Function Registers (programming model) are permitted. If this field is 1, then only privileged write accesses are permitted; user write accesses are ignored. The HPROT[1] signal determines if an access is user or privileged. Note MTB_RAM SFR read access are not controlled by this bit and are always permitted.
6 TSTOPEN	Trace Stop Input Enable If this field is 1 and the TSTOP signal is HIGH, then EN is set to 0. If a trace packet is being written to memory, the write is completed before tracing is stopped.
5 TSTARTEN	Trace Start Input Enable If this field is 1 and the TSTART signal is HIGH, then EN is set to 1. Tracing continues until a stop condition occurs.
MASK	Mask This value determines the maximum size of the trace buffer in RAM. It specifies the most-significant bit of the MTB_POSITION[POINTER] field that can be updated by automatic increment. If the trace tries to advance past this power of 2, the MTB_POSITION[WRAP] bit is set to 1, the MTB_POSITION[MASK+3:3] == MTB_POSITION[POINTER[MASK:0]] bits are set to 0, and the MTB_POSITION[14:MASK+3] == MTB_POSITION[POINTER[11:MASK+1]] bits remain unchanged. This field causes the trace packet information to be stored in a circular buffer of size 2^[MASK+4] bytes, that can be positioned in memory at multiples of this size. As detailed in the MTB_POSITION description, typical "upper limits" for the MTB size are RAM_Size/4 or RAM_Size/2. Values greater than the maximum have the same effect as the maximum.

32.3.1.3 MTB Flow Register (MTB_FLOW)

The MTB_FLOW register contains the watermark address and the autostop/autohalt control bits.

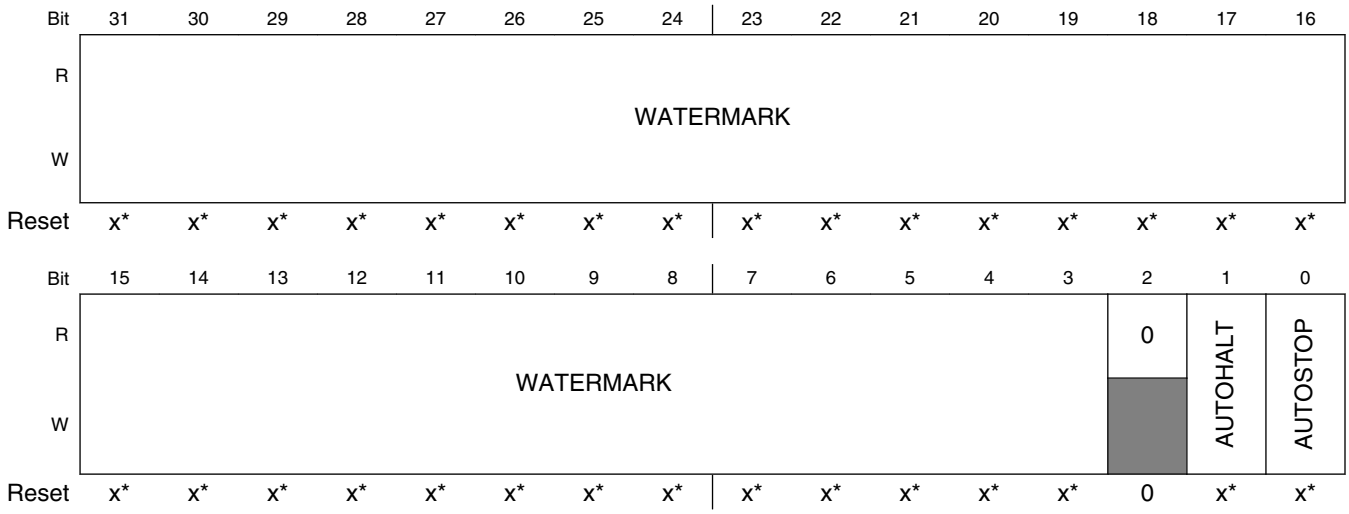
If tracing is stopped using the watermark autostop feature, it cannot be restarted until software clears the watermark autostop. This can be achieved in one of the following ways:

- Changing the MTB_POSITION[POINTER] field value to point to the beginning of the trace buffer, or
- Setting MTB_FLOW[AUTOSTOP] = 0.

A debug agent can use MTB_FLOW[AUTOSTOP] to fill the trace buffer once only without halting the processor.

A debug agent can use MTB_FLOW[AUTOHALT] to fill the trace buffer once before causing the Cortex-M0+ processor to enter the Debug state. To enter Debug state, the Cortex-M0+ processor might have to perform additional branch type operations. Therefore, the MTB_FLOW[WATERMARK] field must be set below the final entry in the trace buffer region.

Address: F000_0000h base + 8h offset = F000_0008h



* Notes:
- x = Undefined at reset.

MTB_FLOW field descriptions

Field	Description
31–3 WATERMARK	WATERMARK[28:0] This field contains an address in the same format as the MTB_POSITION[POINTER] field. When MTB_POSITION[POINTER] matches the WATERMARK field value, actions defined by the AUTOHALT and AUTOSTOP bits are performed.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 AUTOHALT	AUTOHALT If this field is 1 and WATERMARK is equal to MTB_POSITION[POINTER], then MTB_MASTER[HALTREQ] is automatically set to 1. If the DBGEN signal is HIGH, the MTB asserts this halt request to the Cortex-M0+ processor by asserting the EDBGRQ signal.
0 AUTOSTOP	AUTOSTOP If this field is 1 and WATERMARK is equal to MTB_POSITION[POINTER], then MTB_MASTER[EN] is automatically set to 0. This stops tracing.

32.3.1.4 MTB Base Register (MTB_BASE)

The read-only MTB_BASE Register indicates where the RAM is located in the system memory map. This register is provided to enable auto discovery of the MTB RAM location, by a debug agent and is defined by a hardware design parameter. For this device, the base address is defined by the expression: MTB_BASE[BASEADDR] = 0x2000_0000 - (RAM_Size/4)

Address: F000_0000h base + Ch offset = F000_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	BASEADDR																															
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.

MTB_BASE field descriptions

Field	Description
BASEADDR	BASEADDR This value is defined with a hardwired signal and the expression: 0x2000_0000 - (RAM_Size/4). For example, if the total RAM capacity is 16 KB, this field is 0x1FFF_F000.

32.3.1.5 Integration Mode Control Register (MTB_MODECTRL)

This register enables the device to switch from a functional mode, or default behavior, into integration mode. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + F00h offset = F000_0F00h

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	MODECTRL																															
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

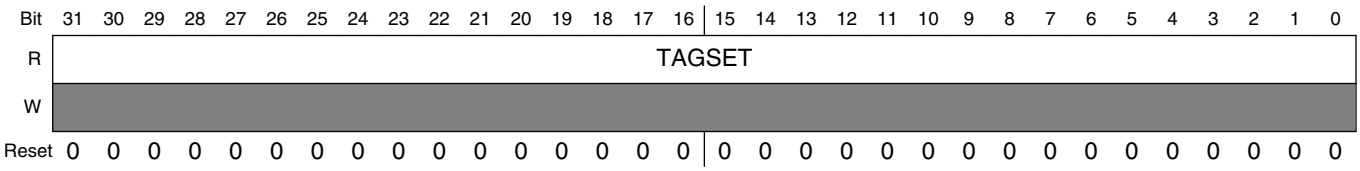
MTB_MODECTRL field descriptions

Field	Description
MODECTRL	MODECTRL Hardwired to 0x0000_0000

32.3.1.6 Claim TAG Set Register (MTB_TAGSET)

The Claim Tag Set Register returns the number of bits that can be set on a read, and enables individual bits to be set on a write. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + FA0h offset = F000_0FA0h



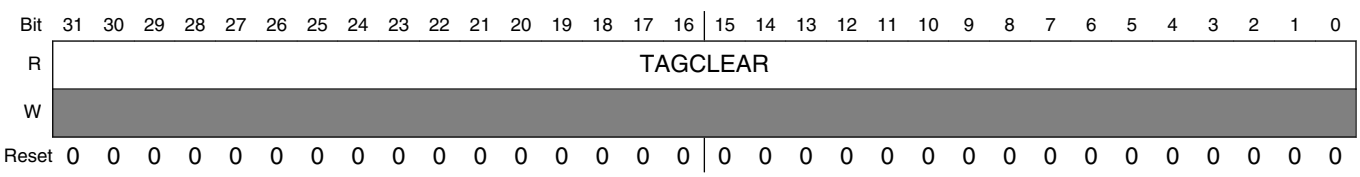
MTB_TAGSET field descriptions

Field	Description
TAGSET	TAGSET Hardwired to 0x0000_0000

32.3.1.7 Claim TAG Clear Register (MTB_TAGCLEAR)

The read/write Claim Tag Clear Register is used to read the claim status on debug resources. A read indicates the claim tag status. Writing 1 to a specific bit clears the corresponding claim tag to 0. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + FA4h offset = F000_0FA4h



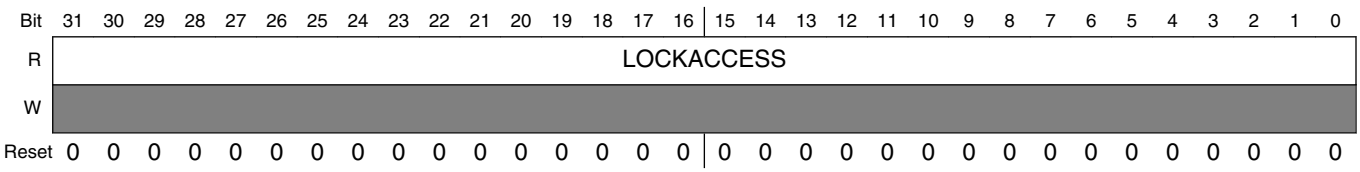
MTB_TAGCLEAR field descriptions

Field	Description
TAGCLEAR	TAGCLEAR Hardwired to 0x0000_0000

32.3.1.8 Lock Access Register (MTB_LOCKACCESS)

The Lock Access Register enables a write access to component registers. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + FB0h offset = F000_0FB0h



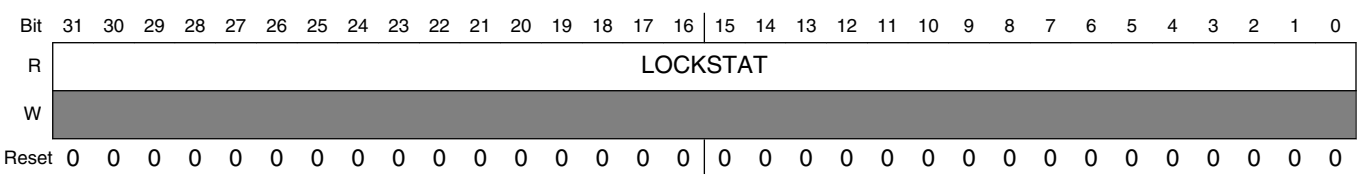
MTB_LOCKACCESS field descriptions

Field	Description
LOCKACCESS	Hardwired to 0x0000_0000

32.3.1.9 Lock Status Register (MTB_LOCKSTAT)

The Lock Status Register indicates the status of the lock control mechanism. This register is used in conjunction with the Lock Access Register. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + FB4h offset = F000_0FB4h



MTB_LOCKSTAT field descriptions

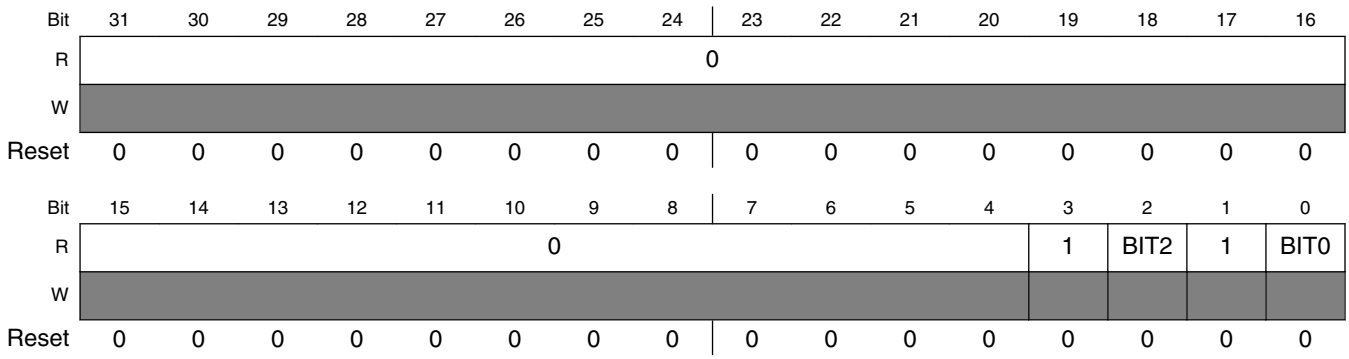
Field	Description
LOCKSTAT	LOCKSTAT Hardwired to 0x0000_0000

32.3.1.10 Authentication Status Register (MTB_AUTHSTAT)

The Authentication Status Register reports the required security level and current status of the security enable bit pairs. Where functionality changes on a given security level, this change must be reported in this register. It is connected to specific signals used during the auto-discovery process by an external debug agent.

MTB_AUTHSTAT[3:2] indicates if nonsecure, noninvasive debug is enabled or disabled, while MTB_AUTHSTAT[1:0] indicates the enabled/disabled state of nonsecure, invasive debug. For both 2-bit fields, 0b10 indicates the functionality is disabled and 0b11 indicates it is enabled.

Address: F000_0000h base + FB8h offset = F000_0FB8h



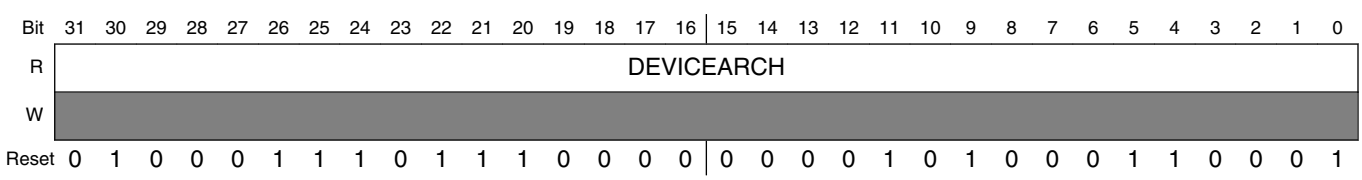
MTB_AUTHSTAT field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	BIT3 This read-only field is reserved and always has the value 1.
2 BIT2	BIT2 Connected to NIDEN or DBGEN signal.
1 Reserved	BIT1 This read-only field is reserved and always has the value 1.
0 BIT0	Connected to DBGEN.

32.3.1.11 Device Architecture Register (MTB_DEVICEARCH)

This register indicates the device architecture. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + FBCh offset = F000_0FBCh



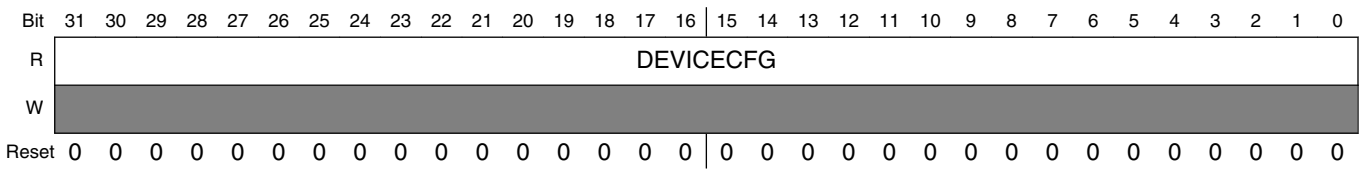
MTB_DEVICEARCH field descriptions

Field	Description
DEVICEARCH	DEVICEARCH Hardwired to 0x4770_0A31.

32.3.1.12 Device Configuration Register (MTB_DEVICECFG)

This register indicates the device configuration. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + FC8h offset = F000_0FC8h



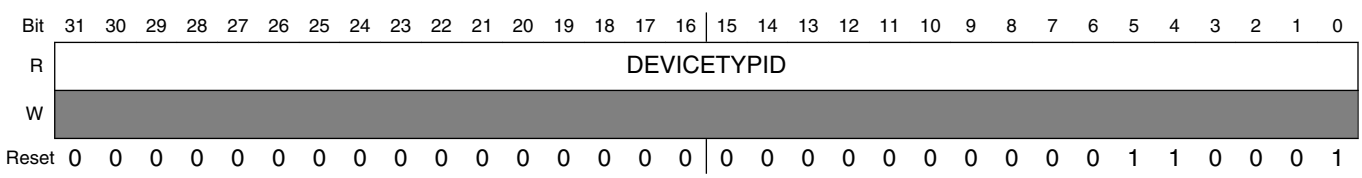
MTB_DEVICECFG field descriptions

Field	Description
DEVICECFG	DEVICECFG Hardwired to 0x0000_0000.

32.3.1.13 Device Type Identifier Register (MTB_DEVICETYPID)

This register indicates the device type ID. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + FCCh offset = F000_0FCCh



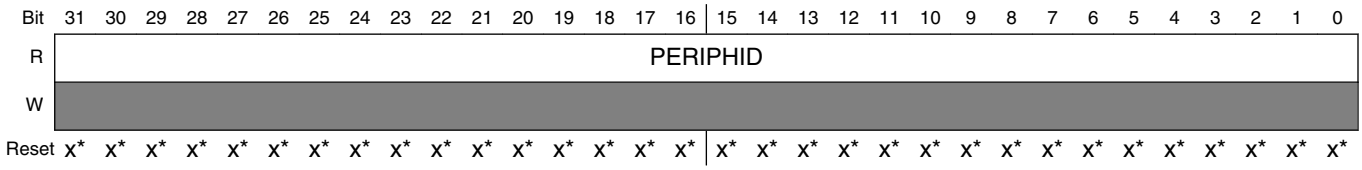
MTB_DEVICETYPID field descriptions

Field	Description
DEVICETYPID	DEVICETYPID Hardwired to 0x0000_0031.

32.3.1.14 Peripheral ID Register (MTB_PERIPHID*n*)

These registers indicate the peripheral IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + FD0h offset + (4d × i), where i=0d to 7d



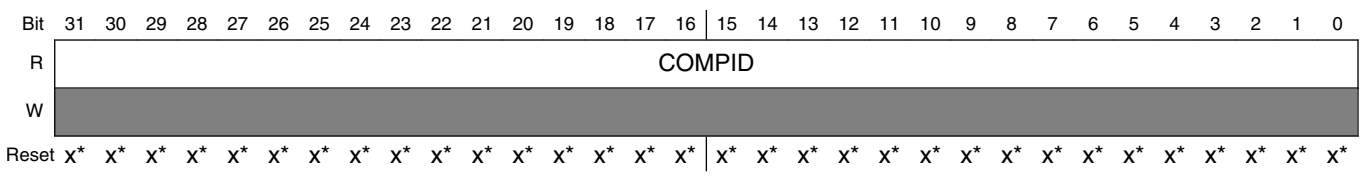
MTB_PERIPHID*n* field descriptions

Field	Description
PERIPHID	PERIPHID Peripheral ID4 is hardwired to 0x0000_0004; ID0 to 0x0000_0032; ID1 to 0x0000_00B9; ID2 to 0x0000_001B; and all the others to 0x0000_0000.

32.3.1.15 Component ID Register (MTB_COMPID*n*)

These registers indicate the component IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + FF0h offset + (4d × i), where i=0d to 3d



MTB_COMPID*n* field descriptions

Field	Description
COMPID	Component ID Component ID0 is hardwired to 0x0000_000D; ID1 to 0x0000_0090; ID2 to 0x0000_0005; ID3 to 0x0000_00B1.

32.3.2 MTB_DWT Memory Map

The MTB_DWT programming model supports a very simplified subset of the v7M debug architecture and follows the standard ARM DWT definition.

MTB_DWT memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F000_1000	MTB DWT Control Register (MTB_DWT_CTRL)	32	R	2F00_0000h	32.3.2.1/ 686
F000_1020	MTB_DWT Comparator Register (MTB_DWT_COMP0)	32	R/W	0000_0000h	32.3.2.2/ 687
F000_1024	MTB_DWT Comparator Mask Register (MTB_DWT_MASK0)	32	R/W	0000_0000h	32.3.2.3/ 687
F000_1028	MTB_DWT Comparator Function Register 0 (MTB_DWT_FCT0)	32	R/W	0000_0000h	32.3.2.4/ 688
F000_1030	MTB_DWT Comparator Register (MTB_DWT_COMP1)	32	R/W	0000_0000h	32.3.2.2/ 687
F000_1034	MTB_DWT Comparator Mask Register (MTB_DWT_MASK1)	32	R/W	0000_0000h	32.3.2.3/ 687
F000_1038	MTB_DWT Comparator Function Register 1 (MTB_DWT_FCT1)	32	R/W	0000_0000h	32.3.2.5/ 690
F000_1200	MTB_DWT Trace Buffer Control Register (MTB_DWT_TBCTRL)	32	R/W	2000_0000h	32.3.2.6/ 691
F000_1FC8	Device Configuration Register (MTB_DWT_DEVICECFG)	32	R	0000_0000h	32.3.2.7/ 693
F000_1FCC	Device Type Identifier Register (MTB_DWT_DEVICETYPID)	32	R	0000_0004h	32.3.2.8/ 693
F000_1FD0	Peripheral ID Register (MTB_DWT_PERIPHID4)	32	R	See section	32.3.2.9/ 694
F000_1FD4	Peripheral ID Register (MTB_DWT_PERIPHID5)	32	R	See section	32.3.2.9/ 694
F000_1FD8	Peripheral ID Register (MTB_DWT_PERIPHID6)	32	R	See section	32.3.2.9/ 694
F000_1FDC	Peripheral ID Register (MTB_DWT_PERIPHID7)	32	R	See section	32.3.2.9/ 694
F000_1FE0	Peripheral ID Register (MTB_DWT_PERIPHID0)	32	R	See section	32.3.2.9/ 694
F000_1FE4	Peripheral ID Register (MTB_DWT_PERIPHID1)	32	R	See section	32.3.2.9/ 694
F000_1FE8	Peripheral ID Register (MTB_DWT_PERIPHID2)	32	R	See section	32.3.2.9/ 694
F000_1FEC	Peripheral ID Register (MTB_DWT_PERIPHID3)	32	R	See section	32.3.2.9/ 694
F000_1FF0	Component ID Register (MTB_DWT_COMPID0)	32	R	See section	32.3.2.10/ 694
F000_1FF4	Component ID Register (MTB_DWT_COMPID1)	32	R	See section	32.3.2.10/ 694
F000_1FF8	Component ID Register (MTB_DWT_COMPID2)	32	R	See section	32.3.2.10/ 694
F000_1FFC	Component ID Register (MTB_DWT_COMPID3)	32	R	See section	32.3.2.10/ 694

32.3.2.1 MTB DWT Control Register (MTB_DWT_CTRL)

The MTBDWT_CTRL register provides read-only information on the watchpoint configuration for the MTB_DWT.

Address: F000_1000h base + 0h offset = F000_1000h

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	NUMCMP				DWTCFGCTRL																											
W																																
Reset	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MTB_DWT_CTRL field descriptions

Field	Description
31–28 NUMCMP	Number of comparators The MTB_DWT implements two comparators.
DWTCFGCTRL	DWT configuration controls This field is hardwired to 0xF00_0000, disabling all the remaining DWT functionality. The specific fields and their state are: MTBDWT_CTRL[27] = NOTRCPKT = 1, trace sample and exception trace is not supported MTBDWT_CTRL[26] = NOEXTTRIG = 1, external match signals are not supported MTBDWT_CTRL[25] = NOCYCCNT = 1, cycle counter is not supported MTBDWT_CTRL[24] = NOPRFCNT = 1, profiling counters are not supported MTBDWT_CTRL[22] = CYCEBTENA = 0, no POSTCNT underflow packets generated MTBDWT_CTRL[21] = FOLDEVTENA = 0, no folded instruction counter overflow events MTBDWT_CTRL[20] = LSUEVTENA = 0, no LSU counter overflow events MTBDWT_CTRL[19] = SLEEPEVTENA = 0, no sleep counter overflow events MTBDWT_CTRL[18] = EXCEVTENA = 0, no exception overhead counter events MTBDWT_CTRL[17] = CPIEVTENA = 0, no CPI counter overflow events MTBDWT_CTRL[16] = EXCTRCENA = 0, generation of exception trace disabled MTBDWT_CTRL[12] = PCSAMPLENA = 0, no periodic PC sample packets generated MTBDWT_CTRL[11:10] = SYNCTAP = 0, no synchronization packets MTBDWT_CTRL[9] = CYCTAP = 0, cycle counter is not supported MTBDWT_CTRL[8:5] = POSTINIT = 0, cycle counter is not supported MTBDWT_CTRL[4:1] = POSTPRESET = 0, cycle counter is not supported MTBDWT_CTRL[0] = CYCCNTENA = 0, cycle counter is not supported

32.3.2.2 MTB_DWT Comparator Register (MTB_DWT_COMP*n*)

The MTBDWT_COMPn registers provide the reference value for comparator n.

Address: F000_1000h base + 20h offset + (16d × 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 W	COMP																															
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

MTB_DWT_COMP*n* field descriptions

Field	Description
COMP	Reference value for comparison If MTBDWT_COMP0 is used for a data value comparator and the access size is byte or halfword, the data value must be replicated across all appropriate byte lanes of this register. For example, if the data is a byte-sized "x" value, then COMP[31:24] = COMP[23:16] = COMP[15:8] = COMP[7:0] = "x". Likewise, if the data is a halfword-size "y" value, then COMP[31:16] = COMP[15:0] = "y".

32.3.2.3 MTB_DWT Comparator Mask Register (MTB_DWT_MASK*n*)

The MTBDWT_MASKn registers define the size of the ignore mask applied to the reference address for address range matching by comparator n. Note the format of this mask field is different than the MTB_MASTER[MASK].

Address: F000_1000h base + 24h offset + (16d × 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																											MASK				
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

MTB_DWT_MASK*n* field descriptions

Field	Description
31–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
MASK	MASK The value of the ignore mask, 0-31 bits, is applied to address range matching. MASK = 0 is used to include all bits of the address in the comparison, except if MASK = 0 and the comparator is configured to watch instruction fetch addresses, address bit [0] is ignored by the hardware since all fetches must be at least halfword aligned. For MASK != 0 and regardless of watch type, address bits [x-1:0] are ignored in the address comparison. Using a mask means the comparator matches on a range of addresses, defined by the unmasked most significant bits of the address, bits [31:x]. The maximum MASK value is 24, producing a 16 Mbyte mask. An attempted write of a MASK value > 24 is limited by the MTBDWT hardware to 24.

Table continues on the next page...

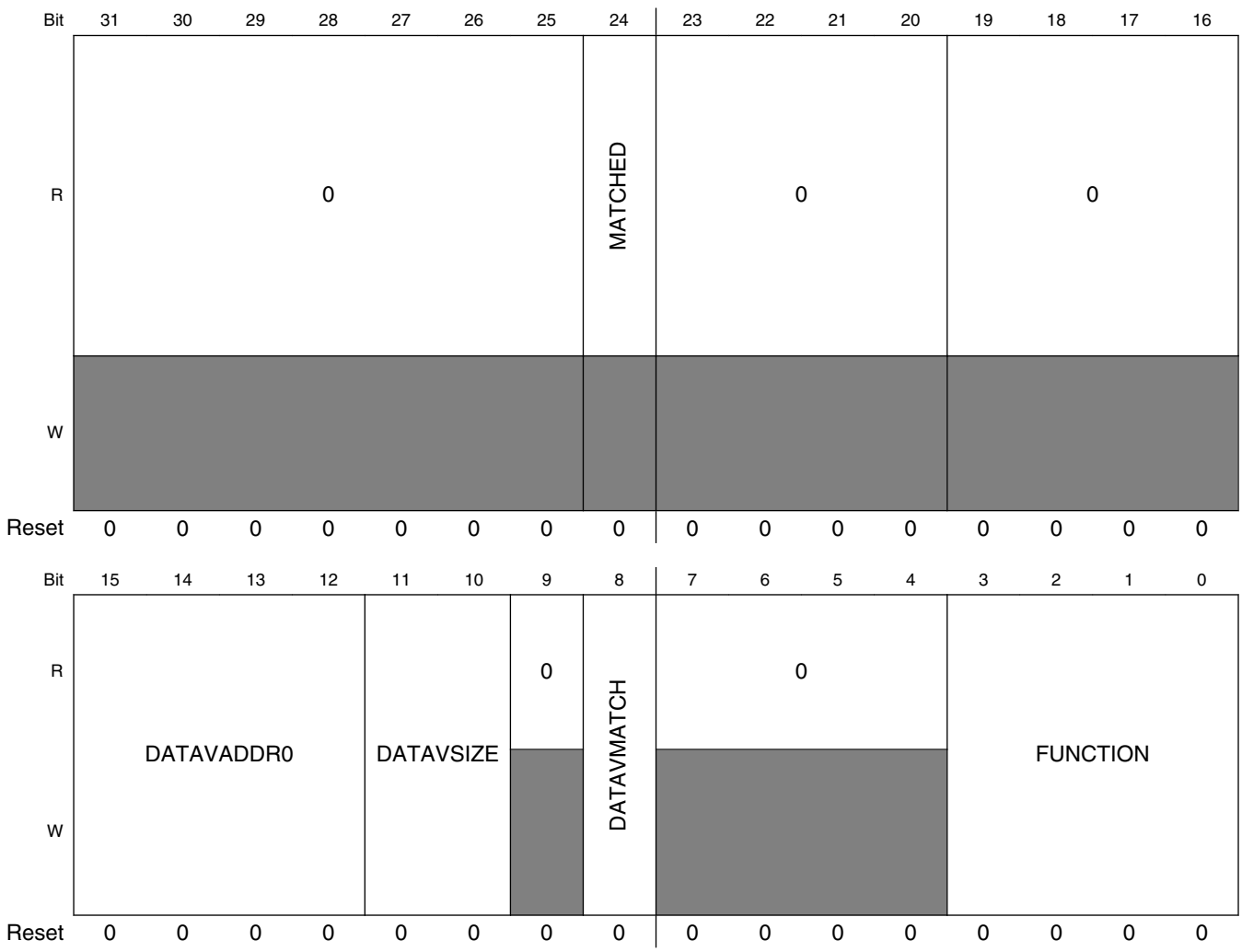
MTB_DWT_MASK*n* field descriptions (continued)

Field	Description
	If MTBDWT_COMP0 is used as a data value comparator, then MTBDWT_MASK0 should be programmed to zero.

32.3.2.4 MTB_DWT Comparator Function Register 0 (MTB_DWT_FCT0)

The MTBDWT_FCTn registers control the operation of comparator n.

Address: F000_1000h base + 28h offset = F000_1028h



MTB_DWT_FCT0 field descriptions

Field	Description
31–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

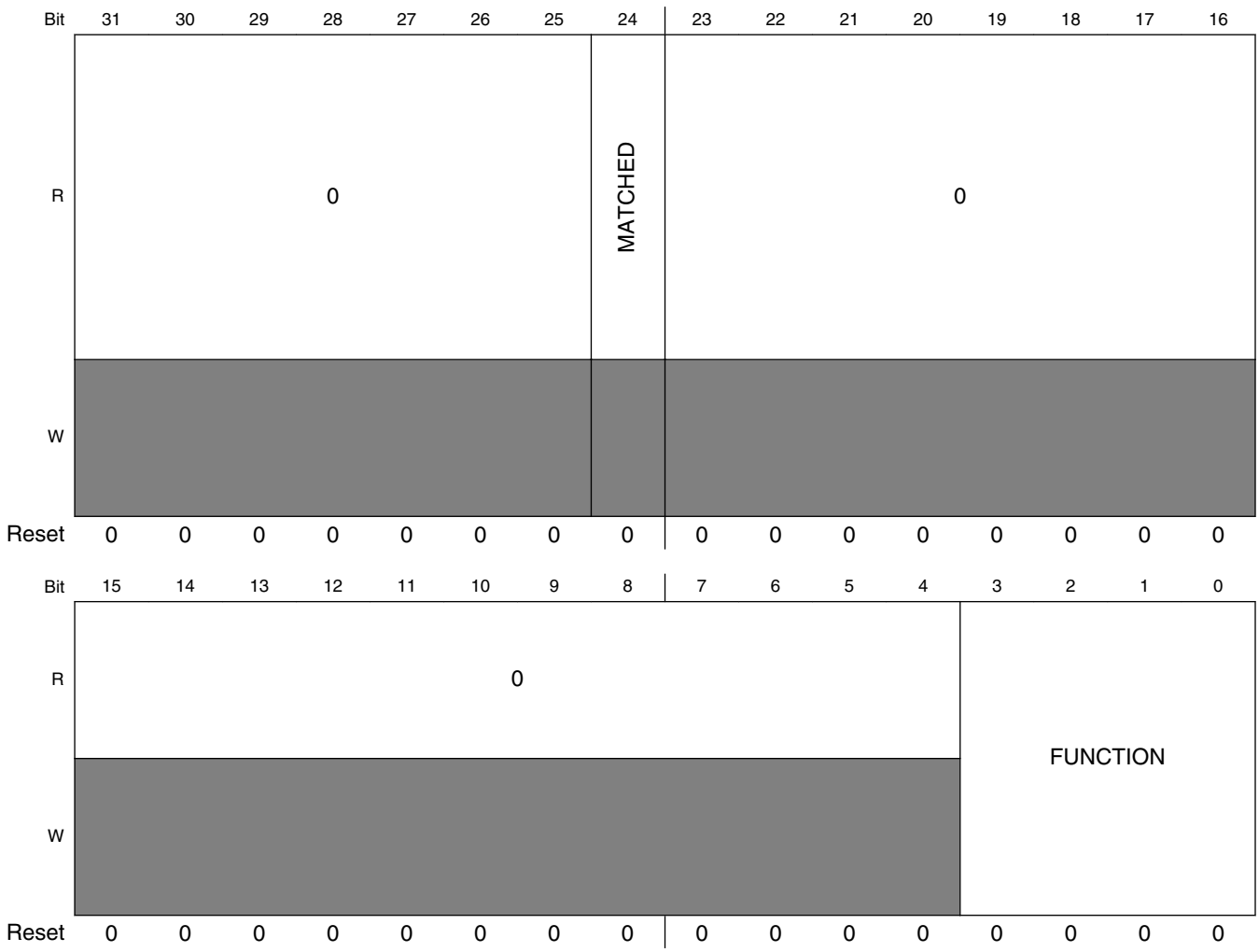
MTB_DWT_FCT0 field descriptions (continued)

Field	Description
24 MATCHED	Comparator match If this read-only flag is asserted, it indicates the operation defined by the FUNCTION field occurred since the last read of the register. Reading the register clears this bit. 0 No match. 1 Match occurred.
23–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–12 DATAVADDR0	Data Value Address 0 Since the MTB_DWT implements two comparators, the DATAVADDR0 field is restricted to values {0,1}. When the DATAVMATCH bit is asserted, this field defines the comparator number to use for linked address comparison. If MTBDWT_COMP0 is used as a data watchpoint and MTBDWT_COMP1 as an address watchpoint, DATAVADDR0 must be set.
11–10 DATAVSIZE	Data Value Size For data value matching, this field defines the size of the required data comparison. 00 Byte. 01 Halfword. 10 Word. 11 Reserved. Any attempts to use this value results in UNPREDICTABLE behavior.
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 DATAVMATCH	Data Value Match When this field is 1, it enables data value comparison. For this implementation, MTBDWT_COMP0 supports address or data value comparisons; MTBDWT_COMP1 only supports address comparisons. 0 Perform address comparison. 1 Perform data value comparison.
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
FUNCTION	Function Selects the action taken on a comparator match. If MTBDWT_COMP0 is used for a data value and MTBDWT_COMP1 for an address value, then MTBDWT_FCT1[FUNCTION] must be set to zero. For this configuration, MTBDWT_MASK1 can be set to a non-zero value, so the combined comparators match on a range of addresses. 0000 Disabled. 0100 Instruction fetch. 0101 Data operand read. 0110 Data operand write. 0111 Data operand (read + write). others Reserved. Any attempts to use this value results in UNPREDICTABLE behavior.

32.3.2.5 MTB_DWT Comparator Function Register 1 (MTB_DWT_FCT1)

The MTBDWT_FCTn registers control the operation of comparator n. Since the MTB_DWT only supports data value comparisons on comparator 0, there are several fields in the MTBDWT_FCT1 register that are RAZ/WI (bits 12, 11:10, 8).

Address: F000_1000h base + 38h offset = F000_1038h



MTB_DWT_FCT1 field descriptions

Field	Description
31–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 MATCHED	Comparator match If this read-only flag is asserted, it indicates the operation defined by the FUNCTION field occurred since the last read of the register. Reading the register clears this bit.

Table continues on the next page...

MTB_DWT_FCT1 field descriptions (continued)

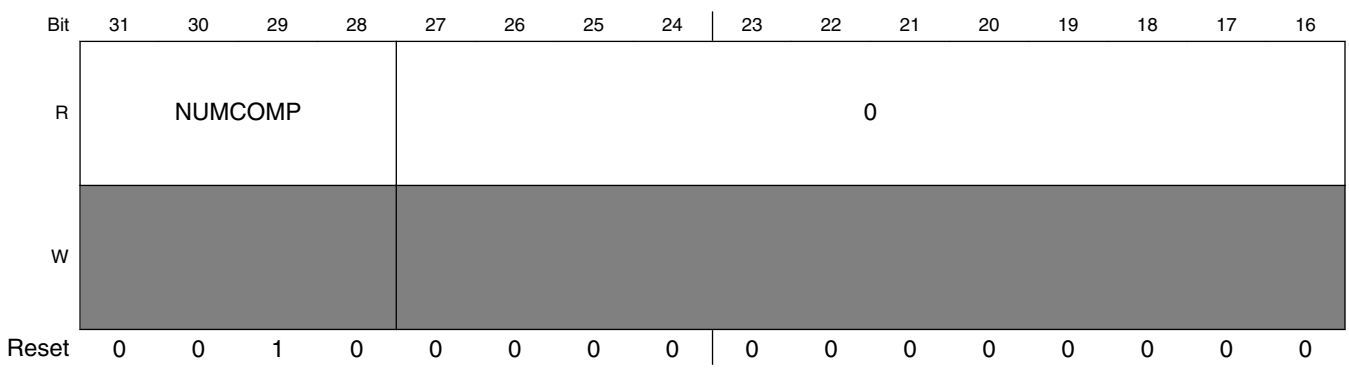
Field	Description
	0 No match. 1 Match occurred.
23–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
FUNCTION	Function Selects the action taken on a comparator match. If MTBDWT_COMP0 is used for a data value and MTBDWT_COMP1 for an address value, then MTBDWT_FCT1[FUNCTION] must be set to zero. For this configuration, MTBDWT_MASK1 can be set to a non-zero value, so the combined comparators match on a range of addresses. 0000 Disabled. 0100 Instruction fetch. 0101 Data operand read. 0110 Data operand write. 0111 Data operand (read + write). others Reserved. Any attempts to use this value results in UNPREDICTABLE behavior.

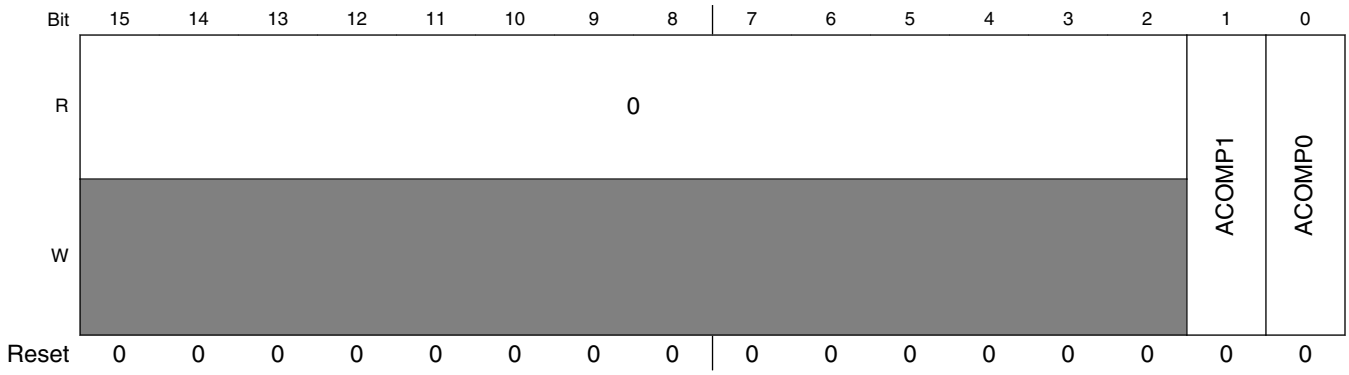
32.3.2.6 MTB_DWT Trace Buffer Control Register (MTB_DWT_TBCTRL)

The MTBDWT_TBCTRL register defines how the watchpoint comparisons control the actual trace buffer operation.

Recall the MTB supports starting and stopping the program trace based on the watchpoint comparisons signaled via TSTART and TSTOP. The watchpoint comparison signals are enabled in the MTB's control logic by setting the appropriate enable bits, MTB_MASTER[TSTARTEN, TSTOPEN]. In the event of simultaneous assertion of both TSTART and TSTOP, TSTART takes priority.

Address: F000_1000h base + 200h offset = F000_1200h





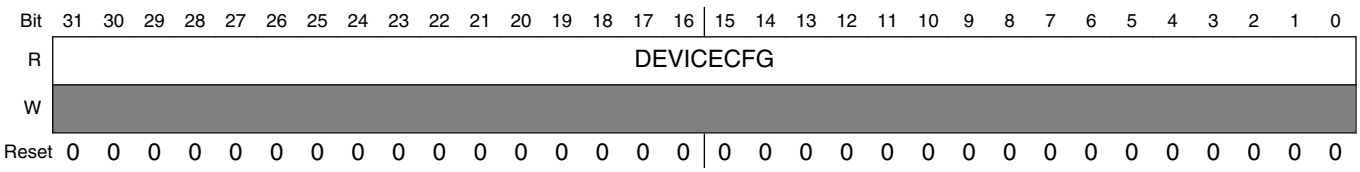
MTB_DWT_TBCTRL field descriptions

Field	Description
31–28 NUMCOMP	Number of Comparators This read-only field specifies the number of comparators in the MTB_DWT. This implementation includes two registers.
27–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 ACOMP1	Action based on Comparator 1 match When the MTBDWT_FCT1[MATCHED] is set, it indicates MTBDWT_COMP1 address compare has triggered and the trace buffer's recording state is changed. 0 Trigger TSTOP based on the assertion of MTBDWT_FCT1[MATCHED]. 1 Trigger TSTART based on the assertion of MTBDWT_FCT1[MATCHED].
0 ACOMP0	Action based on Comparator 0 match When the MTBDWT_FCT0[MATCHED] is set, it indicates MTBDWT_COMP0 address compare has triggered and the trace buffer's recording state is changed. The assertion of MTBDWT_FCT0[MATCHED] is caused by the following conditions: • Address match in MTBDWT_COMP0 when MTBDWT_FCT0[DATAVMATCH] = 0 • Data match in MTBDWT_COMP0 when MTBDWT_FCT0[DATAVMATCH, DATAVADDR0] = {1,0} • Data match in MTBDWT_COMP0 and address match in MTBDWT_COMP1 when MTBDWT_FCT0[DATAVMATCH, DATAVADDR0] = {1,1} 0 Trigger TSTOP based on the assertion of MTBDWT_FCT0[MATCHED]. 1 Trigger TSTART based on the assertion of MTBDWT_FCT0[MATCHED].

32.3.2.7 Device Configuration Register (MTB_DWT_DEVICECFG)

This register indicates the device configuration. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_1000h base + FC8h offset = F000_1FC8h



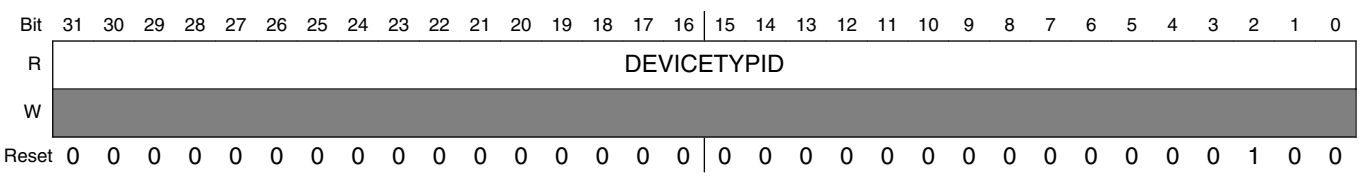
MTB_DWT_DEVICECFG field descriptions

Field	Description
DEVICECFG	DEVICECFG Hardwired to 0x0000_0000.

32.3.2.8 Device Type Identifier Register (MTB_DWT_DEVICETYPID)

This register indicates the device type ID. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_1000h base + FCCh offset = F000_1FCCh



MTB_DWT_DEVICETYPID field descriptions

Field	Description
DEVICETYPID	DEVICETYPID Hardwired to 0x0000_0004.

32.3.2.9 Peripheral ID Register (MTB_DWT_PERIPHID*n*)

These registers indicate the peripheral IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_1000h base + FD0h offset + (4d × 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	PERIPHID																															
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*

MTB_DWT_PERIPHID*n* field descriptions

Field	Description
PERIPHID	PERIPHID Peripheral ID1 is hardwired to 0x0000_00E0; ID2 to 0x0000_0008; and all the others to 0x0000_0000.

32.3.2.10 Component ID Register (MTB_DWT_COMPID*n*)

These registers indicate the component IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_1000h base + FF0h 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	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COMPID																															
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*

MTB_DWT_COMPID*n* field descriptions

Field	Description
COMPID	Component ID Component ID0 is hardwired to 0x0000_000D; ID1 to 0x0000_0090; ID2 to 0x0000_0005; ID3 to 0x0000_00B1.

32.3.3 System ROM Memory Map

The System ROM Table registers are also mapped into a sparsely-populated 4 KB address space.

For core configurations like that supported by Cortex-M0+, ARM recommends that a debugger identifies and connects to the debug components using the CoreSight debug infrastructure.

ARM recommends that a debugger follows the flow as shown in the following figure to discover the components in the CoreSight debug infrastructure. In this case, a debugger reads the peripheral and component ID registers for each CoreSight component in the CoreSight system.

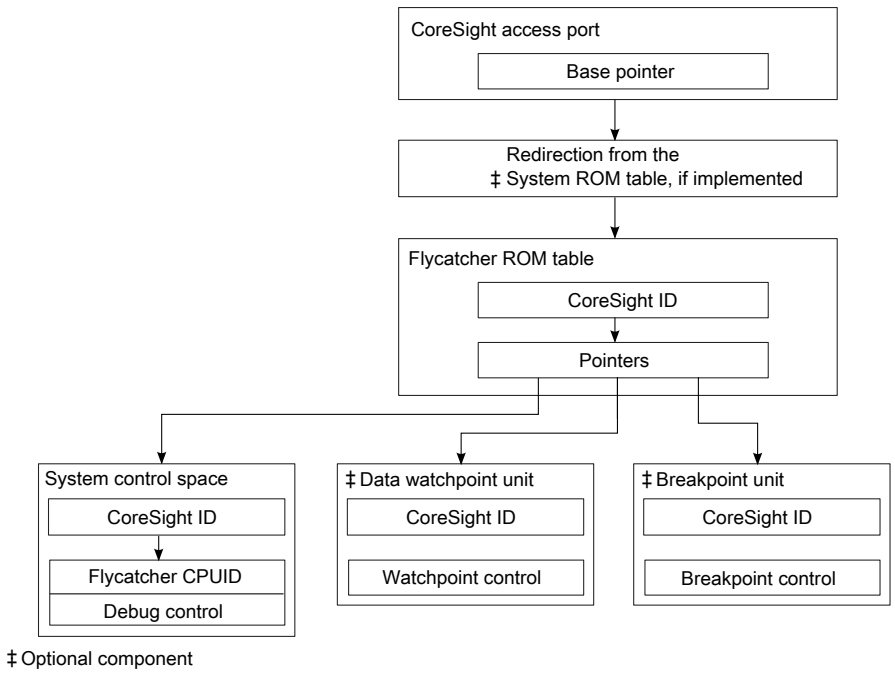


Figure 32-3. CoreSight discovery process

ROM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F000_2000	Entry (ROM_ENTRY0)	32	R	See section	32.3.3.1/ 696
F000_2004	Entry (ROM_ENTRY1)	32	R	See section	32.3.3.1/ 696
F000_2008	Entry (ROM_ENTRY2)	32	R	See section	32.3.3.1/ 696
F000_200C	End of Table Marker Register (ROM_TABLEMARK)	32	R	0000_0000h	32.3.3.2/ 697
F000_2FCC	System Access Register (ROM_SYSACCESS)	32	R	0000_0001h	32.3.3.3/ 697
F000_2FD0	Peripheral ID Register (ROM_PERIPHID4)	32	R	See section	32.3.3.4/ 698

Table continues on the next page...

ROM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F000_2FD4	Peripheral ID Register (ROM_PERIPHID5)	32	R	See section	32.3.3.4/ 698
F000_2FD8	Peripheral ID Register (ROM_PERIPHID6)	32	R	See section	32.3.3.4/ 698
F000_2FDC	Peripheral ID Register (ROM_PERIPHID7)	32	R	See section	32.3.3.4/ 698
F000_2FE0	Peripheral ID Register (ROM_PERIPHID0)	32	R	See section	32.3.3.4/ 698
F000_2FE4	Peripheral ID Register (ROM_PERIPHID1)	32	R	See section	32.3.3.4/ 698
F000_2FE8	Peripheral ID Register (ROM_PERIPHID2)	32	R	See section	32.3.3.4/ 698
F000_2FEC	Peripheral ID Register (ROM_PERIPHID3)	32	R	See section	32.3.3.4/ 698
F000_2FF0	Component ID Register (ROM_COMPID0)	32	R	See section	32.3.3.5/ 698
F000_2FF4	Component ID Register (ROM_COMPID1)	32	R	See section	32.3.3.5/ 698
F000_2FF8	Component ID Register (ROM_COMPID2)	32	R	See section	32.3.3.5/ 698
F000_2FFC	Component ID Register (ROM_COMPID3)	32	R	See section	32.3.3.5/ 698

32.3.3.1 Entry (ROM_ENTRY*n*)

The System ROM Table begins with "n" relative 32-bit addresses, one for each debug component present in the device and terminating with an all-zero value signaling the end of the table at the "n+1"-th value.

It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_2000h base + 0h 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	ENTRY																															
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*

ROM_ENTRY*n* field descriptions

Field	Description
ENTRY	ENTRY

ROM_ENTRY*n* field descriptions (continued)

Field	Description
	Entry 0 (MTB) is hardwired to 0xFFFF_E003; Entry 1 (MTBDWT) to 0xFFFF_F003; Entry 2 (CM0+ ROM Table) to 0xF00F_D003.

32.3.3.2 End of Table Marker Register (ROM_TABLEMARK)

This register indicates end of table marker. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_2000h base + Ch offset = F000_200Ch

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	MARK																															
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

ROM_TABLEMARK field descriptions

Field	Description
MARK	MARK Hardwired to 0x0000_0000

32.3.3.3 System Access Register (ROM_SYSACCESS)

This register indicates system access. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_2000h base + FCCh offset = F000_2FCCh

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	SYSACCESS																															
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	1

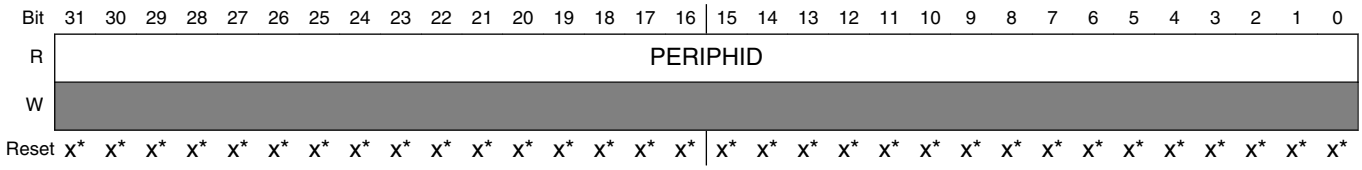
ROM_SYSACCESS field descriptions

Field	Description
SYSACCESS	SYSACCESS Hardwired to 0x0000_0001

32.3.3.4 Peripheral ID Register (ROM_PERIPHID*n*)

These registers indicate the peripheral IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_2000h base + FD0h offset + (4d × i), where i=0d to 7d



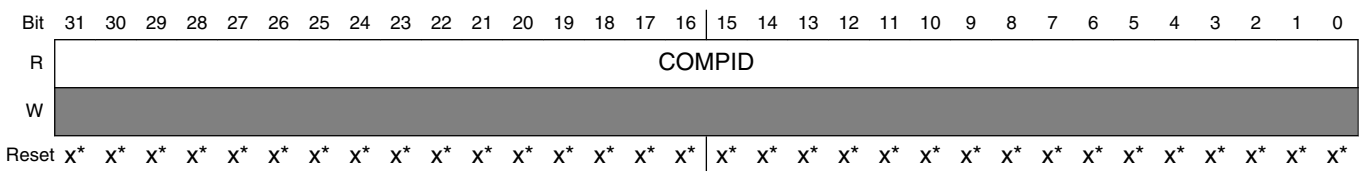
ROM_PERIPHID*n* field descriptions

Field	Description
PERIPHID	PERIPHID Peripheral ID1 is hardwired to 0x0000_00E0; ID2 to 0x0000_0008; and all the others to 0x0000_0000.

32.3.3.5 Component ID Register (ROM_COMPID*n*)

These registers indicate the component IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_2000h base + FF0h offset + (4d × i), where i=0d to 3d



ROM_COMPID*n* field descriptions

Field	Description
COMPID	Component ID Component ID0 is hardwired to 0x0000_000D; ID1 to 0x0000_0010; ID2 to 0x0000_0005; ID3 to 0x0000_00B1.

Chapter 33
Flash Memory Controller (FMC)

33.1 Chip-specific FMC information

33.1.1 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-Light Switch Configuration for details on the master port assignments.

33.2 Introduction

The Flash Memory Controller (FMC) is a memory acceleration unit. A list of features provided by the FMC can be found here.

- an interface between bus masters and the 64-bit program flash memory.
- a buffer and a cache that can accelerate program flash memory data transfers.

33.2.1 Overview

The Flash Memory Controller manages the interface between bus masters and the 64-bit program flash memory. The FMC receives status information detailing the configuration of the flash memory and uses this information to ensure a proper interface. The FMC supports 8-bit, 16-bit, and 32-bit read operations from the program flash memory. A write operation to program flash memory results in a bus error.

In addition, the FMC provides two separate mechanisms for accelerating the interface between bus masters and program flash memory. A 64-bit speculation buffer can prefetch the next 64-bit flash memory location, and a 4-way, 4-set program flash memory cache can store previously accessed program flash memory data for quick access times.

33.2.2 Features

The features of FMC module include:

- Interface between bus masters and the 64-bit program flash memory:
  - 8-bit, 16-bit, and 32-bit read operations to nonvolatile flash memory.
- Acceleration of data transfer from the program flash memory to the device:
  - 64-bit prefetch speculation buffer for program flash accesses with controls for instruction/data access
  - 4-way, 4-set, 64-bit line size program flash memory cache for a total of sixteen 64-bit entries with invalidation control

33.3 Modes of operation

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

In terms of chip power modes:

- The FMC operates only in Run and Wait modes, including VLPR and VLPW modes.
- For any power mode where the program flash memory cannot be accessed, the FMC is disabled.

33.4 External signal description

The FMC has no external (off-chip) signals.

33.5 Memory map and register descriptions

The MCM's programming model provides control and configuration of the FMC's features.

For details, see the description of the MCM's Platform Control Register (PLACR).

33.6 Functional description

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

Besides managing the interface between bus masters and the program flash memory, the FMC can be used to customize the program flash memory cache and buffer to provide single-cycle system clock data access times. Whenever a hit occurs for the prefetch speculation buffer or the cache (when enabled), the requested data is transferred within a single system clock.

Upon system reset, the FMC is configured as follows:

- Flash cache is enabled.
- Instruction speculation and caching are enabled.
- Data speculation is disabled.
- Data caching is enabled.

Though the default configuration provides flash acceleration, advanced users may desire to customize the FMC buffer configurations to maximize throughput for their use cases. For example, the user may adjust the controls to enable buffering per access type (data or instruction).

NOTE

When reconfiguring the FMC, do not program the control and configuration inputs to the FMC while the program flash memory is being accessed. Instead, change them with a routine executing from RAM in supervisor mode.

33.6.1 Flash Access Control (FAC) Function

The Flash Access Control (FAC) is a configurable memory protection scheme optimized to allow end users to use software libraries while offering programmable restrictions to these libraries. The flash memory is divided into *equal size segments* that provide protection to proprietary software libraries. The protection of these segments is controlled: the FAC provides a cycle-by-cycle evaluation of the access rights for each transaction routed to the on-chip flash memory. Two levels of vendors can add their proprietary software to a device; FAC protection of segments for each level are defined once, using the PGMONCE command.

Flash access control aligns to the 3 privilege levels supported by ARM Cortex-M family products:

- Most secure state is supervisor/privileged secure: allows execute-only and provides supervisor-only access control.
- Mid-level state is execute-only.
- Unsecure state is where no access control states are set.

Features:

- Lightweight access control logic for on-chip flash memory
- Flash address space divided into (32 or 64) equal-sized segments (segment size is defined as flash_size [bytes]/(32 or 64))
- Separate control bits for supervisor-only access and execute-only access per segment
- Access control evaluated on each bus cycle routed to the flash
- Access violation errors terminate the bus cycle and return zeroes for read data
- Programming model allows 2 levels of protected segments

33.6.1.1 FAC functional description

The access control functionality is implemented in 2 separate blocks within the SoC. The Flash Management Unit (FMU) includes non-volatile configuration information that is retrieved during reset and and sent to the platform to control access to the flash array during normal operation.

There are (4) 64-bit NVM storage locations to support access control features. These NVM locations are summarized in the table below.

Table 33-1. NVM Locations

NVM location	Description	
NVSACC1, NVSACC2	Two locations are ANDed together and loaded during reset into the *x*_SACC register to provide access configuration.	Segment-wise control for supervisor-only access vs. supervisor and user access
NVXACC1, NVXACC2	Two locations are ANDed together and loaded during reset into the *x*_XACC register to provide access configuration.	Segment-wise control for execute-only vs. data and execute

Each of these NVM locations is programmable through a Program Once flash command and can be programmed one time. These NVM locations are unaffected by Erase All Blocks flash command and debug interface initiated mass erase operations. Since the 2 NVXACCx fields are ANDed, the access protection can only be increased. A segment's access controls can be changed from data read and execute (XA*n* =1) to execute-only (XA*n* =0), or from supervisor and user mode (SAn = 1) to supervisor-only mode (SAn = 0).

The flash is released from reset early while the core continues to be held in reset. The FMU captures the NVM access control information in internal registers. The FMU ANDs the multiple execute-only fields to create a single execute-only field. This execute-only field driven to the platform is static prior to the core being released from reset. The supervisor-only field is handled in the same manner.

The FMU includes the FAC registers that provide control access to the flash address space. During the address phase of every attempted flash transfer, the supervisor access (SA*n*) and execute access (XA*n*) bits are examined to either allow or deny access. If access is denied, then the access is aborted and terminates with a bus error; the read data is also zeroed.

The next table shows segment assignments relative to the flash location.

Table 33-2. Flash Protection Ranges

SA*n* and XA*n* Bit	Protected Segment Address Range	Segment Size (Fraction of total Flash)
64 Segment Encodings		
0	0x0_0000_0000 – (Flash_size/64-1)	1/64
1	(Flash_size/64) – 2*(Flash_size/64-1)	1/64
.....		
63	63*(Flash_size/64) – 62*(Flash_size/64-1)	1/64
32 Segment Encodings		
0	0x0_0000_0000 – (Flash_size/32-1)	1/32
1	(Flash_size/32) – 2*(Flash_size/32-1)	1/32
.....		
31	31*(Flash_size/32) – 30*(Flash_size/32-1)	1/32

Individual segments within the flash memory can be independently protected from user access and data access. Protection is controlled by the individual bits within the *x*_SACC and *x*_XACC registers, as shown in the next figure.

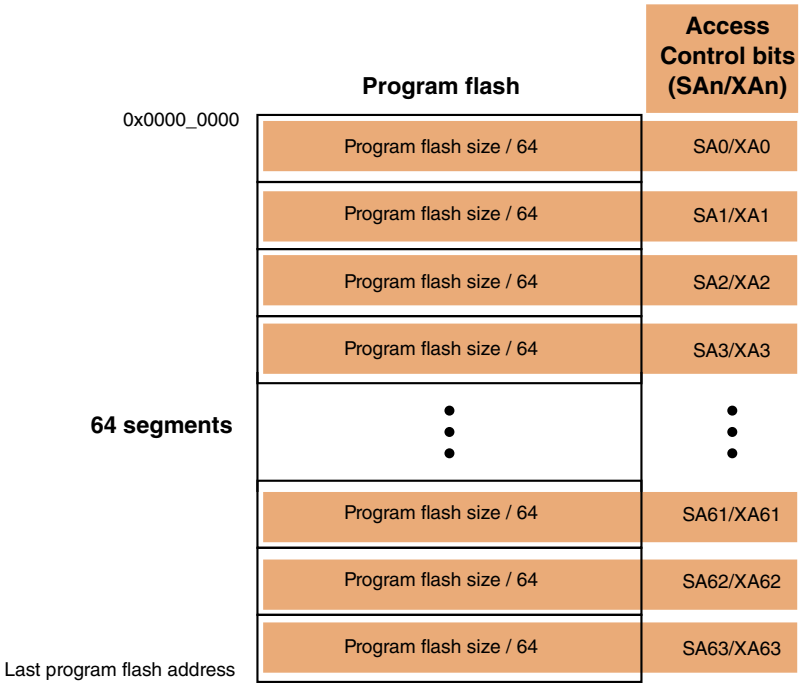


Figure 33-1. Program flash protection (64 segments)

33.6.1.1.1 Interface Signals

Table 33-3. Interface Signals

Signal	Width	From	To	Description
xacc	64 or 32	FMU	Platform	Direct xacc (execute-only access control) register
sacc	64 or 32	FMU	Platform	Direct sacc (supervisor access control) register
numsg	8	FMU	Platform	NUMSG bit field - Binary encoded number of segments 0x40 for 64 segments 0x20 for 32 segments
fac_enable	1	SIM	FMU	SIM Option bit - derived from an IFR bit and captured in SIM_SOPTx. A way to disable the flash access control for phantom devices without this feature. fac_enable==1 - Access Control feature is enabled fac_enable==0 - Access Control feature is disabled • During the reset sequence, XACC registers are written to all "1"s. • During the reset sequence, SACC registers are written to all 1"s. • Implied protection based on XACC registers is turned off.

33.6.1.1.2 Flash Command Impact

Program Longword/Phrase/ Section	If the targeted flash location is in an execute-only protected segment, then these program commands are not allowed unless a Read 1s All Blocks command is executed and returns with a pass code (which means the part has been fully erased). After the Read 1s All Blocks command is executed with a pass code returned, then the protected segment is open to program commands. To close off programmability to execute-only spaces once again, the device must be reset or a Read 1s All Blocks command is executed with a fail result. Attempts to program in a protected segment *when not open to program commands* causes a Protection Violation flag.
PGMCHK	The FMU will not execute the PGMCHK on a segment that has been configured as execute-only. The Flash Protection Violation flag is set if an attempt is made to execute PGMCHK command on an execute-only address.
Erase Flash Sector	If the targeted flash sector is in an execute-only protected segment, then the Erase Flash Sector command is not allowed, and sets the Protection Violation flag. The only means of erasing protected space is by an Erase All operation.
ERSALL	The Erase All Blocks command is not affected by Access Control. An Erase All Blocks command will erase any libraries that have been programmed in any execute-only segment. The programmed execute-only assignment is not erased as part of the Erase All Blocks command, and access control regions remain as previously programmed. **NOTE:** The ERSALL command may be used for field upgrades. Access control states remain programmed. Software must plan accordingly, possibly making extra space available for future use.
SWAP	A new control has been added to the SWAP command to disable the SWAP feature. The IFR SWAP Field and the SWAP indicator are erased during the Erase All Blocks command operation, resulting in the SWAP system being uninitialized. The SWAP command must be run with the initialization code to set the SWAP indicator address and initialize the SWAP system. After being disabled, SWAP cannot be enabled without doing an Erase All. An Erase All erases preloaded code and libraries in the flash array and resets the SWAP system back to uninitialized, but leaves the access controls as previously programmed. If SWAP is intended to be disabled as part of the access control protection, then the disabled setting must be restored after an Erase All Blocks operation.

33.6.1.1.3 Core Platform Impact

Platform core caches (Flash and LMEM caches)	If any segment is marked as *execute-only*, then the caches are hidden from the user. The tag is read-only and cannot be written, and the data caches cannot be read or written. Writes to the tag and data arrays are ignored, and reads of the data array return 0's. This will impact debug breakpoints. See the debug section for details.
Debug	The debugger is a non-processor bus master and cannot step, trace or break in execute-only regions. In supervisor-only mode, the debugger is restricted from changing modes. Debug accesses to any segment of flash space marked as execute-only also terminate with a bus error.
PC-relative addressing	The PC-relative addressing issue is still being understood and this section will be updated in the future. PC relative re-entry to execute-only segments will be allowed..... Restrictions will be placed on software for PC relative addressing, because hardware cannot determine if PC relative data references are crossing segment boundaries.

Table continues on the next page...

	• If ifetch is executing in a protected segment, then data references will be allowed. • Hardware cannot track speculative ifetches across boundaries.
Interrupts	If function calls are used to move into an execute-only segment, then this can be tracked by hardware when typical software controls are used (i.e., saving registers and states before executing new code).
Reset Vector	In the ARM core, the reset vector fetch is supervisor data, which poses issues if the reset vector is located in a segment marked execute-only. Additional logic has been implemented to allow supervisor data fetches to execute-only spaces, after reset until the first valid instruction fetch. After the first valid instruction fetch, the FAC logic follows normal checks.

33.6.1.1.4 Software Impact

As implementation, verification and validation continue, there will be more details on software impact that will need to be communicated to tool and library vendors. The hardware cannot see all states of the ARM core and cannot track the software flow, and may require software restrictions to work with the hardware for a robust solution.

- **Any segment marked as execute-only can see all code in the system.** This means that one execute-only segment can read the execute-only code in another segment. Therefore, if we at the factory are sending pre-loaded code to another vendor, then that vendor will have access to our factory code. NDAs and legal agreements might help deal with this issue.
- **For single pre-loads** (for example, if we at the factory are pre-loading for a general purpose (GP) market or if a vendor with a blank part is pre-loading their proprietary code), then both levels of access control must be programmed, to protect the pre-loaded code.
- **If any portion of a protected segment is not used by pre-loaded code**, then it (the portion of a protected segment that is not used by pre-loaded code) should be programmed with NOPs, to prevent additional code from being programmed in that segment by hackers.

33.6.1.1.5 Access Check Evaluation

The flash controller FAC provides a cycle-by-cycle evaluation of the access rights for each data transaction routed to the on-chip flash memory.

The entire flash storage capacity is partitioned into equal sized segments. Two registers include a supervisor-only access control indicator and a execute-only access control indicator for each segment.

The FAC logic performs the required access control evaluation using the reference address and a 2-bit attribute (or "protection" field) as inputs from the bus cycle plus the contents of the programming model registers.

The following code example illustrates C code for FAC evaluation:

```
unsigned long long sacc;  // supervisor-only map
unsigned long long xacc;  // execute-only map
unsigned int seg_size;    // 8-bit segment size
unsigned int fac_error;

fac_evaluation (addr, prot)
  unsigned int addr;  // access address
  unsigned int hprot; // encoded 2-bit "protection" field {supv, data}
{
  unsigned int sacc_flag; // sacc flag for this segment
  unsigned int xacc_flag; // xacc flag for this segment
  unsigned int i;         // segment index

  i         = (addr >> (8 + seg_size & 0x0f)) & 0x3f; // form 6-bit segment index
  sacc_flag = (sacc >> i) & 1;              // extract sacc bit for this segment
  xacc_flag = (xacc >> i) & 1;              // extract xacc bit for this segment

  // create a 4-tuple concatenating the 2-bit protection field + {sacc, xacc} flags

  switch ((hprot & 3) << 2 | (sacc_flag << 1) | xacc_flag) {
  // all these combinations are allowed accesses
    case 0x2: // {user, ifetch} && {supv+user, ifetch-only}
    case 0x3: // {user, ifetch} && {supv+user, ifetch+data}
    case 0x7: // {user, data}   && {supv+user, ifetch+data}
    case 0x8: // {supv, ifetch} && {supv-only, ifetch-only}
    case 0x9: // {supv, ifetch} && {supv-only, ifetch+data}
    case 0xa: // {supv, ifetch} && {supv+user, ifetch-only}
    case 0xb: // {supv, ifetch} && {supv+user, ifetch+data}
    case 0xd: // {supv, data}   && {supv-only, ifetch+data}
    case 0xf: // {supv, data}   && {supv+user, ifetch+data}
      fac_error = 00;
      break;

  // all these combinations are unallowed, that is, errored accesses
    case 0x0: // {user, ifetch} && {supv-only, ifetch-only}
    case 0x1: // {user, ifetch} && {supv-only, ifetch+data}
    case 0x4: // {user, data}   && {supv-only, ifetch-only}
    case 0x5: // {user, data}   && {supv-only, ifetch+data}
    case 0x6: // {user, data}   && {supv+user, ifetch-only}
    case 0xc: // {supv, data}   && {supv-only, ifetch-only}
    case 0xe: // {supv, data}   && {supv+user, ifetch-only}
      fac_error = 1;
      break;

   } // switch()

} // fac_evaluation()
```

33.6.1.1.6 FAC application tips

In one use case, the NVSACC1 and NVXACC1 locations are programmed by NXP and they protect NXP libraries that have been programmed into associated flash segments in a device. Later, the NVSACC2 and NVXACC2 NVM locations can optionally be programmed by a third-party vendor who wants to program their proprietary software and to extend the protection of protected flash segments to include their software libraries before supplying it all to their customers.

Their customer would then develop their own code to use the available libraries, and program their code into the remaining available on-chip flash. The device continues to support the end user with standard security features that further limit external access to flash resources.

SWAP: If execute-only code is mirrored in both halves of the flash array, then SWAP can be enabled without any issues; otherwise SWAP should be disabled, because hardware does not track access control addressing during SWAP.

Chapter 34
Flash Memory Module (FTFA)

34.1 Chip-specific FTFA information

34.1.1 Erasable Program Once Field descriptions

On this device, the Program Once Field in the program flash erasable IFR with indexes 0x20 - 0x23 provides 16 bytes of user data storage that can be programmed using the Program Once command but cannot be read using the Read Once command. During reset, the data stored inside the Program Once Field with indexes 0x20 – 0x23 will be loaded into the SIM_SECKEY0 - SIM_SECKEY3 registers. If the flash is in secure state, these registers are cleared once read. If the flash is in unsecure state, these registers are always read as “0”.

34.2 Introduction

The flash memory module includes the following accessible memory regions:

- Program flash memory for vector space and code store

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 flash memory 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.

34.2.1 Features

The flash memory module includes the following features.

NOTE

See the device's Chip Configuration details for the exact amount of flash memory available on your device.

34.2.1.1 Program Flash Memory Features

- Sector size of 2 KB
- Program flash protection scheme prevents accidental program or erase of stored data
- Program flash access control scheme prevents unauthorized access to selected code segments
- Automated, built-in, program and erase algorithms with verify

34.2.1.2 Other Flash Memory 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

34.2.2 Block Diagram

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

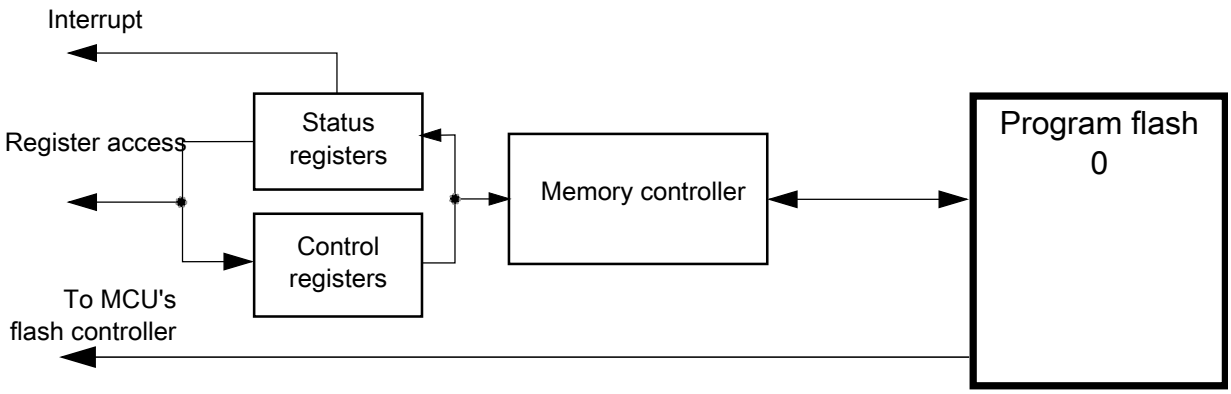


Figure 34-1. Flash Block Diagram

34.2.3 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 flash memory module.

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 flash memory module.

Flash block — A macro within the flash memory module which provides the nonvolatile memory storage.

Flash Memory Module — All flash blocks plus a flash management unit providing high-level control and an interface to MCU buses.

HSRUN — An MCU power mode enabling high-speed access to the memory resources in the flash module. The user has no access to the flash command set when the MCU is in HSRUN mode.

IFR — Nonvolatile information register found in each flash block, separate from the main memory array.

Longword — 32 bits of data with an aligned longword having byte-address[1:0] = 00.

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 flash memory module 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 flash memory module.

NVM Special Mode — An NVM mode enabling external, off-chip access to the memory resources in the flash memory 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.

Phrase — 64 bits of data with an aligned phrase having byte-address[2:0] = 000.

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.

Secure — An MCU state conveyed to the flash memory module as described in the Chip Configuration details for this device. In the secure state, reading and changing NVM contents is restricted.

Word — 16 bits of data with an aligned word having byte-address[0] = 0.

34.3 External Signal Description

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

34.4 Memory Map and Registers

This section describes the memory map and registers for the flash memory module.

Data read from unimplemented memory space in the flash memory module is undefined. Writes to unimplemented or reserved memory space (registers) in the flash memory module are ignored.

34.4.1 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 flash memory module.

Flash Configuration Field Offset Address	Size (Bytes)	Field Description
0x0_0400–0x0_0407	8	Backdoor Comparison Key. Refer to Verify Backdoor Access Key Command and Unsecuring the Chip 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	Reserved
0x0_040E	1	Reserved
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).

34.4.2 Program Flash IFR Map

The program flash IFR is 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 IFR are summarized in the table found here and further described in the subsequent paragraphs.

The program flash IFR is located within the program flash 0 memory block .

Address Range	Size (Bytes)	Field Description
0x00 – 0x9F	160	Reserved
0xA0 – 0xA3	4	Program Once XACCH-1 Field (index = 0x10)
0xA4 – 0xA7	4	Program Once XACCL-1 Field (index = 0x10)
0xA8 – 0xAB	4	Program Once XACCH-2 Field (index = 0x11)
0xAC – 0xAF	4	Program Once XACCL-2 Field (index = 0x11)
0xB0 – 0xB3	4	Program Once SACCH-1 Field (index = 0x12)
0xB4 – 0xB7	4	Program Once SACCL-1 Field (index = 0x12)
0xB8 – 0xBB	4	Program Once SACCH-2 Field (index = 0x13)
0xBC – 0xBF	4	Program Once SACCL-2 Field (index = 0x13)
0xC0 – 0xFF	64	Program Once ID Field (index = 0x00 - 0x0F)

34.4.2.1 Program Once Field

The Program Once Field in the program flash IFR provides 96 bytes of user data storage separate from the program flash 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 IFR is accessed in 4-byte or 8-Byte records using the Read Once and Program Once commands (see Read Once Command and Program Once Command).

34.4.3 Program Flash Erasable IFR Map

The program flash erasable IFR is nonvolatile information memory that can be erased but has limited program capabilities and limited read access. The contents of the program flash erasable IFR fields are summarized in the table found here.

Address Range	Size (Bytes)	Field Description
0x80 – 0x83	4	Erasable Program Once Field (index = 0x20)
0x84 – 0x87	4	Erasable Program Once Field (index = 0x21)
0x88 – 0x8B	4	Erasable Program Once Field (index = 0x22)
0x8C – 0x8F	4	Erasable Program Once Field (index = 0x23)
0xC0 – 0xC3	4	Erasable Program Once Field (index = 0x30)
0xC4 – 0xC7	4	Erasable Program Once Field (index = 0x31)
0xC8 – 0xCB	4	Erasable Program Once Field (index = 0x32)
0xCC – 0xCF	4	Erasable Program Once Field (index = 0x33)

34.4.3.1 Erasable Program Once Field

The Program Once Field in the program flash erasable IFR with indexes 0x20 - 0x23 provides 16 bytes of user data storage that can be programmed using the Program Once command but cannot be read using the Read Once command. The Program Once Field in the program flash erasable IFR with indexes 0x30 - 0x33 provides 16 bytes of user data storage that can be programmed using the Program Once command and read using the Read Once command. The program flash erasable IFR is erased using the Erase All Blocks command, Erase All Blocks Unsecure command, and using the external erase all feature (mass erase).

34.4.4 Register Descriptions

The flash memory 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).

FTFA memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_0000	Flash Status Register (FTFA_FSTAT)	8	R/W	00h	34.4.4.1/ 717
4002_0001	Flash Configuration Register (FTFA_FCNFG)	8	R/W	00h	34.4.4.2/ 719
4002_0002	Flash Security Register (FTFA_FSEC)	8	R	Undefined	34.4.4.3/ 720
4002_0003	Flash Option Register (FTFA_FOPT)	8	R	Undefined	34.4.4.4/ 721
4002_0004	Flash Common Command Object Registers (FTFA_FCCOB3)	8	R/W	00h	34.4.4.5/ 722
4002_0005	Flash Common Command Object Registers (FTFA_FCCOB2)	8	R/W	00h	34.4.4.5/ 722
4002_0006	Flash Common Command Object Registers (FTFA_FCCOB1)	8	R/W	00h	34.4.4.5/ 722
4002_0007	Flash Common Command Object Registers (FTFA_FCCOB0)	8	R/W	00h	34.4.4.5/ 722
4002_0008	Flash Common Command Object Registers (FTFA_FCCOB7)	8	R/W	00h	34.4.4.5/ 722
4002_0009	Flash Common Command Object Registers (FTFA_FCCOB6)	8	R/W	00h	34.4.4.5/ 722
4002_000A	Flash Common Command Object Registers (FTFA_FCCOB5)	8	R/W	00h	34.4.4.5/ 722
4002_000B	Flash Common Command Object Registers (FTFA_FCCOB4)	8	R/W	00h	34.4.4.5/ 722
4002_000C	Flash Common Command Object Registers (FTFA_FCCOBB)	8	R/W	00h	34.4.4.5/ 722
4002_000D	Flash Common Command Object Registers (FTFA_FCCOBA)	8	R/W	00h	34.4.4.5/ 722
4002_000E	Flash Common Command Object Registers (FTFA_FCCOB9)	8	R/W	00h	34.4.4.5/ 722
4002_000F	Flash Common Command Object Registers (FTFA_FCCOB8)	8	R/W	00h	34.4.4.5/ 722
4002_0010	Program Flash Protection Registers (FTFA_FPROT3)	8	R/W	Undefined	34.4.4.6/ 723
4002_0011	Program Flash Protection Registers (FTFA_FPROT2)	8	R/W	Undefined	34.4.4.6/ 723
4002_0012	Program Flash Protection Registers (FTFA_FPROT1)	8	R/W	Undefined	34.4.4.6/ 723
4002_0013	Program Flash Protection Registers (FTFA_FPROT0)	8	R/W	Undefined	34.4.4.6/ 723

Table continues on the next page...

FTFA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_0018	Execute-only Access Registers (FTFA_XACCH3)	8	R	Undefined	34.4.4.7/ 725
4002_0019	Execute-only Access Registers (FTFA_XACCH2)	8	R	Undefined	34.4.4.7/ 725
4002_001A	Execute-only Access Registers (FTFA_XACCH1)	8	R	Undefined	34.4.4.7/ 725
4002_001B	Execute-only Access Registers (FTFA_XACCH0)	8	R	Undefined	34.4.4.7/ 725
4002_001C	Execute-only Access Registers (FTFA_XACCL3)	8	R	Undefined	34.4.4.7/ 725
4002_001D	Execute-only Access Registers (FTFA_XACCL2)	8	R	Undefined	34.4.4.7/ 725
4002_001E	Execute-only Access Registers (FTFA_XACCL1)	8	R	Undefined	34.4.4.7/ 725
4002_001F	Execute-only Access Registers (FTFA_XACCL0)	8	R	Undefined	34.4.4.7/ 725
4002_0020	Supervisor-only Access Registers (FTFA_SACCH3)	8	R	Undefined	34.4.4.8/ 726
4002_0021	Supervisor-only Access Registers (FTFA_SACCH2)	8	R	Undefined	34.4.4.8/ 726
4002_0022	Supervisor-only Access Registers (FTFA_SACCH1)	8	R	Undefined	34.4.4.8/ 726
4002_0023	Supervisor-only Access Registers (FTFA_SACCH0)	8	R	Undefined	34.4.4.8/ 726
4002_0024	Supervisor-only Access Registers (FTFA_SACCL3)	8	R	Undefined	34.4.4.8/ 726
4002_0025	Supervisor-only Access Registers (FTFA_SACCL2)	8	R	Undefined	34.4.4.8/ 726
4002_0026	Supervisor-only Access Registers (FTFA_SACCL1)	8	R	Undefined	34.4.4.8/ 726
4002_0027	Supervisor-only Access Registers (FTFA_SACCL0)	8	R	Undefined	34.4.4.8/ 726
4002_0028	Flash Access Segment Size Register (FTFA_FACSS)	8	R	Undefined	34.4.4.9/ 727
4002_002B	Flash Access Segment Number Register (FTFA_FACSN)	8	R	Undefined	34.4.4.10/ 728

34.4.4.1 Flash Status Register (FTFA_FSTAT)

The FSTAT register reports the operational status of the flash memory 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 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

FTFA_FSTAT field descriptions

Field	Description
7 CCIF	Command Complete Interrupt Flag Indicates that a flash command 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. CCIF 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. 0 Flash command in progress 1 Flash command has completed
6 RDCOLERR	Flash Read Collision Error Flag Indicates that the MCU attempted a read from a flash memory resource that was being manipulated by a flash 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. 0 No collision error detected 1 Collision error detected
5 ACCERR	Flash Access Error Flag Indicates an illegal access has occurred to a flash memory resource caused by a violation of the command write sequence or issuing an illegal flash 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 ACCERR while CCIF is set. Writing a 0 to the ACCERR bit has no effect. 0 No access error detected 1 Access error detected
4 FPVIOL	Flash Protection Violation Flag Indicates an attempt was made to program or erase an address in a protected area of program flash memory during a command write sequence . While FPVIOL is set, the CCIF flag cannot be cleared to launch a command. The FPVIOL bit is cleared by writing a 1 to FPVIOL while CCIF is set. Writing a 0 to the FPVIOL bit has no effect. 0 No protection violation detected 1 Protection violation detected

Table continues on the next page...

FTFA_FSTAT field descriptions (continued)

Field	Description
3–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 MGSTAT0	Memory Controller Command Completion Status Flag The MGSTAT0 status flag is set if an error is detected during execution of a flash command or during the flash reset sequence. As a status flag, this field cannot (and need not) be cleared by the user like the other error flags in this register. 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.

34.4.4.2 Flash Configuration Register (FTFA_FCNFG)

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

The erase control bits (ERSAREQ and ERSSUSP) have write restrictions. The unassigned bits read as noted and are not writable.

Address: 4002_0000h base + 1h offset = 4002_0001h

Bit	7	6	5	4	3	2	1	0
Read	CCIE	RDCOLLIE	ERSAREQ	ERSSUSP	0	0	0	0
Write								
Reset	0	0	0	0	0	0	0	0

FTFA_FCNFG field descriptions

Field	Description
7 CCIE	Command Complete Interrupt Enable Controls interrupt generation when a flash command completes. 0 Command complete interrupt disabled 1 Command complete interrupt enabled. An interrupt request is generated whenever the FSTAT[CCIF] flag is set.
6 RDCOLLIE	Read Collision Error Interrupt Enable Controls interrupt generation when a flash memory read collision error occurs. 0 Read collision error interrupt disabled 1 Read collision error interrupt enabled. An interrupt request is generated whenever a flash memory read collision error is detected (see the description of FSTAT[RDCOLERR]).
5 ERSAREQ	Erase All Request

Table continues on the next page...

FTFA_FCNFG field descriptions (continued)

Field	Description
	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. ERSAREQ sets when an erase all request is triggered external to the flash memory module and CCIF is set (no command is currently being executed). ERSAREQ is cleared by the flash memory module when the operation completes. 0 No request or request complete 1 Request to: 1. run the Erase All Blocks command, 2. verify the erased state, 3. program the security byte in the Flash Configuration Field to the unsecure state, and 4. release MCU security by setting the FSEC[SEC] field to the unsecure state.
4 ERSSUSP	Erase Suspend Allows the user to suspend (interrupt) the Erase Flash Sector command while it is executing. 0 No suspend requested 1 Suspend the current Erase Flash Sector command execution.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

34.4.4.3 Flash Security Register (FTFA_FSEC)

This read-only register holds all bits associated with the security of the MCU and flash memory 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.

FTFA_FSEC field descriptions

Field	Description
7–6 KEYEN	Backdoor Key Security Enable Enables or disables backdoor key access to the flash memory module. 00 Backdoor key access disabled 01 Backdoor key access disabled (preferred KEYEN state to disable backdoor key access) 10 Backdoor key access enabled 11 Backdoor key access disabled
5–4 MEEN	Mass Erase Enable Enables and disables mass erase capability of the flash memory module. The state of this field is relevant only when SEC is set to secure. When SEC is set to unsecure, the MEEN setting does not matter. 00 Mass erase is enabled 01 Mass erase is enabled 10 Mass erase is disabled 11 Mass erase is enabled
3–2 FSLACC	Freescale Failure Analysis Access Code Enables or disables 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. 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 SEC is set to secure. When SEC is set to unsecure, the FSLACC setting does not matter. 00 Freescale factory access granted 01 Freescale factory access denied 10 Freescale factory access denied 11 Freescale factory access granted
SEC	Flash Security Defines the security state of the MCU. In the secure state, the MCU limits access to flash memory module resources. The limitations are defined per device and are detailed in the Chip Configuration details. If the flash memory module is unsecured using backdoor key access, SEC is forced to 10b. 00 MCU security status is secure. 01 MCU security status is secure. 10 MCU security status is unsecure. (The standard shipping condition of the flash memory module is unsecure.) 11 MCU security status is secure.

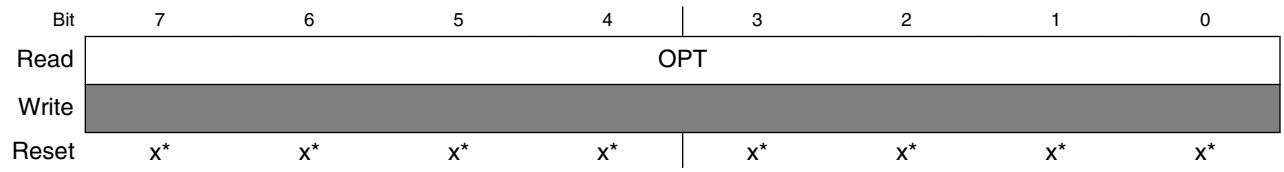
34.4.4.4 Flash Option Register (FTFA_FOPT)

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. However, the register is written to 0xFF if the contents of the flash nonvolatile option byte are 0x00.

Address: 4002_0000h base + 3h offset = 4002_0003h



* Notes:
- x = Undefined at reset.

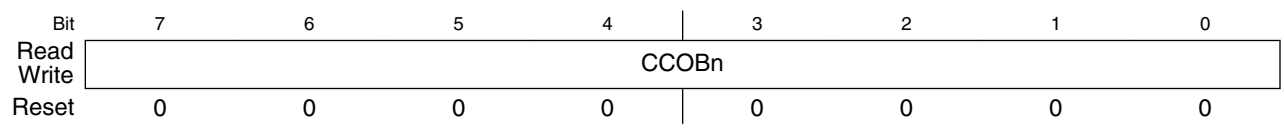
FTFA_FOPT field descriptions

Field	Description
OPT	Nonvolatile Option 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.

34.4.4.5 Flash Common Command Object Registers (FTFA_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



FTFA_FCCOB*n* field descriptions

Field	Description
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.

FTFA_FCCOB*n* field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
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.

The following table shows a generic flash 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 flash command, typically an address and/or data values.

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.

<table>
<tr><th>FCCOB Number</th><th>Typical Command Parameter Contents [7:0]</th></tr>
<tr><td>0</td><td>FCMD (a code that defines the flash command)</td></tr>
<tr><td>1</td><td>Flash address [23:16]</td></tr>
<tr><td>2</td><td>Flash address [15:8]</td></tr>
<tr><td>3</td><td>Flash address [7:0]</td></tr>
<tr><td>4</td><td>Data Byte 0</td></tr>
<tr><td>5</td><td>Data Byte 1</td></tr>
<tr><td>6</td><td>Data Byte 2</td></tr>
<tr><td>7</td><td>Data Byte 3</td></tr>
<tr><td>8</td><td>Data Byte 4</td></tr>
<tr><td>9</td><td>Data Byte 5</td></tr>
<tr><td>A</td><td>Data Byte 6</td></tr>
<tr><td>B</td><td>Data Byte 7</td></tr>
</table>

FCCOB Endianness and Multi-Byte Access :

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).
</td></tr>
</table>

34.4.4.6 Program Flash Protection Registers (FTFA_FPROT*n*)

The FPROT registers define which 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 flash command. Unprotected regions can be changed by program and erase operations.

The four FPROT registers allow up to 32 protectable regions. Each bit protects a 1/32 region of the program flash memory except for memory configurations with less than 64 KB of program flash where each assigned bit protects 2 KB . For configurations with 48 KB of program flash memory or less, FPROT0 is not used. For configurations with 32

KB of program flash memory or less, FPROT1 is not used. For configurations with 16 KB of program flash memory, FPROT2 is not used. The bitfields are defined in each register as follows:

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 Write	PROT							
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FTFA_FPROT*n* field descriptions

Field	Description
PROT	Program Flash Region Protect Each program flash region can be protected from program and erase operations by setting the associated PROT bit. **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.

FTFA_FPROT*n* field descriptions (continued)

Field	Description
	In NVM Special mode: All bits of FPROT are writable without restriction. Unprotected areas can be protected and protected areas can be unprotected. **Restriction:** The user must never write to any FPROT register while a command is running (CCIF=0). 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. Each bit in the 32-bit protection register represents 1/32 of the total program flash except for memory configurations with less than 64 KB of program flash where each assigned bit protects 2 KB . 0 Program flash region is protected. 1 Program flash region is not protected

34.4.4.7 Execute-only Access Registers (FTFA_XACC*n*)

The XACC registers define which program flash segments are restricted to data read or execute only or both data and instruction fetches.

The eight XACC registers allow up to 64 restricted segments of equal memory size.

Execute-only access register	Program flash execute-only access bits
XACCH0	XA[63:56]
XACCH1	XA[55:48]
XACCH2	XA[47:40]
XACCH3	XA[39:32]
XACCL0	XA[31:24]
XACCL1	XA[23:16]
XACCL2	XA[15:8]
XACCL3	XA[7:0]

During the reset sequence, the XACC registers are loaded with the logical AND of Program Flash IFR addresses A and B as indicated in the following table.

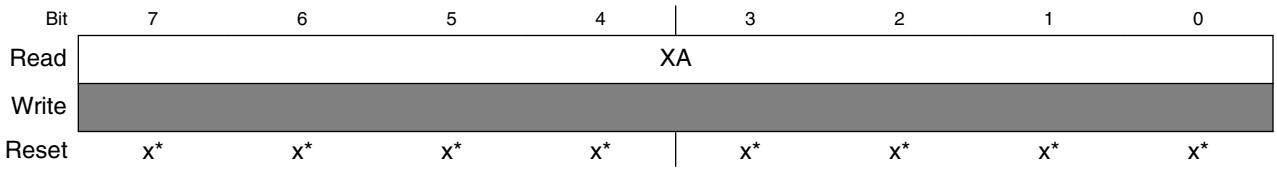
Execute-only access register	Program Flash IFR address A	Program Flash IFR address B
XACCH0	0xA3	0xAB
XACCH1	0xA2	0xAA
XACCH2	0xA1	0xA9
XACCH3	0xA0	0xA8
XACCL0	0xA7	0xAF
XACCL1	0xA6	0xAE
XACCL2	0xA5	0xAD

Table continues on the next page...

Execute-only access register	Program Flash IFR address A	Program Flash IFR address B
XACCL3	0xA4	0xAC

Use the Program Once command to program the execute-only access control fields that are loaded during the reset sequence.

Address: 4002_0000h base + 18h offset + (1d × i), where i=0d to 7d



* Notes:
- x = Undefined at reset.

FTFA_XACC*n* field descriptions

Field	Description
XA	Execute-only access control 0 Associated segment is accessible in execute mode only (as an instruction fetch) 1 Associated segment is accessible as data or in execute mode

34.4.4.8 Supervisor-only Access Registers (FTFA_SACC*n*)

The SACC registers define which program flash segments are restricted to supervisor only or user and supervisor access.

The eight SACC registers allow up to 64 restricted segments of equal memory size.

Supervisor-only access register	Program flash supervisor-only access bits
SACCH0	SA[63:56]
SACCH1	SA[55:48]
SACCH2	SA[47:40]
SACCH3	SA[39:32]
SACCL0	SA[31:24]
SACCL1	SA[23:16]
SACCL2	SA[15:8]
SACCL3	SA[7:0]

During the reset sequence, the SACC registers are loaded with the logical AND of Program Flash IFR addresses A and B as indicated in the following table.

Supervisor-only access register	Program Flash IFR address A	Program Flash IFR address B
SACCH0	0xB3	0xBB
SACCH1	0xB2	0xBA
SACCH2	0xB1	0xB9
SACCH3	0xB0	0xB8
SACCL0	0xB7	0xBF
SACCL1	0xB6	0xBE
SACCL2	0xB5	0xBD
SACCL3	0xB4	0xBC

Use the Program Once command to program the supervisor-only access control fields that are loaded during the reset sequence.

Address: 4002_0000h base + 20h offset + (1d × i), where i=0d to 7d

Bit	7	6	5	4	3	2	1	0
Read	SA							
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FTFA_SACC*n* field descriptions

Field	Description
SA	Supervisor-only access control 0 Associated segment is accessible in supervisor mode only 1 Associated segment is accessible in user or supervisor mode

34.4.4.9 Flash Access Segment Size Register (FTFA_FACSS)

The flash access segment size register determines which bits in the address are used to index into the SACC and XACC bitmaps to get the appropriate permission flags.

All bits in the register are read-only.

The contents of this register are loaded during the reset sequence.

Address: 4002_0000h base + 28h offset = 4002_0028h

Bit	7	6	5	4	3	2	1	0
Read	SGSIZE							
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FTFA_FACSS field descriptions

Field	Description
SGSIZE	Segment Size The segment size is a fixed value based on the available program flash size divided by NUMSG.

Program Flash Size	Segment Size	Segment Size Encoding
64 KBytes	2 KBytes	0x3
128 KBytes	4 KBytes	0x4
160 KBytes	4 KBytes	0x4
256 KBytes	4 KBytes	0x4
512 KBytes	8 KBytes	0x5

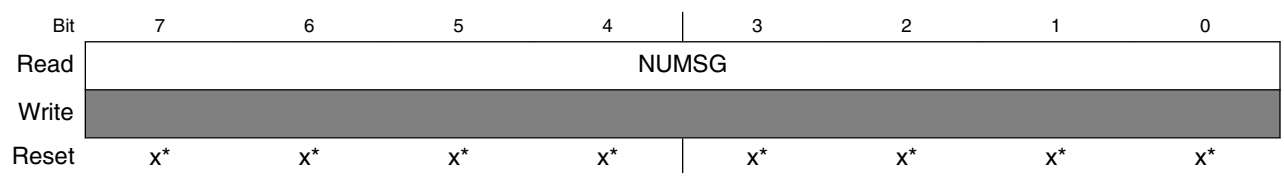
34.4.4.10 Flash Access Segment Number Register (FTFA_FACSN)

The flash access segment number register provides the number of program flash segments that are available for XACC and SACC permissions.

All bits in the register are read-only.

The contents of this register are loaded during the reset sequence.

Address: 4002_0000h base + 2Bh offset = 4002_002Bh



* Notes:
- x = Undefined at reset.

FTFA_FACSN field descriptions

Field	Description
NUMSG	Number of Segments Indicator The NUMSG field indicates the number of equal-sized segments in the program flash. 0x20 Program flash memory is divided into 32 segments (64 Kbytes, 128 Kbytes) 0x28 Program flash memory is divided into 40 segments (160 Kbytes) 0x40 Program flash memory is divided into 64 segments (256 Kbytes, 512 Kbytes)

FTFA_FACSN field descriptions (continued)

Field	Description

34.5 Functional Description

The information found here describes functional details of the flash memory module.

34.5.1 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* —
  - For 2^n program flash sizes, four registers typically protect 32 regions of the program flash memory as shown in the following figure

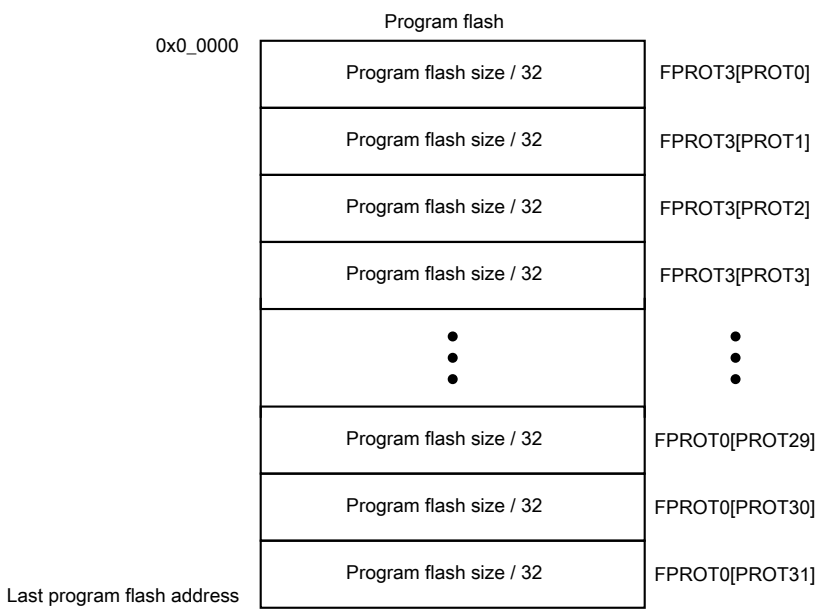


Figure 34-2. Program flash protection

NOTE

Flash protection features are discussed further in AN4507: Using the Kinetis Security and Flash Protection Features . Not all features described in the application note are available on this device.

34.5.2 Flash Access Protection

Individual segments within the program flash memory can be designated for restricted access. Specific flash commands (Program Check, Program Longword, Erase Flash Sector) monitor FXACC contents to protect flash memory but the FSACC contents do not impact flash command operation. Access is controlled by the following registers:

- FTFA_XACC —
  - For 2^n program flash sizes 128KB or less, four registers control 32 segments of the program flash memory as shown in the following figure

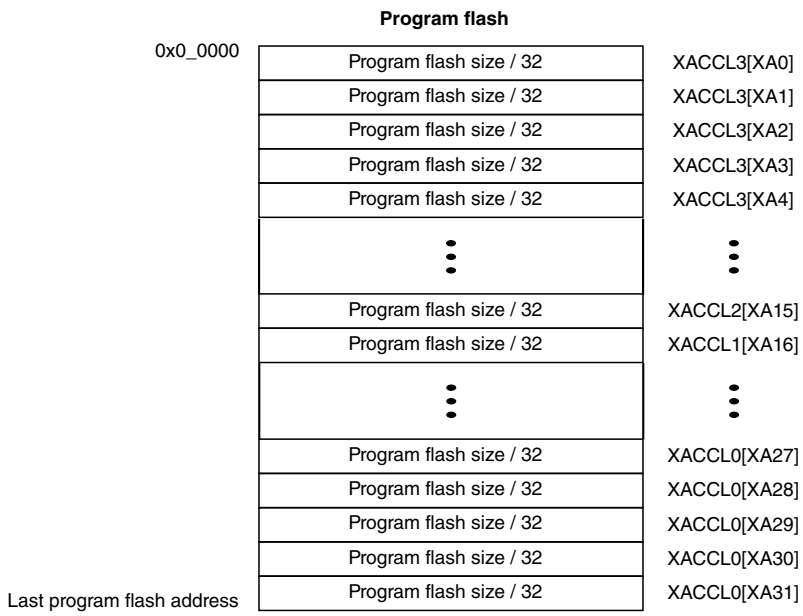


Figure 34-3. Program flash access control (64KB or 128KB of program flash)

- FTFA_SACC —
  - For 2^n program flash sizes 128KB or less, four registers control 32 segments of the program flash memory as shown in the following figure

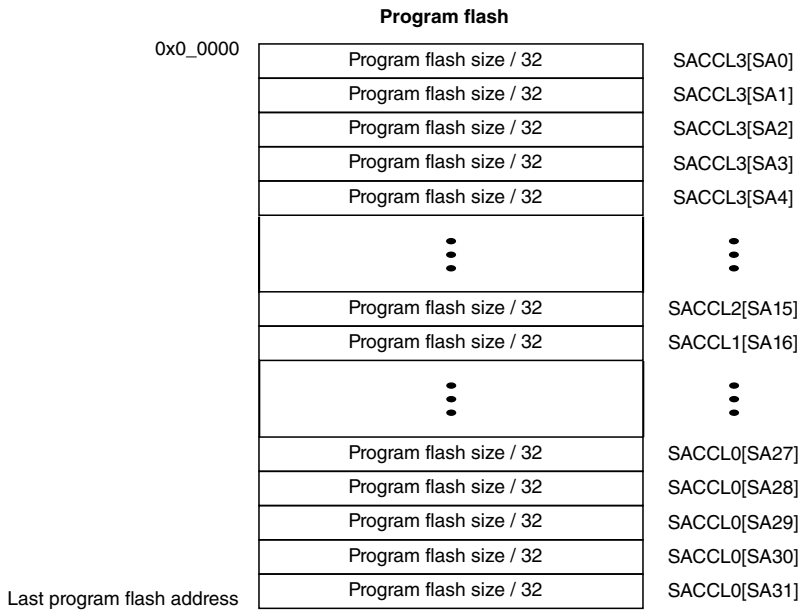


Figure 34-4. Program flash access control (64KB or 128KB of program flash)

34.5.3 Interrupts

The flash memory module can generate interrupt requests to the MCU upon the occurrence of various flash events.

These interrupt events and their associated status and control bits are shown in the following table.

Table 34-1. Flash Interrupt Sources

Flash Event	Readable Status Bit	Interrupt Enable Bit
Flash Command Complete	FSTAT[CCIF]	FCNFG[CCIE]
Flash Read Collision Error	FSTAT[RDCOLERR]	FCNFG[RDCOLLIE]

Note

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

Some devices also generate a bus error response as a result of a Read Collision Error event. See the chip configuration information to determine if a bus error response is also supported.

34.5.4 Flash Access Control (FAC) function

See the section Flash Access Control (FAC) Function.

34.5.5 Flash Operation in Low-Power Modes

34.5.5.1 Wait Mode

When the MCU enters wait mode, the flash memory module is not affected. The flash memory module can recover the MCU from wait via the command complete interrupt (see Interrupts).

34.5.5.2 Stop Mode

When the MCU requests stop mode, if a flash 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 flash command is running (CCIF = 0).

NOTE

While the MCU is in very-low-power modes (VLPR, VLPW, VLPS), the flash memory module does not accept flash commands.

34.5.6 Functional Modes of Operation

The flash memory module has two operating modes: NVM Normal and NVM Special.

The operating mode affects the command set availability (see Table 34-2). Refer to the Chip Configuration details of this device for how to activate each mode.

34.5.7 Flash Reads and Ignored Writes

The flash memory module requires only the flash address to execute a flash memory read.

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).

34.5.8 Read While Write (RWW)

The following simultaneous accesses are not allowed:

- Reading from program flash memory space while a flash command is active (CCIF=0).

34.5.9 Flash Program and Erase

All flash functions except read require the user to setup and launch a flash command through a series of peripheral bus writes.

The user cannot initiate any further flash commands until notified that the current command has completed. The flash command structure and operation are detailed in Flash Command Operations.

34.5.10 Flash Command Operations

Flash command operations are typically used to modify flash memory contents.

The next sections describe:

- The command write sequence used to set flash command parameters and launch execution
- A description of all flash commands available

34.5.10.1 Command Write Sequence

Flash commands are specified using a command write sequence illustrated in Figure 34-5. The flash memory 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 a flash command in VLP mode will be ignored. Attempts to launch a flash command in HSRUN mode will be trapped with the ACCERR flag being set.

34.5.10.1.1 Load the FCCOB Registers

The user must load the FCCOB registers with all parameters required by the desired flash command. The individual registers that make up the FCCOB data set can be written in any order.

34.5.10.1.2 Launch the Command by Clearing CCIF

Once all relevant command parameters have been loaded, the user launches the command by clearing FSTAT[CCIF] by writing a '1' to it. FSTAT[CCIF] remains 0 until the flash command completes.

The FSTAT register contains a blocking mechanism that prevents a new command from launching (can't clear FSTAT[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.

34.5.10.1.3 Command Execution and Error Reporting

The command processing has several steps:

1. The flash memory module 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. FSTAT[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, 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 FSTAT[CCIF].

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 FSTAT[MGSTAT0]. 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 applicable, are reported back to the user via the FCCOB and FSTAT registers.
4. The flash memory module sets FSTAT[CCIF] signifying that the command has completed.

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

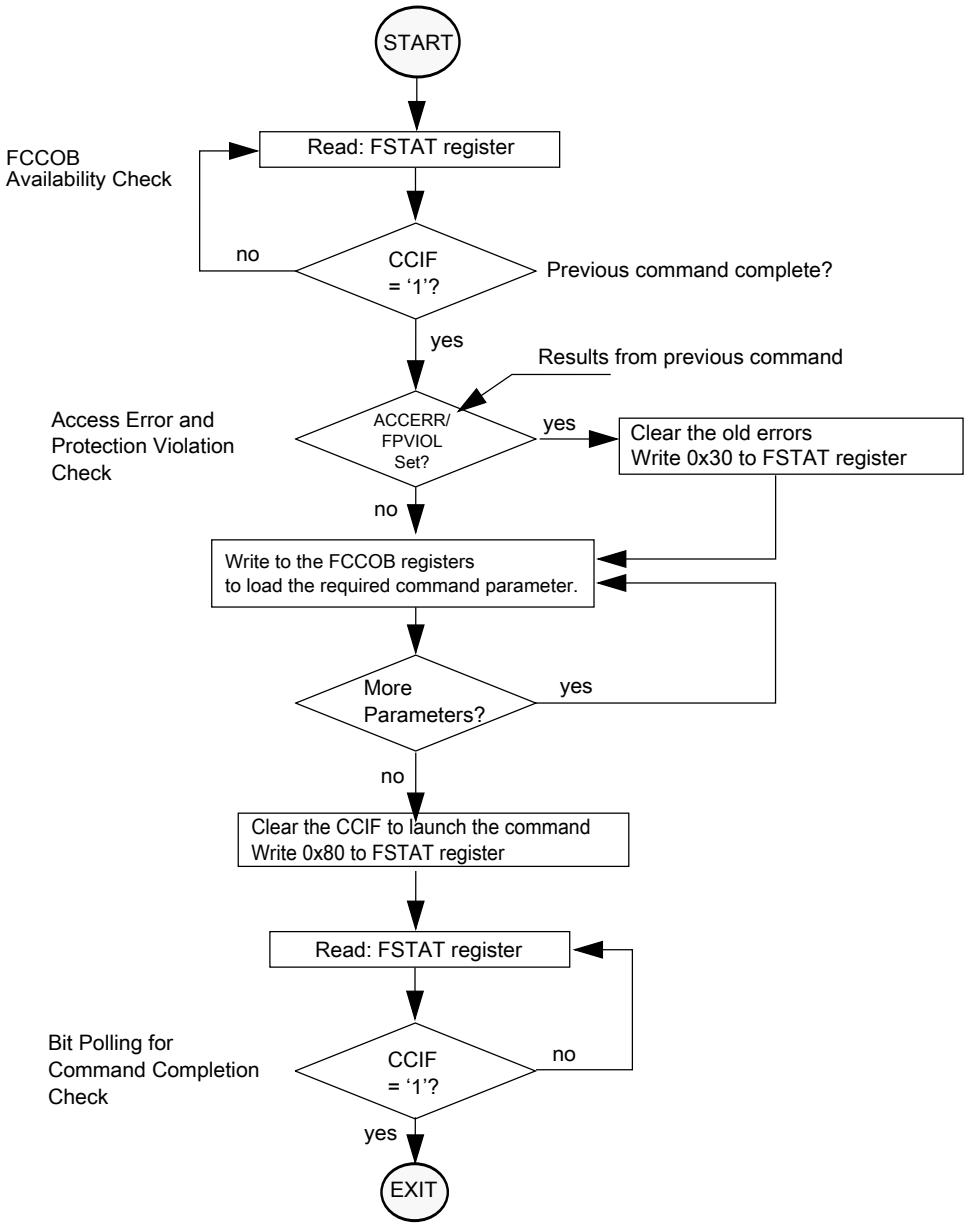


Figure 34-5. Generic flash command write sequence flowchart

34.5.10.2 Flash Commands

The following table summarizes the function of all flash commands.

FCMD	Command	Program flash	Function
0x01	Read 1s Section	×	Verify that a given number of program flash locations from a starting address are erased.

Table continues on the next page...

FCMD	Command	Program flash	Function
0x02	Program Check	×	Tests previously-programmed locations at margin read levels.
0x03	Read Resource	IFR, ID	Read 4 bytes from program flash IFR or version ID.
0x06	Program Longword	×	Program 4 bytes in a program flash block.
0x09	Erase Flash Sector	×	Erase all bytes in a program flash sector.
0x40	Read 1s All Blocks	×	Verify that the program flash block is erased then release MCU security.
0x41	Read Once	IFR	Read 4 bytes of a dedicated 64 byte field in the program flash 0 IFR.
0x43	Program Once	IFR	One-time program of 4 bytes of a dedicated 64-byte field in the program flash 0 IFR.
0x44	Erase All Blocks	×	Erase the program flash block, verify-erase and release MCU security. **NOTE:** An erase is only possible when all memory locations are unprotected.
0x45	Verify Backdoor Access Key	×	Release MCU security after comparing a set of user-supplied security keys to those stored in the program flash.
0x49	Erase All Blocks Unsecure	×	Erase the program flash block, verify-erase, program security byte to unsecure state, release MCU security.
0x4A	Read 1s All Execute-only Segments	×	Verify that all program flash execute-only (XA) segments are erased then release flash access control.
0x4B	Erase All Execute-only Segments	×	Erase all program flash execute-only (XA) segments then release flash access control.

34.5.10.3 Flash Commands by Mode

The following table shows the flash commands that can be executed in each flash operating mode.

Table 34-2. Flash Commands by Mode

FCMD	Command	NVM Normal			NVM Special		
		Unsecure	Secure	MEEN=10	Unsecure	Secure	MEEN=10
0x01	Read 1s Section	×	×	×	×	—	—
0x02	Program Check	×	×	×	×	—	—
0x03	Read Resource	×	×	×	×	—	—
0x06	Program Longword	×	×	×	×	—	—
0x09	Erase Flash Sector	×	×	×	×	—	—
0x40	Read 1s All Blocks	×	×	×	×	×	—
0x41	Read Once	×	×	×	×	—	—
0x43	Program Once	×	×	×	×	—	—
0x44	Erase All Blocks	×	×	×	×	×	—
0x45	Verify Backdoor Access Key	×	×	×	×	—	—
0x49	Erase All Blocks Unsecure	×	×	—	×	×	—
0x4A	Read 1s All Execute-only Segments	×	×	×	×	—	—
0x4B	Erase All Execute-only Segments	×	×	×	×	—	—

34.5.11 Margin Read Commands

The Read-1s commands (Read 1s All Blocks, Read 1s Section, Read 1s All Execute-only Segments) 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 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. Basic flash array reads 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.

34.5.12 Flash Command Description

This section describes all flash commands that can be launched by a command write sequence.

The flash memory module 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 FSTAT[ACCERR] and FSTAT[FPVIOL] 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 flash memory module is running a command (FSTAT[CCIF] = 0) on that same block. The flash memory module may return invalid data to the MCU with the collision error flag (FSTAT[RDCOLERR]) set.

CAUTION

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

34.5.12.1 Read 1s Section Command

The Read 1s Section command checks if a section of program flash memory is erased to the specified read margin level. The Read 1s Section command defines the starting address and the number of phrases to be verified.

Table 34-3. Read 1s Section Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x01 (RD1SEC)
1	Flash address [23:16] of the first phrase to be verified
2	Flash address [15:8] of the first phrase to be verified
3	Flash address [7:0][1] of the first phrase to be verified
4	Number of phrases to be verified [15:8]
5	Number of phrases to be verified [7:0]
6	Read-1 Margin Choice

1. Must be phrase aligned (Flash address [2:0] = 000).

Upon clearing CCIF to launch the Read 1s Section command, the flash memory module sets the read margin for 1s according to Table 34-4 and then reads all locations within the specified section of flash memory. If the flash memory module fails to read all 1s (that is, the flash section is not erased), FSTAT[MGSTAT0] is set. FSTAT[CCIF] sets after the Read 1s Section operation completes.

Table 34-4. 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 34-5. 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]

Table continues on the next page...

Table 34-5. Read 1s Section Command Error Handling (continued)

Error condition	Error bit
Flash address is not phrase aligned.	FSTAT[ACCERR]
The requested section crosses a Flash block boundary.	FSTAT[ACCERR]
The requested number of phrases is 0.	FSTAT[ACCERR]
Read-1s fails.	FSTAT[MGSTAT0]

34.5.12.2 Program Check Command

The Program Check command tests a previously programmed program flash longword to see if it reads correctly at the specified margin level.

Table 34-6. 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 flash memory module sets the read margin for 1s according to Table 34-7, 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, FSTAT[MGSTAT0] is set.

The flash memory module then sets the read margin for 0s, re-reads, and compares again. If the comparison at margin-0 fails, FSTAT[MGSTAT0] is set. FSTAT[CCIF] is set after the Program Check operation completes.

The supplied address must be longword aligned (the lowest two bits of the byte address must be 00):

- Byte 3 data is written to the supplied byte address ('start'),
- Byte 2 data is programmed to byte address start+0b01,
- Byte 1 data is programmed to byte address start+0b10,
- Byte 0 data is programmed to byte address start+0b11.

NOTE

See the description of margin reads, Margin Read Commands

Table 34-7. 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 34-8. 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]
Flash address is located in an XA controlled segment and the Erase All Blocks, Erase All Blocks Unsecure or the Read 1s All Blocks command has not successfully completed since the last reset	FSTAT[FPVIOL]
Either of the margin reads does not match the expected data	FSTAT[MGSTAT0]

34.5.12.3 Read Resource Command

The Read Resource command allows the user to read data from special-purpose memory resources located within the flash memory module. The special-purpose memory resources available include program flash IFR space and the Version ID field. Each resource is assigned a select code as shown in Table 34-10.

Table 34-9. 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]
Returned Values	
4	Read Data [31:24]
5	Read Data [23:16]
6	Read Data [15:8]
7	Read Data [7:0]
User-provided values	
8	Resource Select Code (see Table 34-10)

1. Must be longword aligned (Flash address [1:0] = 00).

Table 34-10. Read Resource Select Codes

Resource Select Code	Description	Resource Size	Local Address Range
0x00	Program Flash 0 IFR	256 Bytes	0x00_0000–0x00_00FF
0x01[1]	Version ID	8 Bytes	0x00_0000–0x00_0007

1. Located in program flash 0 reserved space.

After clearing CCIF to launch the Read Resource command, four consecutive bytes are read from the selected resource at the provided relative address and stored in the FCCOB register. The CCIF flag sets after the Read Resource operation completes. 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 34-11. 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 longword aligned	FSTAT[ACCERR]

34.5.12.4 Program Longword Command

The Program Longword command programs four previously-erased bytes in the program 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 34-12. Program Longword Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x06 (PGM4)
1	Flash address [23:16]
2	Flash address [15:8]

Table continues on the next page...

Table 34-12. Program Longword Command FCCOB Requirements (continued)

FCCOB Number	FCCOB Contents [7:0]
3	Flash address [7:0][1]
4	Byte 0 program value
5	Byte 1 program value
6	Byte 2 program value
7	Byte 3 program value

1. Must be longword aligned (Flash address [1:0] = 00).

Upon clearing CCIF to launch the Program Longword command, the flash memory module programs the data bytes into the flash using the supplied address. The targeted flash locations must be currently unprotected (see the description of the FPROT registers) to permit execution of the Program Longword 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 FSTAT[MGSTAT0]. The CCIF flag is set after the Program Longword operation completes.

The supplied address must be longword aligned (flash address [1:0] = 00):

- Byte 3 data is written to the supplied byte address ('start'),
- Byte 2 data is programmed to byte address start+0b01,
- Byte 1 data is programmed to byte address start+0b10, and
- Byte 0 data is programmed to byte address start+0b11.

Table 34-13. Program Longword 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]
Flash address points to a protected area	FSTAT[FPVIOL]
Flash address is located in an XA controlled segment and the Erase All Blocks, Erase All Blocks Unsecure or the Read 1s All Blocks command has not successfully completed since the last reset	FSTAT[FPVIOL]
Any errors have been encountered during the verify operation	FSTAT[MGSTAT0]

34.5.12.5 Erase Flash Sector Command

The Erase Flash Sector operation erases all addresses in a flash sector.

Table 34-14. 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 phrase aligned (flash address [2:0] = 000).

After clearing CCIF to launch the Erase Flash Sector command, the flash memory module erases the selected program 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). 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 34-6).

Table 34-15. 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 phrase aligned	FSTAT[ACCERR]
The selected program flash sector is protected	FSTAT[FPVIOL]
The selected program flash sector is located in an XA controlled segment and the Erase All Blocks, Erase All Blocks Unsecure or the Read 1s All Blocks command has not successfully completed since the last reset	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.

34.5.12.5.1 Suspending an Erase Flash Sector Operation

To suspend an Erase Flash Sector operation set the FCNFG[ERSSUSP] bit when CCIF, ACCERR, and FPVIOL are 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 memory module samples the state of the ERSSUSP bit at convenient points. If the flash memory module detects that the ERSSUSP bit is set, the Erase Flash Sector operation is suspended and the flash memory module 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 flash memory module detects that a suspend request has been made, the flash memory module clears the ERSSUSP bit prior to setting CCIF. When an Erase Flash Sector operation has been

successfully suspended, the flash memory module 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 flash memory module has acknowledged it.

34.5.12.5.2 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 flash memory module 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 of 4.3 msec 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.

34.5.12.5.3 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 flash memory module starts the new command using the new FCCOB contents.

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.

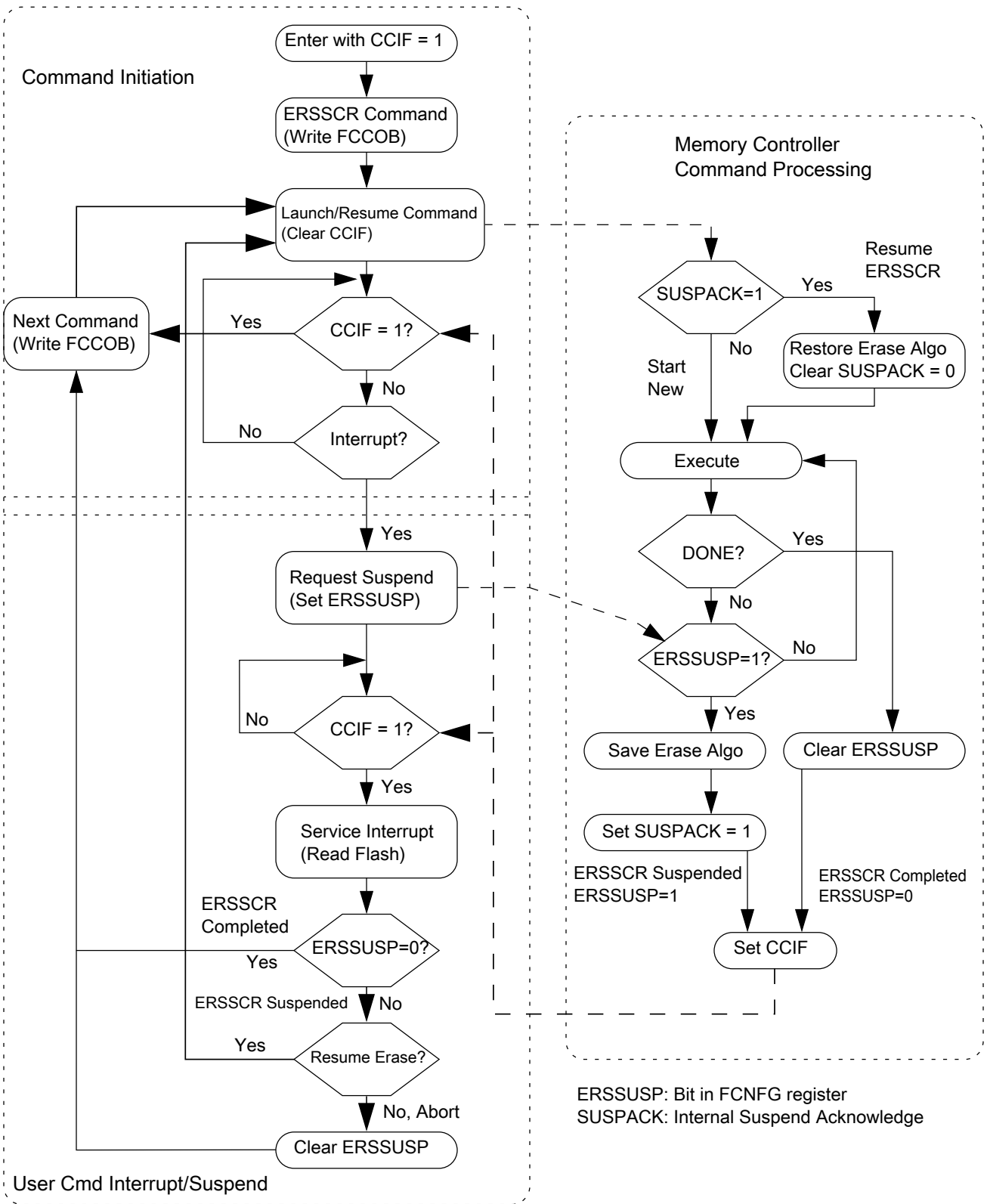


Figure 34-6. Suspend and Resume of Erase Flash Sector Operation

34.5.12.6 Read 1s All Blocks Command

The Read 1s All Blocks command checks if the program flash blocks and program flash erasable 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 34-16. 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 flash memory module :

- sets the read margin for 1s according to Table 34-17,
- checks the contents of the program flash and program flash erasable IFR are in the erased state.

If the flash memory module confirms that these memory resources are erased, access control is disabled and 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 memory resources are not in the fully erased state, the FSTAT[MGSTAT0] bit is set.

The CCIF flag sets after the Read 1s All Blocks operation has completed.

Table 34-17. 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 34-18. Read 1s All Blocks Command Error Handling

Error Condition	Error Bit
An invalid margin choice is specified	FSTAT[ACCERR]
Read-1s fails	FSTAT[MGSTAT0]

34.5.12.7 Read Once Command

The Read Once command provides read access to special 96-byte fields located in the program flash 0 IFR (see Program Flash IFR Map and Program Once Field). Access to the Program Once ID field is via 16 records (index values 0x00 - 0x0F), each 4 bytes long. Access to the Program Once XACC and SACC fields are via 4 records (index values 0x10 - 0x13), each of which is 8 bytes long. These fields are programmed using the Program Once command described in Program Once Command.

The Read Once command also provides read access to special 4-byte records (index values 0x30 - 0x33) located in the program flash erasable IFR (see Program Flash Erasable IFR Map and Erasable Program Once Field). These fields are programmed using the Program Once command.

Table 34-19. Read Once Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x41 (RDONCE)
1	Program Once record index (0x00 - 0x13, 0x30 - 0x33)
2	Not used
3	Not used
Returned Values	
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 (index 0x10 - 0x13)
9	Program Once byte 5 value (index 0x10 - 0x13)
10	Program Once byte 6 value (index 0x10 - 0x13)
11	Program Once byte 7 value (index 0x10 - 0x13)

After clearing CCIF to launch the Read Once command, a 4-byte or 8-byte Program Once record is read 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 - 0x13, 0x30 - 0x33. During execution of the Read Once command, any attempt to read addresses within the program flash block containing the selected record index returns invalid data. The Read Once command can be executed any number of times.

Table 34-20. 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]

34.5.12.8 Program Once Command

The Program Once command enables programming to special 96-byte fields in the program flash 0 IFR (see Program Flash IFR Map and Program Once Field). Access to the Program Once ID field is via 16 records (index values 0x00 - 0x0F), each 4 bytes long. Access to the Program Once XACC and SACC fields are via 4 records (index values 0x10 - 0x13), each of which is 8 bytes long. These records can be read using the Read Once command (see Read Once Command) or using the Read Resource command (see Read Resource Command). These records can be programmed only once since the program flash 0 IFR cannot be erased.

The Program Once command enables programming to special 4-byte records located in the program flash erasable IFR (see Program Flash Erasable IFR Map and Erasable Program Once Field). Record index values 0x20 - 0x23 cannot be read using the Read Once command. Record index values 0x30 - 0x33 can be read using the Read Once command. These records can be reprogrammed since the program flash erasable IFR can be erased using the Erase All Blocks command and Erase All Blocks Unsecure command.

Table 34-21. Program Once Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x43 (PGMONCE)
1	Program Once record index (0x00 - 0x13, 0x20 - 0x23, 0x30 - 0x33)
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 (index 0x10 - 0x13)
9	Program Once byte 5 value (index 0x10 - 0x13)
10	Program Once byte 6 value (index 0x10 - 0x13)
11	Program Once byte 7 value (index 0x10 - 0x13)

After clearing CCIF to launch the Program Once command, the flash memory module 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.

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 - 0x13, 0x20 - 0x23, 0x30 - 0x33. During execution of the Program Once command, any attempt to read addresses within the program flash block containing the selected record index returns invalid data.

Table 34-22. 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-FFFF 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 (0xFFFF_FFFF_FFFF_FFFF for index 0x10 - 0x13), the Program Once command is allowed to execute again on that same record.

34.5.12.9 Erase All Blocks Command

The Erase All Blocks operation erases all flash memory, verifies all memory contents, and releases MCU security.

Table 34-23. 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 flash memory module erases all program flash memory and the program flash erasable IFR space, then verifies that all are erased.

If the flash memory module verifies that all flash memories were properly erased, access control is disabled and security is released by setting the FSEC[SEC] field to the unsecure state. The Erase All Blocks command aborts if any flash region is protected. 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.

Access control determined by the contents of the FXACC registers will not block execution of the Erase All Blocks command. While most Flash memory will be erased, the program flash IFR space containing the Program Once XACC and SACC fields will not be erased and, therefore, the contents of the Program Once XACC and SACC fields will not change. The contents of the FXACC and FSACC registers will not be impacted

by the execution of the Erase All Blocks command. After completion of the Erase All Blocks command, access control is disabled until the next reset of the flash module or the Read 1s All Blocks command is executed and fails (FSTAT[MGSTAT0] is set).

Table 34-24. 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 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.

34.5.12.9.1 Triggering an Erase All External to the Flash Memory Module

The functionality of the Erase All Blocks/Erase All Blocks Unsecure 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 FSTAT[ACCERR and PVIOL] flags must be cleared and the FCCOB0 register must not contain 0x44. When invoked, the erase-all function erases all program flash memory and the program flash erasable IFR space regardless of the protection settings. If the post-erase verify passes, access control determined by the contents of the FXACC registers is disabled and the routine then releases security by setting the FSEC[SEC] field register to the unsecure state. 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, except FPVIOL, as described in Erase All Blocks Command/ Erase All Blocks Unsecure Command.

34.5.12.10 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 (see Flash Configuration Field Description). The column labelled Flash Configuration Field offset address shows the location of the matching byte in the Flash Configuration Field.

Table 34-25. 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_0003
5	Key Byte 1	0x0_0002
6	Key Byte 2	0x0_0001
7	Key Byte 3	0x0_0000
8	Key Byte 4	0x0_0007
9	Key Byte 5	0x0_0006
A	Key Byte 6	0x0_0005
B	Key Byte 7	0x0_0004

After clearing CCIF to launch the Verify Backdoor Access Key command, the flash memory module checks the FSEC[KEYEN] bits to verify that this command is enabled. If not enabled, the flash memory module sets the FSTAT[ACCERR] bit and terminates. If the command is enabled, the flash memory module 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 flash memory module 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 34-26. Verify Backdoor Access Key Command Error Handling

Error Condition	Error Bit
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]

34.5.12.11 Erase All Blocks Unsecure Command

The Erase All Blocks Unsecure operation erases all flash memory, verifies all memory contents, programs the security byte in the Flash Configuration Field to the unsecure state, and releases MCU security.

Table 34-27. Erase All Blocks Unsecure Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x49 (ERSALLU)

After clearing CCIF to launch the Erase All Blocks Unsecure command, the flash memory module erases all program flash memory and the program flash erasable IFR space, then verifies that all are erased.

If the flash memory module verifies that all program flash memory and the program flash erasable IFR space was properly erased, access control is disabled, security is released by setting the FSEC[SEC] field to the unsecure state, and the security byte (see Flash Configuration Field Description) is programmed to the unsecure state by the Erase All Blocks Unsecure command. If the erase or program verify fails, the FSTAT[MGSTAT0] bit is set. The CCIF flag is set after the Erase All Blocks Unsecure operation completes.

Access control determined by the contents of the FXACC registers will not block execution of the Erase All Blocks Unsecure command. While most Flash memory will be erased, the program flash IFR space containing the Program Once XACC and SACC fields will not be erased and, therefore, the contents of the Program Once XACC and SACC fields will not change. The contents of the FXACC and FSACC registers will not be impacted by the execution of the Erase All Blocks Unsecure command. After completion of the Erase All Blocks Unsecure command, access control is disabled until the next reset of the flash module or the Read 1s All Blocks command is executed and fails (FSTAT[MGSTAT0] is set).

Table 34-28. Erase All Blocks Unsecure Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
Any errors have been encountered during erase or program verify operations	FSTAT[MGSTAT0]

34.5.12.12 Read 1s All Execute-only Segments Command

The Read 1s All Execute-only Segments command checks if the program flash execute-only segments defined by the FXACC registers have been erased to the specified read margin level, if applicable, and releases flash access control if the readout passes, i.e. all data reads as '1'.

Table 34-29. Read 1s All Execute-only Segments Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x4A (RD1XA)
1	Read-1 Margin Choice

After clearing CCIF to launch the Read 1s All Execute-only Segments command, the flash memory module :

- sets the read margin for 1s according to Table 34-30,
- checks the contents of the program flash execute-only segments are in the erased state.

If the flash memory module confirms that these segments are erased, flash access control is disabled until the next reset or, after programming any of the execute-only segments, the Read 1s All Execute-only Segments command is executed and fails with the FSTAT[MGSTAT0] bit set. If the read fails, i.e. all segments are not in the fully erased state, the FSTAT[MGSTAT0] bit is set.

The CCIF flag sets after the Read 1s All Execute-only Segments operation has completed.

Table 34-30. Margin Level Choices for Read 1s All Execute-only Segments

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 34-31. Read 1s All Execute-only Segments Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid margin choice is specified	FSTAT[ACCERR]
Read-1s fails	FSTAT[MGSTAT0]

34.5.12.13 Erase All Execute-only Segments Command

The Erase All Execute-only Segments operation erases all program flash execute-only segments defined by the FXACC registers, verifies all segments are erased, and releases flash access control.

Table 34-32. Erase All Execute-only Segments Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x4B (ERSXA)

After clearing CCIF to launch the Erase All Execute-only Segments command, the flash memory module erases all program flash execute-only segments, then verifies that all segments are erased.

If the flash memory module verifies that all segments were properly erased, flash access control is disabled until the next reset or, after programming any of the execute-only segments, the Read 1s All Execute-only Segments command is executed and fails with the FSTAT[MGSTAT0] bit set. The Erase All Execute-only Segments command aborts if any XA controlled segment is protected. If the erase-verify fails, the FSTAT[MGSTAT0] bit is set. The CCIF flag is set after the Erase All Execute-only Segments operation completes.

Access control determined by the contents of the FXACC registers will not block execution of the Erase All Execute-only Segments command. While all XA controlled segments will be erased, the program flash IFR space containing the Program Once XACC fields will not be erased and, therefore, the contents of the Program Once XACC fields will not change. The contents of the FXACC registers will not be impacted by the execution of the Erase All Execute-only Segments command.

Table 34-33. Erase All Execute-only Segments Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
Any XA controlled segment in the program flash memory is protected	FSTAT[FPVIOL]
Any errors have been encountered during the verify operation	FSTAT[MGSTAT0]

34.5.13 Security

The flash memory module provides security information to the MCU based on contents of the FSEC security register.

The MCU then limits access to flash memory resources as defined in the device's Chip Configuration details. During reset, the flash memory 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. The settings are described in the Flash Security Register (FTFA_FSEC) details.

Flash security features are discussed further in AN4507: Using the Kinetis Security and Flash Protection Features . Note that not all features described in the application note are available on this device.

Table 34-34. FSEC register fields

FSEC field	Description
KEYEN	Backdoor Key Access
MEEN	Mass Erase Capability
FSLACC	Factory Security Level Access
SEC	MCU security

34.5.13.1 Flash Memory Access by Mode and Security

The following table summarizes how access to the flash memory module is affected by security and operating mode.

Table 34-35. Flash Memory Access Summary

Operating Mode	Chip Security State	
	Unsecure	Secure
NVM Normal	Full command set	
NVM Special	Full command set	Only the Erase All Blocks, Erase All Blocks Unsecure and Read 1s All Blocks commands.

34.5.13.2 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 chip reset.

34.5.13.2.1 Unsecuring the Chip Using Backdoor Key Access

The chip 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; it 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 chip. The entire 8-byte key cannot be all 0s or all 1s; that is, 0000_0000_0000_0000h and FFFF_FFFF_FFFF_FFFFh 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 chip can be unsecured by the following backdoor key access sequence:

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 chip 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 further use of the Verify Backdoor Access Key command. A reset of the chip 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 chip 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 chip, the security state of the flash memory 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 chip has full control of the contents of the Flash Configuration Field. The chip 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.

34.5.14 Reset Sequence

On each system reset the flash memory module executes a sequence which establishes initial values for the flash block configuration parameters, FPROT, FOPT, FSEC, FXACC, FSACC, and FACNFG registers.

FSTAT[CCIF] is cleared throughout the reset sequence. The flash memory module holds off CPU access during 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 flash 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 35
Quad Serial Peripheral Interface (QuadSPI)

35.1 Chip-specific QuadSPI information

35.1.1 QSPI Module Instantiations

This device contains one QSPI module . This device contains a single dual-QUADSPI module with upto 8 data lines for XiP functionality. The controller supports singles,dual, quad or octal data lines in single (SDR) or double (DDR) data rate configurations. SDR mode supports upto 96 MHz and DDR mode supports up to 72 MHz. Several QuadSPI configurations are possible depending on QuadSPI memory selected. See QSPI Modes of Interfacing for more details.

35.1.2 QSPI Modes of Interfacing

Table 35-1. QSPI implementations supported

	2 separate QSPI	2 separate QSPI	RPC	Octal flash	Octal flash	Dual Die flash	Dual Die flash	Single Die flash	Single Die flash (A)	Single Die flash (B)
	DDR	SDR	DDR	DDR	SDR	DDR	SDR	DDR	SDR	SDR
Why use?	2 Ind. FLASH	2 Ind. FLASH	BW & Cap.	BW & Cap.	BW & Cap.	Cap.	Cap.	Cap.	min pins	min pins
Total pins	14 pins	12 pins	12 pins	11 pins	10 pins	8 pins	7 pins	7 pins	6 pins	6 pins
Flash die	2	2	1	1	1	2	2	1	1	1
Data pins per QSPI flash die	4	4	8	8	8	4	4	4	4	4
A_DQS	1	0	1	1	0	1	0	1	0	0
B_DQS	1	0	0	0	0	0	0	0	0	0
A_SS0	1	1	1	1	1	1	1	1	1	0

Table continues on the next page...

Table 35-1. QSPI implementations supported (continued)

	2 separate QSPI	2 separate QSPI	RPC	Octal flash	Octal flash	Dual Die flash	Dual Die flash	Single Die flash	Single Die flash (A)	Single Die flash (B)
	DDR	SDR	DDR	DDR	SDR	DDR	SDR	DDR	SDR	SDR
A_SS1	0	0	0	0	0	1	1	0	0	0
B_SS0	0	1	0	0	0	0	0	0	0	1
B_SS1	1	0	0	0	0	0	0	0	0	0
A_SCLK	1	1	1	1	1	1	1	1	1	0
B_SCLK	1	1	1	0	0	0	0	0	0	1
QSPI Mode Support by Package										
80 LQFP	N	N	N	N	N	Y	Y	Y	Y	N
121 MAPBGA	Y[1]	Y	Y	Y	Y	Y	Y	Y	Y	Y
100 LQFP	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
64 MAPBGA/LQFP	N	N	N	N	N	Y	Y	Y	Y	N

1. Support two separate DDR QSPI flash without DQS signal.

35.1.2.1 QSPI Pin Muxing

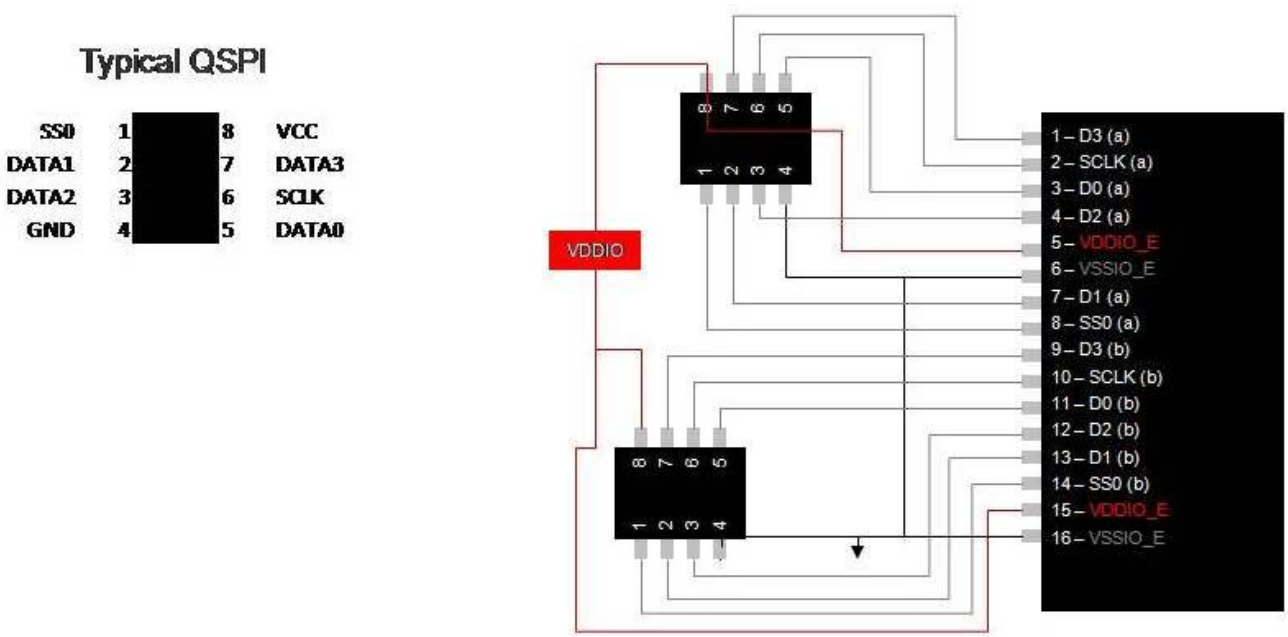


Figure 35-1. QSPI Interface to 2 Separate QSPI in SDR mode

35.1.3 QSPI clocking

Refer to clock distribution for QSPI clock source options.

35.1.4 QSPI Advanced Configuration

35.1.4.1 QSPI Clock Configuration

- Clock source selection for the clock divider is controlled by QuadSPIx_SOCCR[2:0]

Field	Description
2 – 0 QSPISRC	QSPI clock source select • 000 Core/system clock • 001 MCGFLL clock • 010 MCGPLL clock • 011 MCGPLL 2x clock (DDR mode specific) • 100 IRC48M clock. • 101 OSCERCLK clock • 110 MCGIRCLK clock. • 111 Reserved

- Clock divider is configured by QuadSPIx_MCR[27:24]

Field	Description
27 – 24 SCLKCFG	Serial clock divider • 0000 Divide by 1 • 0001 Divide by 2 • 0010 Divide by 3 • 0011 Divide by 4 • .... • 1111 Divide by 16

35.1.4.2 QSPI Octal Flash Pinmux Configuration

- Octal flash pinmux is configured by QuadSPIx_SOCCR[15:13] in addition to the Port Control module.

Field	Description
15 OCTEN	Octal data pins enable When this bit is set QSPI0B_DATAx pins are switched to QSPI Port A to composite eight bit data bus 0 QSPI0B_DATAx pins are assigned to QSPI Port B

Table continues on the next page...

Field	Description
	1 QSPI0B_DATAx pins are assigned to QSPI Port A
14 DIFFCKEN	Differential flash clock pins enable When this bit is set QSPI0B_SCLK pin serves as an inverse QSPI0A_SCLK, QSPI0B_SCLK pin serves as an inverse CK2 clock if CK2EN bit is set as well. 0 Differential flash clock is disabled. 1 Differential flash clock is enabled.
13 CK2EN	Flash CK2 clock pin enable When this bit is set QSPI0B_DQS pin serves as flash CK2 clock, QSPI0B_SCLK pin serves as an inverse CK2 clock if DIFFCKEN bit is set as well. 0 CK2 flash clock is disabled 1 CK2 flash clock is enabled

- After configure the Alt selection and QuadSPIx_SOCCR[15:13], QSPI Port A can operate in octal mode.

Pin Name	Alt Selection	Alt Function	QSPIx_SOCCR	Octal Flash Pin
PTE0	ALT5	QSPI0A_DATA3	–	DQ3
PTE1/LLWU_P0	ALT5	QSPI0A_SCLK	–	CK
PTE2/LLWU_P1	ALT5	QSPI0A_DATA0	–	DQ0
PTE3	ALT5	QSPI0A_DATA2	–	DQ2
PTE4/LLWU_P2	ALT5	QSPI0A_DATA1	–	DQ1
PTE5	ALT5	QSPI0A_SS0B	–	CS#
PTE6/LLWU_P16	ALT5	QSPI0B_DATA3	QSPIx_SOCCR[OCTEN]	DQ7
PTE7	ALT5	QSPI0B_SCLK	QSPIx_SOCCR[DIFFCKEN]	CK#
PTE8	ALT5	QSPI0B_DATA0	QSPIx_SOCCR[OCTEN]	DQ4
PTE9/LLWU_P17	ALT5	QSPI0B_DATA2	QSPIx_SOCCR[OCTEN]	DQ6
PTE10/LLWU_P18	ALT5	QSPI0B_DATA1	QSPIx_SOCCR[OCTEN]	DQ5
PTE11	ALT7	QSPI0A_DQS	–	DQS
PTE7	ALT5	QSPI0B_SCLK	QSPIx_SOCCR[CK2EN]	CK2
PTE7	ALT5	QSPI0B_SCLK	QSPIx_SOCCR[DIFFCKEN] QSPIx_SOCCR[CK2EN]	CK2#

35.1.4.3 QSPI DQS Configuration

- To support high flash clock frequency DQS function need be enabled. Besides the QuadSPIx_MCR[DQS_EN] and QuadSPIx_MCR[DQS_LAT_EN], there are other control bits in QuadSPIx_SOCCR register as below.

Field	Description
29 – 24 DLYTAPSELB	Delay chain tap number selection for QSPI Port B DQS 000000 Select 1 delay chain tap 000001 Select 2 delay chain taps 000010 Select 3 delay chain taps .... 111111 Select 64 delay chain tap
21 – 16 DLYTAPSELA	Delay chain tap number selection for QSPI Port A DQS 000000 Select 1 delay chain tap 000001 Select 2 delay chain taps 000010 Select 3 delay chain taps .... 111111 Select 64 delay chain tap
12 DQSINVSEL	Select clock source for internal DQS generation 0 Use 1x internal reference clock for the DQS generation 1 Use inverse 1x internal reference clock for the DQS generation
11 – 10 DQSPHASEL	Select phase shift for internal DQS generation. These bits are always zero in SDR mode. 00 No phase shift 01 Select 45 degree phase shift 10 Select 90 degree phase shift 11 Select 135 degree phase shift
9 DQSPADLPEN	When this bit is set the internal generated DQS will be sent to the DQS pad first and then looped back to QuadSPI. DQSLPEN should be cleared when this bit is set. 0 DQS loop back from DQS pad is disabled 1 DQS loop back from DQS pad is enabled
8 DQSLPEN	When this bit is set the internal generated DQS is selected and looped back to QuadSPI, without going to DQS pad. DQSPADLPEN should be cleared when this bit is set. 0 DQS loop back is disabled 1 DQS loop back is enabled

35.1.5 QuadSPI_MCR[SCLKCFG] implementation

The following table shows the serial clock configuration for this chip.

Table 35-2. QuadSPI_MCR[SCLKCFG] field description

Field	Description
31-24 SCLKCFG	Serial Clock Configuration. Used for dividing clocks. 0x00: Divide by 1 0x01: Divide by 2

Table 35-2. QuadSPI_MCR[SCLKCFG] field description

Field	Description
	0x02: Divide by 3 0x04: Divide by 4 ... 0x0F: Divide by 16 All others: reserved

35.1.6 QuadSPI register reset values

All reset values for the QuadSPI registers that are device specific are shown in the following tables. All other register reset values are shown in the module memory map.

Table 35-3. QuadSPI Module Configuration Register Reset Values

Register	Reset Value
Module Configuration Register (QuadSPI_MCR)	000F_400Ch

Table 35-4. QuadSPI Buffer Configuration Register Reset Values

Register	Reset Value
Buffer0 Configuration Register (QuadSPI_BUF0CR)	0000_0002h
Buffer1 Configuration Register (QuadSPI_BUF1CR)	0000_0003h
Buffer2 Configuration Register (QuadSPI_BUF2CR)	0000_0004h
Buffer3 Configuration Register (QuadSPI_BUF3CR)	8000_0000h

Table 35-5. Serial Flash A1 Top Address Reset Values

Register	Reset Value
QuadSPI0_SFA1AD	6fff_fc00h

Table 35-6. Serial Flash A2 Top Address Reset Values

Register	Reset Value
QuadSPI0_SFA2AD	6fff_fc00h

Table 35-7. Serial Flash B1 Top Address Reset Values

Register	Reset Value
QuadSPI0_SFB1AD	6fff_fc00h

Table 35-8. Serial Flash B2 Top Address Reset Values

Register	Reset Value
QuadSPI0_SFB2AD	6fff_fc00h

35.1.7 QuadSPI AHB flexible buffer size

For this chip, the size of the complete buffer mentioned in Flexible AHB buffers is 512 bytes.

35.2 Introduction

The Quad Serial Peripheral Interface (QuadSPI) block acts as an interface to one single or two external serial flash devices, each with up to eight bidirectional data lines.

35.2.1 Features

The QuadSPI supports the following features:

- Flexible sequence engine to support various flash vendor devices.
- Single, dual, quad and octal modes of operation.
- DDR/DTR mode wherein the data is generated on every edge of the serial flash clock.
- Support for flash data strobe signal for data sampling in DDR and SDR mode.
- Support for parallel writes via register mapped interface in single I/O mode.
- Two identical serial flash devices can be connected and accessed in parallel for data read operations, forming one (virtual) flash memory with doubled readout bandwidth.
- DMA support to read RX Buffer data via AMBA AHB bus (64-bit width interface) or IP registers space (32-bit access) and DMA support to fill TX Buffer via IPS register space (32-bit access).
    - Inner loop size of DMA access can be configured.
- Multimaster accesses with priority
    - Flexible and configurable buffer for each master

- Multiple interrupt conditions (see Table 35-18)
- Memory mapped read access to connected flash devices.
- Programmable sequence engine to cater to future command/protocol changes and able to support all existing vendor commands and operations.
  - Supports all types of addressing.

35.2.2 Block Diagram

The following figure is a block diagram of the Quad Serial Peripheral Interface (QuadSPI) module.

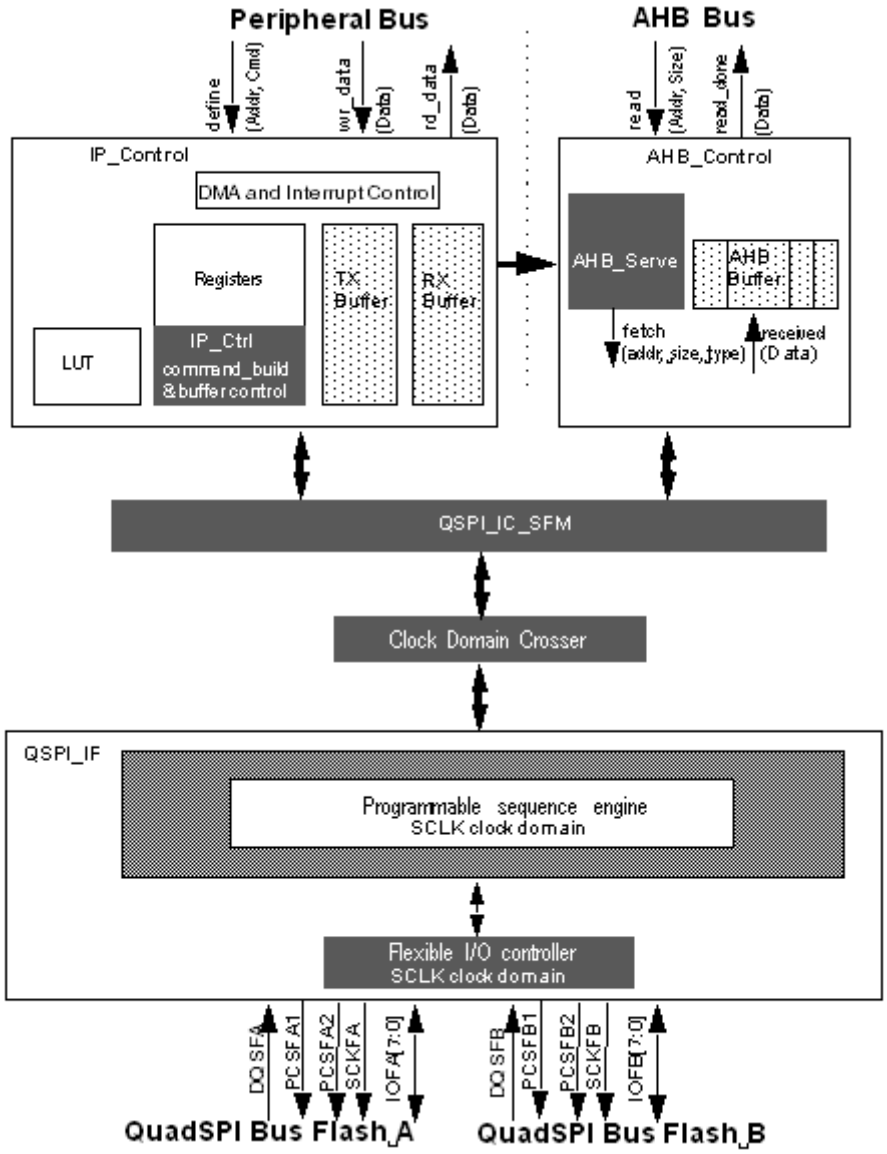


Figure 35-2. QuadSPI Block Diagram

35.2.3 QuadSPI Modes of Operation

This section provides information about the modes in which the QuadSPI module can be used.

35.2.3.1 Normal Mode

In this mode, one or two external serial flash memory devices can be accessed. Further details about this mode of operation can be found in Modes of Operation (Normal Mode).

35.2.3.2 Module Disable Mode

This mode is used for power management of the device containing the QuadSPI module. It is controlled by signals external to the QuadSPI. The clock to the non-memory mapped logic in the QuadSPI can be stopped while in Module Disable Mode.

35.2.3.3 Stop Mode

This mode is also used for power management. When a request is made to enter Stop Mode, the QuadSPI block completes the action currently being processed, before the request is acknowledged.

35.2.4 Acronyms and Abbreviations

The following table contains acronyms and abbreviations used in this document.

Table 35-9. Acronyms and Abbreviations

Terms	Description
AHB	Advanced High-performance Bus, version of AMBA
AMBA	Advanced Microcontroller Bus Architecture
BE	Big Endian Byte Ordering
CS	Chip Select
DMA	Direct Memory Access
MB	Megabyte. Each MB is 1024 * 1024 bytes
IFM	Individual Flash Mode
PFM	Parallel Flash Mode
LSB	Least Significant Bit
MSB	Most Significant Bit
PCS	Peripheral Chip Select
QSPI, QuadSPI	Quad Serial Peripheral Interface
SCK	Serial Communications Clock
w1c	Write 1 to clear, writing a 1 to this field resets the flag

35.2.5 Glossary for QuadSPI module

Table 35-10. Glossary

Term	Definition
AHB Command	An AHB Command is a SFM Command triggered by a read access to the address range belonging to the memory mapped access defined in Table 35-15. Refer to AHB Commands for details.
Asserted	A signal that is asserted is in its active state. An active low signal changes from logic level one to logic level zero when asserted, and an active high signal changes from logic level zero to logic level one.
Clear	To clear a bit or bits means to establish logic level zero on the bit or bits.
Clock Phase	Determines when the data should be sampled relative to the active edge of SCK
Clock Polarity	Determines the idle state of the SCK signal.
Drain	To remove entries from a FIFO by software or hardware.
Endianness	Byte Ordering scheme.
Field	Two or more register bits grouped together.
Fill	To add entries to a FIFO by software or hardware.
Host	Refers to another functional block in the device containing the QuadSPI module
Instruction Code	8 bits defining the type of command to be executed.
IP Command	An IP Command is a SFM Command triggered by writing into the QSPI_IPCR[SEQID] field.
Logic level one	The voltage that corresponds to Boolean true (1) state.
Logic level zero	The voltage that corresponds to Boolean false (0) state.
Negated	A signal that is negated is in its inactive state. An active low signal changes from logic level 0 to logic level 1 when negated, and an active high signal changes from logic level 1 to logic level 0.
QSPI_AMBA_BASE	First address of QuadSPI address space on system memory map.
QSPI_ARDB_BASE	First address of QuadSPI Rx Buffer on system memory map.
Set	To set a bit or bits means to establish logic level one on the bit or bits.
RX Buffer PUSH Event	Addition of valid entries into the RX Buffer. In the default case each Buffer PUSH Event adds 2 entries to the RX Buffer since the interface to the serial clock domain is 64 bits in width. Depending on the number of bytes read from the serial flash device it is possible for the very last Buffer PUSH Event that only one entry is added. The QSPI_RBSR[RDBFL] field is incremented by the number of entries added to the RX Buffer.
RX Buffer POP Event	Removal of valid entries from the RX Buffer. Each Buffer POP Event removes (QSPI_RBCT[WMRK] + 1) valid entries from the buffer. The QSPI_RBSR[RDBFL] field is decremented by the same number and the QSPI_RBSR[RDCTR] field is incremented accordingly.
Individual Flash Mode	Access to a single, individual serial flash device. Refer to Serial Flash Access Schemes for details.
Parallel Flash Mode	Read access to two serial flash devices attached to the QuadSPI module in parallel. Refer to Serial Flash Access Schemes for details.
SFM Command	Serial Flash Memory Command. A SFM command consists of an instruction code and all other parameters (for example, size or mode bytes) needed for that specific instruction code. Triggering a command either initiates a transaction on the external serial flash or results in an error. Refer to Table 35-29 for details on errors.
Single/Dual/Quad/Octal Instructions	Depending on the serial flash device connected to the QuadSPI module there will be instructions using a different number of data lines. • Single: Single line I/O with one data out and one data in line to/from the serial flash device. • Dual: Dual line I/O with two bidirectional I/O lines, driven alternatively by the serial flash device or the QuadSPI module

Table continues on the next page...

Table 35-10. Glossary (continued)

Term	Definition
	• Quad: Quad line I/O with 4 bidirectional I/O lines, driven alternatively by the serial flash device or the QuadSPI • Octal: Octal line I/O with 8 bidirectional I/O lines, driven alternatively by the serial flash device or the QuadSPI
Transaction	A transaction consists of all flags, data and signals in either direction to execute a command for an attached serial flash device. It is a combination of chip select, sclk, instruction code, address, mode- and/or dummy bytes, transmit and/or receive data.
LUT	Look-up table.

35.3 External Signal Description

This section provides the external signal information of the QuadSPI module.

The following table lists the external signals belonging to the module in conjunction with the different modes of operation.

Table 35-11. Signal Properties

Signal Name	Function	Direction	Description
PCSFA1	Peripheral Chip Select Flash A1	O	This signal is the chip select for the serial flash device A1. A1 represents the first device in a dual-die package flash A or the first of the two flash devices that share IOFA. Refer to Dual Die Flashes for more details.
PCSFA2	Peripheral Chip Select Flash A2	O	This signal is the chip select for the serial flash device A2. A2 represents the second device in a dual-die package flash A or the second of the two flash devices that share IOFA. Refer to Dual Die Flashes for more details.
PCSFB1	Peripheral Chip Select Flash B1	O	This signal is the chip select for the serial flash device B1. B1 represents the first device in a dual-die package flash B or the first of the two flash devices that share IOFB. Refer to Dual Die Flashes for more details.
PCSFB2	Peripheral Chip Select Flash B2	O	This signal is the chip select for the serial flash device B2. B2 represents the second device in a dual-die package flash B or the second of the two flash devices that share IOFB. Refer to Dual Die Flashes for more details.
SCKFA	Serial Clock Flash A	O	This signal is the serial clock output to the serial flash device A.
SCKFB	Serial Clock Flash B	O	This signal is the serial clock output to the serial flash device B.
IOFA[7:0]	Serial I/O Flash A	I/O	These signals are the data I/O lines to/from the serial flash device A. Refer to Driving External Signals for details about the signal drive and timing behavior. Note that the signal pins of the serial flash device may change their function according to the SFM Command executed, leaving them as control inputs when Single and Dual Instructions are executed. The module supports driving these inputs to dedicated values. In single I/O mode, QuadSPI drives data on IOFA[0] and expects data on IOFA[1].
IOFB[7:0]	Serial I/O Flash B	I/O	These signals are the data I/O lines to/from the serial flash device B. Refer to Driving External Signals for details about the signal drive and timing behavior. Note that the signal pins of the serial flash device may change their function according to the SFM Command executed, leaving them as

Table continues on the next page...

Table 35-11. Signal Properties (continued)

Signal Name	Function	Direction	Description
			control inputs when Single and Dual Instructions are executed. The module supports driving these inputs to dedicated values. In single I/O mode, QuadSPI drives data on IOFB[0] and expects data on IOFB[1].
DQSFA	Data Strobe signal Flash A	I	Data strobe signal for port A. Some flash vendors provide the DQS signal to which the read data is aligned in DDR mode.
DQSFB	Data Strobe signal Flash B	I	Data strobe signal for port B. Some flash vendors provide the DQS signal to which the read data is aligned in DDR mode.

35.3.1 Driving External Signals

The different phases of the serial flash access scheme are shown in the following figure.

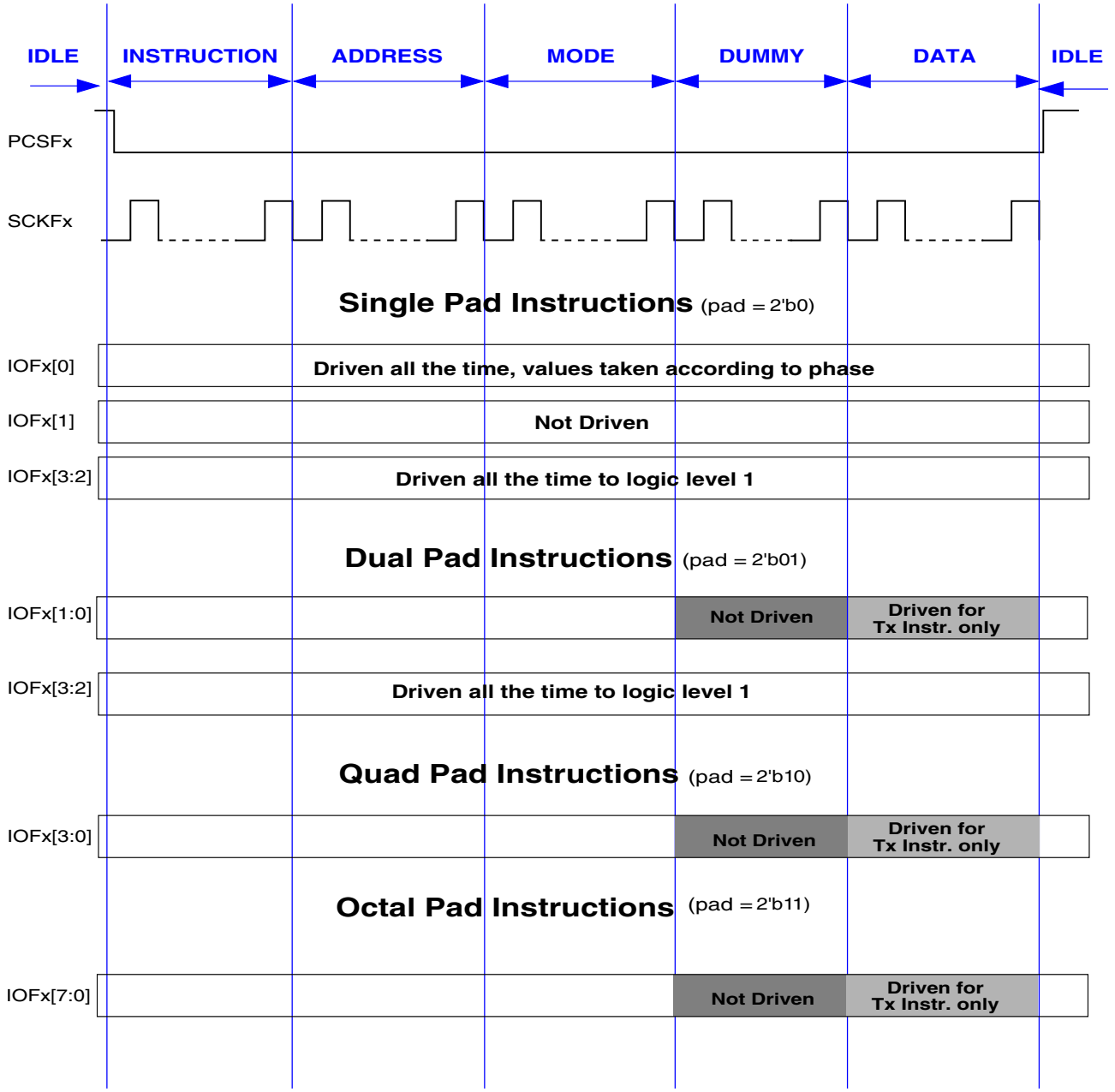


Figure 35-3. Serial Flash Access Scheme

The different phases and the I/O driving characteristics of the QuadSPI module are characterized in the following way:

- IDLE: Serial flash device not selected. No interaction with the serial flash device. All IOFx signals driven.
- INSTRUCTION: Serial flash device selected. The instruction is sent to the serial flash device. All IOFx signals are driven.

- ADDRESS: Serial Flash Address is sent to the device. All IOFx signals are driven. Note that this phase is not applicable for all SFM Commands.
- MODE: Mode bytes are sent to the serial flash device. All IOFx signals are driven. Note that this phase is not applicable for all SFM Commands.
- DUMMY: Dummy clocks are provided to the serial flash device. Refer to the Figure 35-3 for the IOFx signals driven. The actual data lines required for the SFM Command executed are not driven for data read commands. Note that this phase is not applicable for all SFM Commands.
- DATA: Serial flash data are sent to or received from the serial flash device. Refer to the preceding figure for the IOFx signals driven. The actual data lines required for the SFM Command executed are not driven for data read commands. Note that this phase is not applicable for all SFM Commands.

The PCSFx and SCKFx signals are driven permanently throughout all the phases.

In Individual Flash Mode this applies to the selected flash device. In Parallel Flash Mode this applies to both serial flash devices simultaneously.

35.4 Memory Map and Register Definition

This section provides the memory map and register definitions of the QuadSPI module.

35.4.1 Register Write Access

This section describes the write access restriction terms that apply to all registers, which can be one of the following:

- **Register Write Access Restriction**

  For each register bit and register field, the write access conditions are specified in the detailed register description. A description of the write access conditions is given in the following table. If, for a specific register bit or field, none of the given write access conditions is fulfilled, any write attempt to this register bit or field is ignored without any notification. The values of the bits or fields are not changed.

  The condition term [A or B] indicates that the register or field can be written to if at least one of the conditions is fulfilled.

Table 35-12. Register Write Access Restrictions

Condition	Description
Anytime	No write access restriction.
Disabled Mode	Write access only if *QSPI_MCR[MDIS] = 1*.
Normal Mode	Write access only if the module is in *Normal Mode*.

- **Register Write Access Requirements**

  All registers can be accessed with 8-bit, 16-bit, and 32-bit wide operations. For some of the registers, at least a 16/32-bit wide write access is required to ensure correct operation. This write access requirement is stated in the detailed register description for each register affected.

35.4.2 Peripheral Bus Register Descriptions

This section provides the memory map and register definitions of the QuadSPI module.

NOTE

Accesses to location 0x4 do not give transfer errors.

QuadSPI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4005_A000	Module Configuration Register (QuadSPI0_MCR)	32	R/W	See section	35.4.2.1/ 781
4005_A008	IP Configuration Register (QuadSPI0_IPCR)	32	R/W	0000_0000h	35.4.2.2/ 784
4005_A00C	Flash Configuration Register (QuadSPI0_FLSHCR)	32	R/W	0000_0303h	35.4.2.3/ 785
4005_A010	Buffer0 Configuration Register (QuadSPI0_BUF0CR)	32	R/W	See section	35.4.2.4/ 786
4005_A014	Buffer1 Configuration Register (QuadSPI0_BUF1CR)	32	R/W	See section	35.4.2.5/ 787
4005_A018	Buffer2 Configuration Register (QuadSPI0_BUF2CR)	32	R/W	See section	35.4.2.6/ 789
4005_A01C	Buffer3 Configuration Register (QuadSPI0_BUF3CR)	32	R/W	See section	35.4.2.7/ 790

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4005_A020	Buffer Generic Configuration Register (QuadSPI0_BFGENCR)	32	R/W	0000_0000h	35.4.2.8/ 791
4005_A024	SOC Configuration Register (QuadSPI0_SOCCR)	32	R/W	0000_0000h	35.4.2.9/ 792
4005_A030	Buffer0 Top Index Register (QuadSPI0_BUF0IND)	32	R/W	0000_0000h	35.4.2.10/ 792
4005_A034	Buffer1 Top Index Register (QuadSPI0_BUF1IND)	32	R/W	0000_0000h	35.4.2.11/ 793
4005_A038	Buffer2 Top Index Register (QuadSPI0_BUF2IND)	32	R/W	0000_0000h	35.4.2.12/ 794
4005_A100	Serial Flash Address Register (QuadSPI0_SFAR)	32	R/W	0000_0000h	35.4.2.13/ 795
4005_A104	Serial Flash Address Configuration Register (QuadSPI0_SFACR)	32	R/W	0000_0000h	35.4.2.14/ 795
4005_A108	Sampling Register (QuadSPI0_SMPR)	32	R/W	0000_0000h	35.4.2.15/ 796
4005_A10C	RX Buffer Status Register (QuadSPI0_RBSR)	32	R	0000_0000h	35.4.2.16/ 798
4005_A110	RX Buffer Control Register (QuadSPI0_RBCT)	32	R/W	0000_0000h	35.4.2.17/ 799
4005_A150	TX Buffer Status Register (QuadSPI0_TBSR)	32	R	0000_0000h	35.4.2.18/ 800
4005_A154	TX Buffer Data Register (QuadSPI0_TBDR)	32	R/W	0000_0000h	35.4.2.19/ 801
4005_A158	Tx Buffer Control Register (QuadSPI0_TBCT)	32	R/W	0000_0000h	35.4.2.20/ 802
4005_A15C	Status Register (QuadSPI0_SR)	32	R	0200_3800h	35.4.2.21/ 803
4005_A160	Flag Register (QuadSPI0_FR)	32	w1c	0800_0000h	35.4.2.22/ 806
4005_A164	Interrupt and DMA Request Select and Enable Register (QuadSPI0_RSER)	32	R/W	0000_0000h	35.4.2.23/ 809
4005_A168	Sequence Suspend Status Register (QuadSPI0_SPNDST)	32	R	0000_0000h	35.4.2.24/ 813
4005_A16C	Sequence Pointer Clear Register (QuadSPI0_SPTRCLR)	32	R/W	0000_0000h	35.4.2.25/ 815
4005_A180	Serial Flash A1 Top Address (QuadSPI0_SFA1AD)	32	R/W	See section	35.4.2.26/ 816
4005_A184	Serial Flash A2 Top Address (QuadSPI0_SFA2AD)	32	R/W	See section	35.4.2.27/ 816
4005_A188	Serial Flash B1Top Address (QuadSPI0_SFB1AD)	32	R/W	See section	35.4.2.28/ 817
4005_A18C	Serial Flash B2Top Address (QuadSPI0_SFB2AD)	32	R/W	See section	35.4.2.29/ 817

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4005_A190	Data Learn Pattern Register (QuadSPI0_DLPR)	32	R/W	AA55_3443h	35.4.2.30/ 818
4005_A200	RX Buffer Data Register (QuadSPI0_RBDR0)	32	R	0000_0000h	35.4.2.31/ 818
4005_A204	RX Buffer Data Register (QuadSPI0_RBDR1)	32	R	0000_0000h	35.4.2.31/ 818
4005_A208	RX Buffer Data Register (QuadSPI0_RBDR2)	32	R	0000_0000h	35.4.2.31/ 818
4005_A20C	RX Buffer Data Register (QuadSPI0_RBDR3)	32	R	0000_0000h	35.4.2.31/ 818
4005_A210	RX Buffer Data Register (QuadSPI0_RBDR4)	32	R	0000_0000h	35.4.2.31/ 818
4005_A214	RX Buffer Data Register (QuadSPI0_RBDR5)	32	R	0000_0000h	35.4.2.31/ 818
4005_A218	RX Buffer Data Register (QuadSPI0_RBDR6)	32	R	0000_0000h	35.4.2.31/ 818
4005_A21C	RX Buffer Data Register (QuadSPI0_RBDR7)	32	R	0000_0000h	35.4.2.31/ 818
4005_A220	RX Buffer Data Register (QuadSPI0_RBDR8)	32	R	0000_0000h	35.4.2.31/ 818
4005_A224	RX Buffer Data Register (QuadSPI0_RBDR9)	32	R	0000_0000h	35.4.2.31/ 818
4005_A228	RX Buffer Data Register (QuadSPI0_RBDR10)	32	R	0000_0000h	35.4.2.31/ 818
4005_A22C	RX Buffer Data Register (QuadSPI0_RBDR11)	32	R	0000_0000h	35.4.2.31/ 818
4005_A230	RX Buffer Data Register (QuadSPI0_RBDR12)	32	R	0000_0000h	35.4.2.31/ 818
4005_A234	RX Buffer Data Register (QuadSPI0_RBDR13)	32	R	0000_0000h	35.4.2.31/ 818
4005_A238	RX Buffer Data Register (QuadSPI0_RBDR14)	32	R	0000_0000h	35.4.2.31/ 818
4005_A23C	RX Buffer Data Register (QuadSPI0_RBDR15)	32	R	0000_0000h	35.4.2.31/ 818
4005_A300	LUT Key Register (QuadSPI0_LUTKEY)	32	R/W	5AF0_5AF0h	35.4.2.32/ 819
4005_A304	LUT Lock Configuration Register (QuadSPI0_LCKCR)	32	R/W	0000_0002h	35.4.2.33/ 820
4005_A310	Look-up Table register (QuadSPI0_LUT0)	32	R/W	See section	35.4.2.34/ 821
4005_A314	Look-up Table register (QuadSPI0_LUT1)	32	R/W	See section	35.4.2.34/ 821
4005_A318	Look-up Table register (QuadSPI0_LUT2)	32	R/W	See section	35.4.2.34/ 821

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4005_A31C	Look-up Table register (QuadSPI0_LUT3)	32	R/W	See section	35.4.2.34/ 821
4005_A320	Look-up Table register (QuadSPI0_LUT4)	32	R/W	See section	35.4.2.34/ 821
4005_A324	Look-up Table register (QuadSPI0_LUT5)	32	R/W	See section	35.4.2.34/ 821
4005_A328	Look-up Table register (QuadSPI0_LUT6)	32	R/W	See section	35.4.2.34/ 821
4005_A32C	Look-up Table register (QuadSPI0_LUT7)	32	R/W	See section	35.4.2.34/ 821
4005_A330	Look-up Table register (QuadSPI0_LUT8)	32	R/W	See section	35.4.2.34/ 821
4005_A334	Look-up Table register (QuadSPI0_LUT9)	32	R/W	See section	35.4.2.34/ 821
4005_A338	Look-up Table register (QuadSPI0_LUT10)	32	R/W	See section	35.4.2.34/ 821
4005_A33C	Look-up Table register (QuadSPI0_LUT11)	32	R/W	See section	35.4.2.34/ 821
4005_A340	Look-up Table register (QuadSPI0_LUT12)	32	R/W	See section	35.4.2.34/ 821
4005_A344	Look-up Table register (QuadSPI0_LUT13)	32	R/W	See section	35.4.2.34/ 821
4005_A348	Look-up Table register (QuadSPI0_LUT14)	32	R/W	See section	35.4.2.34/ 821
4005_A34C	Look-up Table register (QuadSPI0_LUT15)	32	R/W	See section	35.4.2.34/ 821
4005_A350	Look-up Table register (QuadSPI0_LUT16)	32	R/W	See section	35.4.2.34/ 821
4005_A354	Look-up Table register (QuadSPI0_LUT17)	32	R/W	See section	35.4.2.34/ 821
4005_A358	Look-up Table register (QuadSPI0_LUT18)	32	R/W	See section	35.4.2.34/ 821
4005_A35C	Look-up Table register (QuadSPI0_LUT19)	32	R/W	See section	35.4.2.34/ 821
4005_A360	Look-up Table register (QuadSPI0_LUT20)	32	R/W	See section	35.4.2.34/ 821
4005_A364	Look-up Table register (QuadSPI0_LUT21)	32	R/W	See section	35.4.2.34/ 821
4005_A368	Look-up Table register (QuadSPI0_LUT22)	32	R/W	See section	35.4.2.34/ 821
4005_A36C	Look-up Table register (QuadSPI0_LUT23)	32	R/W	See section	35.4.2.34/ 821
4005_A370	Look-up Table register (QuadSPI0_LUT24)	32	R/W	See section	35.4.2.34/ 821

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4005_A374	Look-up Table register (QuadSPI0_LUT25)	32	R/W	See section	35.4.2.34/ 821
4005_A378	Look-up Table register (QuadSPI0_LUT26)	32	R/W	See section	35.4.2.34/ 821
4005_A37C	Look-up Table register (QuadSPI0_LUT27)	32	R/W	See section	35.4.2.34/ 821
4005_A380	Look-up Table register (QuadSPI0_LUT28)	32	R/W	See section	35.4.2.34/ 821
4005_A384	Look-up Table register (QuadSPI0_LUT29)	32	R/W	See section	35.4.2.34/ 821
4005_A388	Look-up Table register (QuadSPI0_LUT30)	32	R/W	See section	35.4.2.34/ 821
4005_A38C	Look-up Table register (QuadSPI0_LUT31)	32	R/W	See section	35.4.2.34/ 821
4005_A390	Look-up Table register (QuadSPI0_LUT32)	32	R/W	See section	35.4.2.34/ 821
4005_A394	Look-up Table register (QuadSPI0_LUT33)	32	R/W	See section	35.4.2.34/ 821
4005_A398	Look-up Table register (QuadSPI0_LUT34)	32	R/W	See section	35.4.2.34/ 821
4005_A39C	Look-up Table register (QuadSPI0_LUT35)	32	R/W	See section	35.4.2.34/ 821
4005_A3A0	Look-up Table register (QuadSPI0_LUT36)	32	R/W	See section	35.4.2.34/ 821
4005_A3A4	Look-up Table register (QuadSPI0_LUT37)	32	R/W	See section	35.4.2.34/ 821
4005_A3A8	Look-up Table register (QuadSPI0_LUT38)	32	R/W	See section	35.4.2.34/ 821
4005_A3AC	Look-up Table register (QuadSPI0_LUT39)	32	R/W	See section	35.4.2.34/ 821
4005_A3B0	Look-up Table register (QuadSPI0_LUT40)	32	R/W	See section	35.4.2.34/ 821
4005_A3B4	Look-up Table register (QuadSPI0_LUT41)	32	R/W	See section	35.4.2.34/ 821
4005_A3B8	Look-up Table register (QuadSPI0_LUT42)	32	R/W	See section	35.4.2.34/ 821
4005_A3BC	Look-up Table register (QuadSPI0_LUT43)	32	R/W	See section	35.4.2.34/ 821
4005_A3C0	Look-up Table register (QuadSPI0_LUT44)	32	R/W	See section	35.4.2.34/ 821
4005_A3C4	Look-up Table register (QuadSPI0_LUT45)	32	R/W	See section	35.4.2.34/ 821
4005_A3C8	Look-up Table register (QuadSPI0_LUT46)	32	R/W	See section	35.4.2.34/ 821

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4005_A3CC	Look-up Table register (QuadSPI0_LUT47)	32	R/W	See section	35.4.2.34/ 821
4005_A3D0	Look-up Table register (QuadSPI0_LUT48)	32	R/W	See section	35.4.2.34/ 821
4005_A3D4	Look-up Table register (QuadSPI0_LUT49)	32	R/W	See section	35.4.2.34/ 821
4005_A3D8	Look-up Table register (QuadSPI0_LUT50)	32	R/W	See section	35.4.2.34/ 821
4005_A3DC	Look-up Table register (QuadSPI0_LUT51)	32	R/W	See section	35.4.2.34/ 821
4005_A3E0	Look-up Table register (QuadSPI0_LUT52)	32	R/W	See section	35.4.2.34/ 821
4005_A3E4	Look-up Table register (QuadSPI0_LUT53)	32	R/W	See section	35.4.2.34/ 821
4005_A3E8	Look-up Table register (QuadSPI0_LUT54)	32	R/W	See section	35.4.2.34/ 821
4005_A3EC	Look-up Table register (QuadSPI0_LUT55)	32	R/W	See section	35.4.2.34/ 821
4005_A3F0	Look-up Table register (QuadSPI0_LUT56)	32	R/W	See section	35.4.2.34/ 821
4005_A3F4	Look-up Table register (QuadSPI0_LUT57)	32	R/W	See section	35.4.2.34/ 821
4005_A3F8	Look-up Table register (QuadSPI0_LUT58)	32	R/W	See section	35.4.2.34/ 821
4005_A3FC	Look-up Table register (QuadSPI0_LUT59)	32	R/W	See section	35.4.2.34/ 821
4005_A400	Look-up Table register (QuadSPI0_LUT60)	32	R/W	See section	35.4.2.34/ 821
4005_A404	Look-up Table register (QuadSPI0_LUT61)	32	R/W	See section	35.4.2.34/ 821
4005_A408	Look-up Table register (QuadSPI0_LUT62)	32	R/W	See section	35.4.2.34/ 821
4005_A40C	Look-up Table register (QuadSPI0_LUT63)	32	R/W	See section	35.4.2.34/ 821

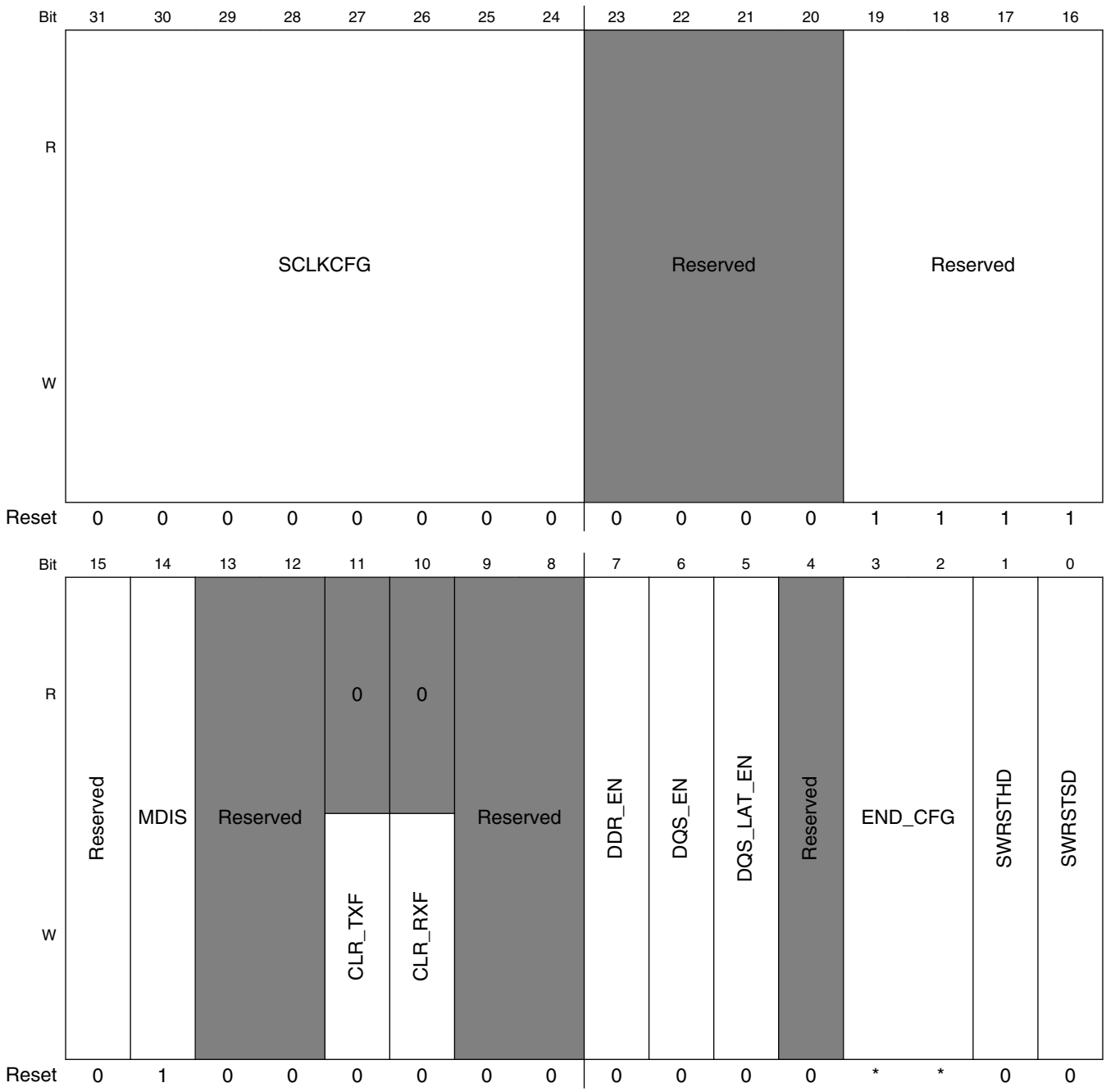
35.4.2.1 Module Configuration Register (QuadSPIx_MCR)

The QuadSPI_MCR holds configuration data associated with QuadSPI operation.

Write:

- *All other fields: Anytime*

Address: 4005_A000h base + 0h offset = 4005_A000h



* Notes:

- END_CFG field: See the module configuration for the device specific reset value.

QuadSPIx_MCR field descriptions

Field	Description
31–24 SCLKCFG	Serial Clock Configuration. This field configuration is chip specific. For details, refer to chip-specific QuadSPI information. It may be used for dividing clocks.

Table continues on the next page...

QuadSPIx_MCR field descriptions (continued)

Field	Description
23–20 Reserved	This field is reserved.
19–16 Reserved	This field is reserved. This field is reserved and should always be set to 0xF.
15 Reserved	This field is reserved. This field is reserved.
14 MDIS	Module Disable. The MDIS bit allows the clock to the non-memory mapped logic in the QuadSPI to be stopped, putting the QuadSPI in a software controlled power-saving state. 0 Enable QuadSPI clocks. 1 Allow external logic to disable QuadSPI clocks.
13–12 Reserved	This field is reserved.
11 CLR_TXF	Clear TX FIFO/Buffer. Invalidates the TX Buffer content. This is a self-clearing field. 0 No action. 1 Read and write pointers of the TX Buffer are reset to 0. QSPI_TBSR[TRCTR] is reset to 0.
10 CLR_RXF	Clear RX FIFO. Invalidates the RX Buffer. This is a self-clearing field. 0 No action. 1 Read and write pointers of the RX Buffer are reset to 0. QSPI_RBSR[RDBFL] is reset to 0.
9–8 Reserved	This field is reserved.
7 DDR_EN	DDR mode enable 0 2x and 4x clocks are disabled for SDR instructions only 1 2x and 4x clocks are enabled supports both SDR and DDR instruction.
6 DQS_EN	DQS enable. This field is valid for both SDR and DDR mode. For more details, refer to Data Strobe (DQS) sampling method. 0 DQS disabled. 1 DQS enabled. When enabled, the incoming data is sampled on both the edges of DQS input when QSPI_MCR[DDR_EN] is set, else, on only one edge when QSPI_MCR[DDR_EN] is 0. The QSPI_SMPR[DDR_SMP] values are ignored.
5 DQS_LAT_EN	DQS Latency Enable. This field is valid when latency is included in between read access from flash memory in cases when QSPI_MCR[DQS_EN] is 1. For more details, refer to Data Strobe (DQS) sampling method. 0 DQS Latency disabled 1 DQS feature with latency included enabled
4 Reserved	This field is reserved.
3–2 END_CFG	Defines the endianness of the QuadSPI module. For more details refer to Byte Ordering Endianess
1 SWRSTHD	Software reset for AHB domain

Table continues on the next page...

QuadSPIx_MCR field descriptions (continued)

Field	Description
	0 No action 1 AHB domain flops are reset. Does not reset configuration registers. It is advisable to reset both the serial flash domain and AHB domain at the same time. Resetting only one domain might lead to side effects. **NOTE:** The software resets need the clock to be running to propagate to the design. The MCR[MDIS] should therefore be set to 0 when the software reset bits are asserted. Also, before they can be deasserted again (by setting MCR[SWRSTHD] to 0), it is recommended to set the MCR[MDIS] bit to 1. Once the software resets have been deasserted, the normal operation can be started by setting the MCR[MDIS] bit to 0.
0 SWRSTSD	Software reset for serial flash domain 0 No action 1 Serial Flash domain flops are reset. Does not reset configuration registers. It is advisable to reset both the serial flash domain and AHB domain at the same time. Resetting only one domain might lead to side effects. **NOTE:** The software resets need the clock to be running to propagate to the design. The MCR[MDIS] should therefore be set to 0 when the software reset bits are asserted. Also, before they can be deasserted again (by setting MCR[SWRSTSD] to 0), it is recommended to set the MCR[MDIS] bit to 1. Once the software resets have been deasserted, the normal operation can be started by setting the MCR[MDIS] bit to 0.

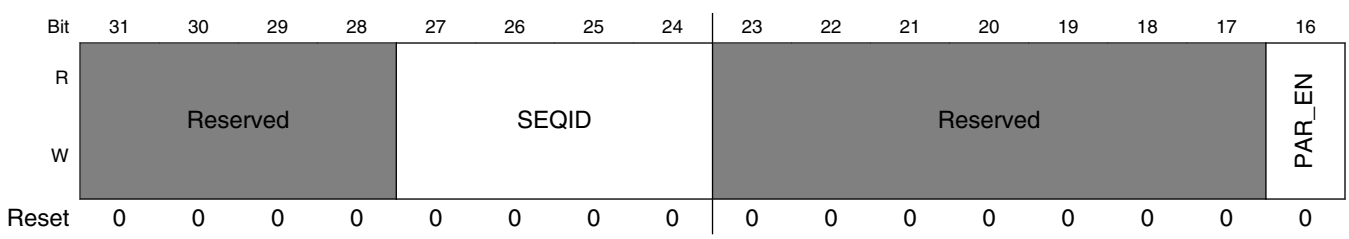
35.4.2.2 IP Configuration Register (QuadSPIx_IPCR)

The IP configuration register provides all the configuration required for an IP initiated command. An IP command can be triggered by writing in the SEQID field of this register. If the SEQID field is written successfully, a new command to the external serial flash is started as per the sequence pointed to by the SEQID field. Refer to Normal Mode , for details about the command triggering and command execution.

Write:

- *QSPI_SR[IP_ACC]=0*

Address: 4005_A000h base + 8h offset = 4005_A008h



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

QuadSPIx_IPCR field descriptions

Field	Description
31–28 Reserved	This field is reserved.
27–24 SEQID	Points to a sequence in the Look-up table. This field defines bits [6:2] of the LUT index. The bits [1:0] are always assumed to be 0. Refer to Look-up Table for more details. A write to this field triggers a transaction on the serial flash interface.
23–17 Reserved	This field is reserved.
16 PAR_EN	When set, a transaction to two serial flash devices is triggered in parallel mode. Refer to Parallel Flash Mode for more details.
IDATSZ	IP data transfer size. Defines the data transfer size in bytes of the IP command.

35.4.2.3 Flash Configuration Register (QuadSPIx_FLSHCR)

The Flash configuration register contains the flash device specific timings that must be met by the QuadSPI controller for the device to function correctly.

Write:

- *QSPI_SR[AHB_ACC] = 0*
- *QSPI_SR[IP_ACC] = 0*

Address: 4005_A000h base + Ch offset = 4005_A00Ch

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 W	Reserved														TDH		Reserved				TCSH				Reserved				TCSS			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1

QuadSPIx_FLSHCR field descriptions

Field	Description
31–18 Reserved	This field is reserved.
17–16 TDH	Serial flash data in hold time. This helps in meeting the Data In Hold time requirement of a flash. This is valid only in DDR mode. **NOTE:** This field should be set to 0x00 in SDR mode (QuadSPI_MCR[DDR_EN]=0). Refer to Data input hold requirement of Flash for details.

Table continues on the next page...

QuadSPIx_FLSHCR field descriptions (continued)

Field	Description
	00 Data aligned with the posedge of Internal reference clock of QuadSPI 01 Data aligned with 2x serial flash half clock 10 Data aligned with 4x serial flash half clock
15–12 Reserved	This field is reserved.
11–8 TCSH	Serial flash CS hold time in terms of serial flash clock cycles. **NOTE:** The actual delay between chip select and clock is defined as: • TCSH = 1 SCK clk if N= 0/1 else, N SCK clk if N>1, where N is the setting of TCSH
7–4 Reserved	This field is reserved. Reserved.
TCSS	Serial flash CS setup time in terms of serial flash clock cycles. **NOTE:** 1. The actual delay between chip select and clock is defined as: • TCSS = 0.5 SCK clk if N= 0/1 else, N+0.5 SCK clk if N>1, where N is the setting of TCSS. 2. Any update to the TCSS register bits is visible on the flash interface only from the second transaction following the update.

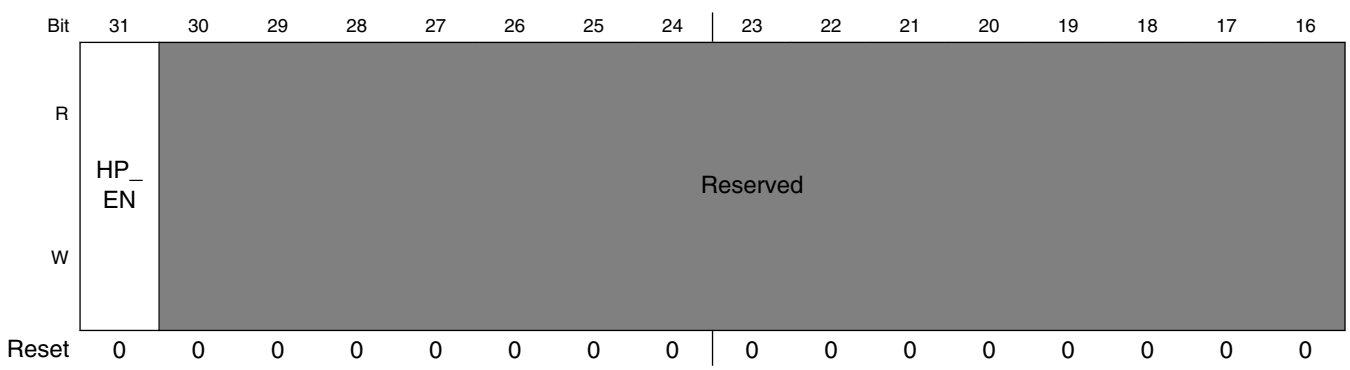
35.4.2.4 Buffer0 Configuration Register (QuadSPIx_BUF0CR)

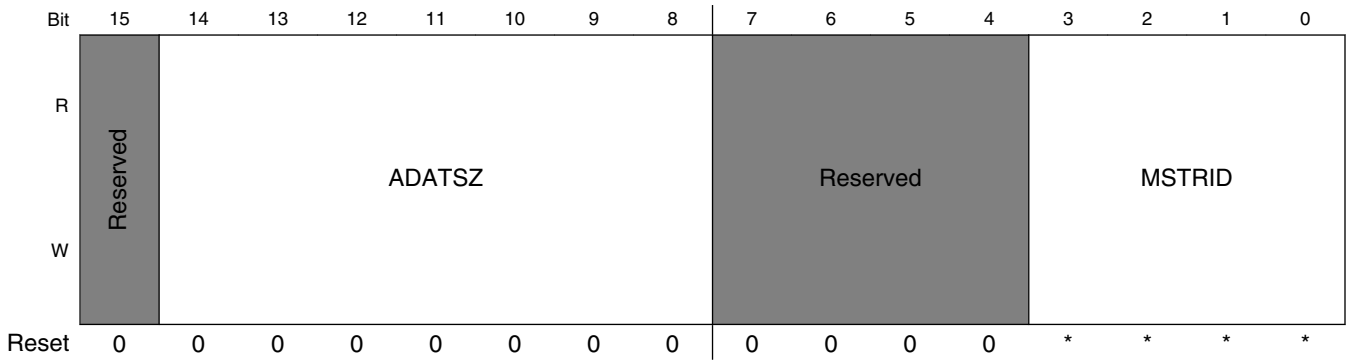
This register provides the configuration for any access to buffer0. An access is routed to buffer0 when the master port number of the incoming AHB request matches the MSTRID field of BUF0CR. Any buffer "miss" leads to a serial flash transaction being triggered as per the sequence pointed to the SEQID field. Buffer0 may also be configured as a high priority buffer by setting the HP_EN field of this register.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 4005_A000h base + 10h offset = 4005_A010h





* Notes:
- MSTRID field: See the module configuration for reset value.

QuadSPIx_BUF0CR field descriptions

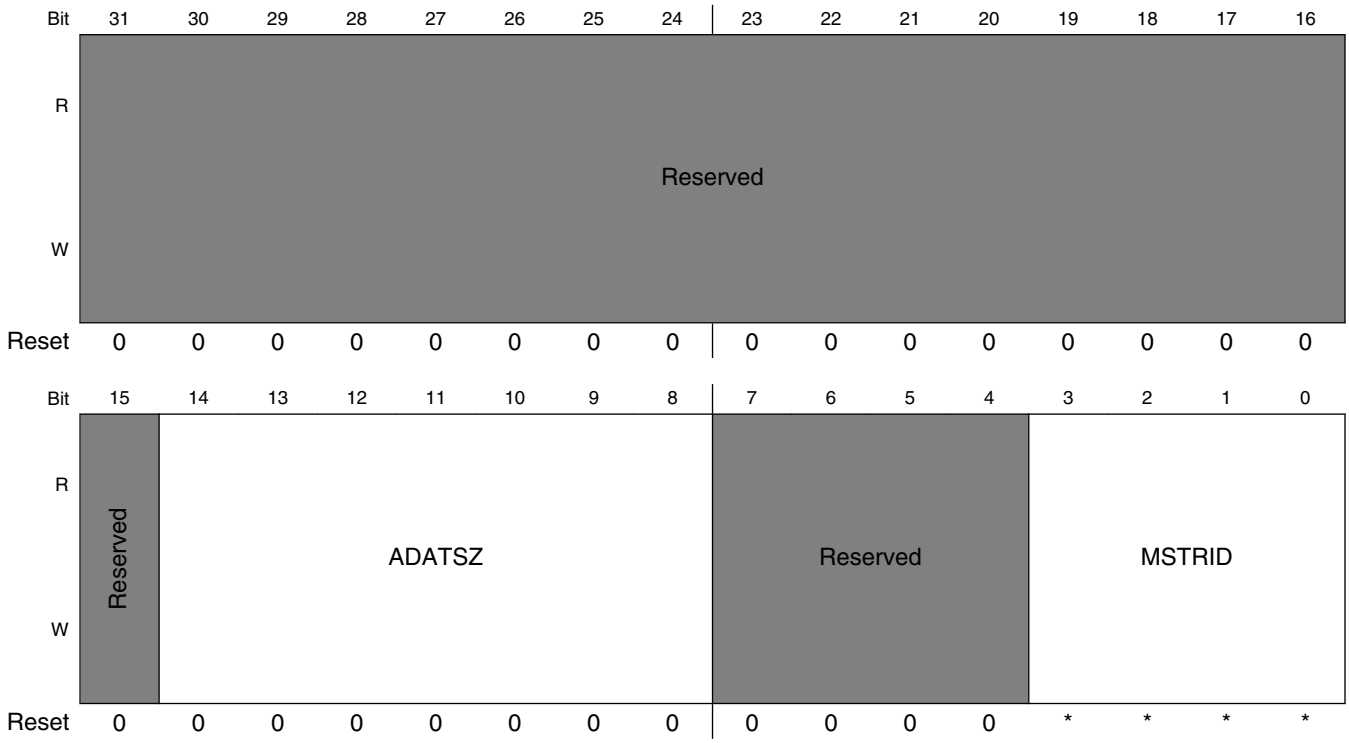
Field	Description
31 HP_EN	High Priority Enable. When set, the master associated with this buffer is assigned a priority higher than the rest of the masters. An access by a high priority master will suspend any ongoing prefetch by another AHB master and will be serviced on high priority. Refer to Flexible AHB Buffers for details.
30–16 Reserved	This field is reserved.
15 Reserved	This field is reserved.
14–8 ADATSZ	AHB data transfer size Defines the data transfer size in 8 bytes of an AHB triggered access to serial flash. For example, a value of 0x2 will set transfer size to 16 bytes. When ADATSZ = 0, the data size mentioned the sequence pointed to by the SEQID field overrides this value. Software should ensure that this transfer size is not greater than the size of this buffer.
7–4 Reserved	This field is reserved.
MSTRID	Master ID. The ID of the AHB master associated with BUFFER0. Any AHB access with this master port number is routed to this buffer. It must be ensured that the master IDs associated with all buffers must be different.

35.4.2.5 Buffer1 Configuration Register (QuadSPIx_BUF1CR)

This register provides the configuration for any access to buffer1. An access is routed to buffer1 when the master port number of the incoming AHB request matches the MSTRID field of BUF1CR. Any buffer "miss" leads to the buffer being flushed and a serial flash transaction being triggered as per the sequence pointed to by the SEQID field.

Write:
- *QSPI_SR[AHB_ACC] = 0*

Address: 4005_A000h base + 14h offset = 4005_A014h



* Notes:

- MSTRID field: See the module configuration for reset value.

QuadSPIx_BUF1CR field descriptions

Field	Description
31–16 Reserved	This field is reserved.
15 Reserved	This field is reserved.
14–8 ADATSZ	AHB data transfer size Defines the data transfer size in 8 bytes of an AHB triggered access to serial flash.For example, a value of 0x2 will set transfer size to 16bytes. When ADATSZ = 0, the data size mentioned the sequence pointed to by the SEQID field overrides this value. Software should ensure that this transfer size is not greater than the size of this buffer.
7–4 Reserved	This field is reserved.
MSTRID	Master ID. The ID of the AHB master associated with BUFFER1. Any AHB access with this master port number is routed to this buffer. It must be ensured that the master IDs associated with all buffers must be different.

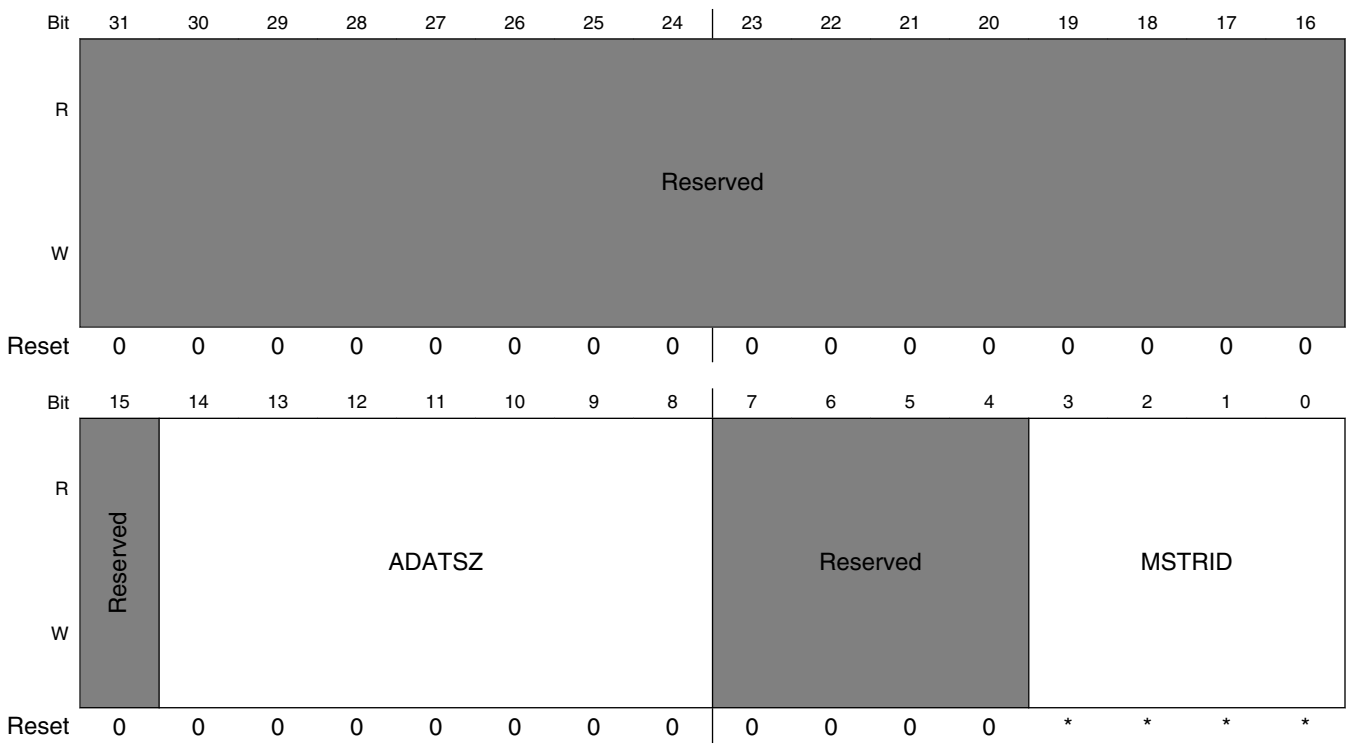
35.4.2.6 Buffer2 Configuration Register (QuadSPIx_BUF2CR)

This register provides the configuration for any access to buffer2. An access is routed to buffer2 when the master port number of the incoming AHB request matches the MSTRID field of BUF2CR. Any buffer "miss" leads to the buffer being flushed and a serial flash transaction being triggered as per the sequence pointed to by the SEQID field.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 4005_A000h base + 18h offset = 4005_A018h



* Notes:

- MSTRID field: See the module configuration for the device specific reset value.

QuadSPIx_BUF2CR field descriptions

Field	Description
31–16 Reserved	This field is reserved. Reserved.
15 Reserved	This field is reserved.
14–8 ADATSZ	AHB data transfer size Defines the data transfer size in 8 bytes of an AHB triggered access to serial flash. For example, a value of 0x2 will set transfer size to 16 bytes. When ADATSZ = 0, the data size mentioned the sequence pointed

Table continues on the next page...

QuadSPIx_BUF2CR field descriptions (continued)

Field	Description
	to by the SEQID field overrides this value. Software should ensure that this transfer size is not greater than the size of this buffer.
7–4 Reserved	This field is reserved. Reserved.
MSTRID	Master ID. The ID of the AHB master associated with BUFFER2. Any AHB access with this master port number is routed to this buffer. It must be ensured that the master IDs associated with all buffers must be different.

35.4.2.7 Buffer3 Configuration Register (QuadSPIx_BUF3CR)

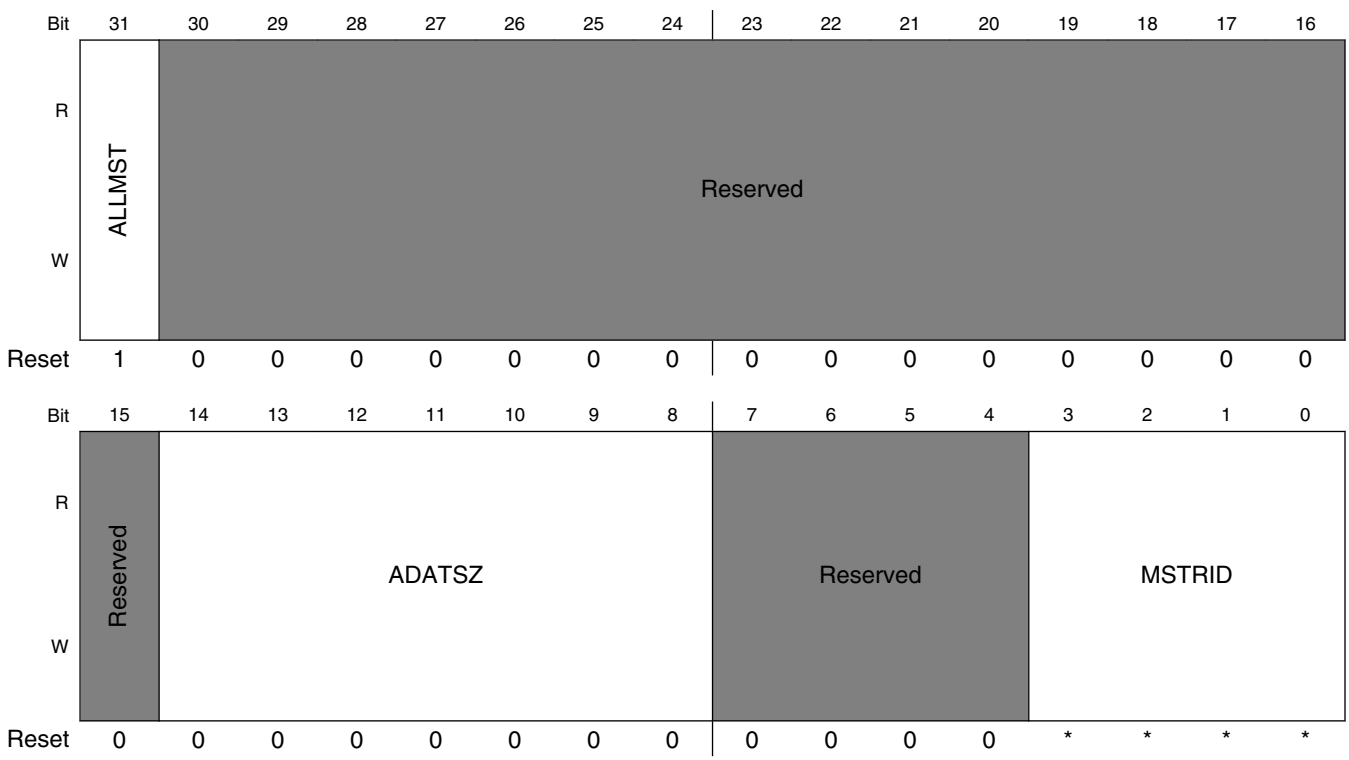
This register provides the configuration for any access to buffer3. An access is routed to buffer3 when the master port number of the incoming AHB request matches the MSTRID field of BUF3CR. Any buffer "miss" leads to the buffer being flushed a serial flash transaction being triggered as per the sequence pointed to by the SEQID field.

In the case that the ALLMST field is not set, any such transaction (where master port number does not match any of the MSTRID fields) will be returned an ERROR response.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 4005_A000h base + 1Ch offset = 4005_A01Ch



* Notes:

- MSTRID field: See the module configuration for the device specific reset value.

QuadSPIx_BUF3CR field descriptions

Field	Description
31 ALLMST	All master enable. When set, buffer3 acts as an all-master buffer. Any AHB access with a master port number not matching with the master ID of buffer0 or buffer1 or buffer2 is routed to buffer3. When set, the MSTRID field of this register is ignored.
30–16 Reserved	This field is reserved. Reserved.
15 Reserved	This field is reserved.
14–8 ADATSZ	AHB data transfer size Defines the data transfer size in 8 Bytes of an AHB triggered access to serial flash. When ADATSZ = 0, the data size mentioned the sequence pointed to by the SEQID field overrides this value. Software should ensure that this transfer size is not greater than the size of this buffer.
7–4 Reserved	This field is reserved.
MSTRID	Master ID. The ID of the AHB master associated with BUFFER3. Any AHB access with this master port number is routed to this buffer. It must be ensured that the master IDs associated with all buffers must be different.

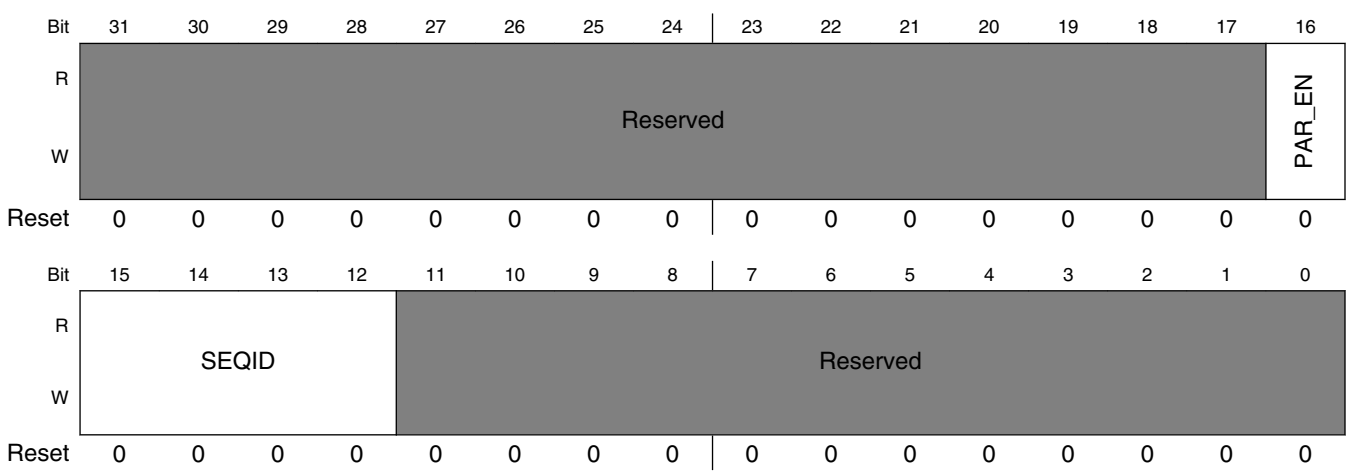
35.4.2.8 Buffer Generic Configuration Register (QuadSPIx_BFGENCR)

This register provides the generic configuration to any of the buffer accesses. Any buffer "miss" leads to the buffer being flushed and a serial flash transaction being triggered as per the sequence pointed to by the SEQID field. If the PAR_EN field is set, all the buffer accesses result in parallel accesses to the flashes.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 4005_A000h base + 20h offset = 4005_A020h



QuadSPIx_BFGENCR field descriptions

Field	Description
31–17 Reserved	This field is reserved.
16 PAR_EN	When set, a transaction to two serial flash devices is triggered in parallel mode. Refer to Parallel Flash Mode for more details.
15–12 SEQID	Points to a sequence in the Look-up-table. The SEQID defines the bits [6:2] of the LUT index. The bits [1:0] are always assumed to be 0. Refer to Look-up Table. **NOTE:** If the sequence pointer differs between the new and previous sequence then the user should reset this. See QSPI_SPTRCLR for more information.
Reserved	This field is reserved.

35.4.2.9 SOC Configuration Register (QuadSPIx_SOCCR)

This register is programmed at chip level for QuadSPI delay chain configuration. For details, refer to chip-specific QuadSPI information.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 4005_A000h base + 24h offset = 4005_A024h

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 W	SOCCFG																															
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

QuadSPIx_SOCCR field descriptions

Field	Description
SOCCFG	SOC Configuration For details, refer to chip-specific QuadSPI information.

35.4.2.10 Buffer0 Top Index Register (QuadSPIx_BUF0IND)

This register specifies the top index of buffer0, which defines its size. Note that that the 3 LSBs of this register are set to zero. This ensures that the buffer is 64-bit aligned, as each buffer entry is 64 bits long.

The register value should be set to the desired number of bytes less 8. For example, setting BUF0IND to 0 gives 8 bytes, 1 gives 16 bytes etc.

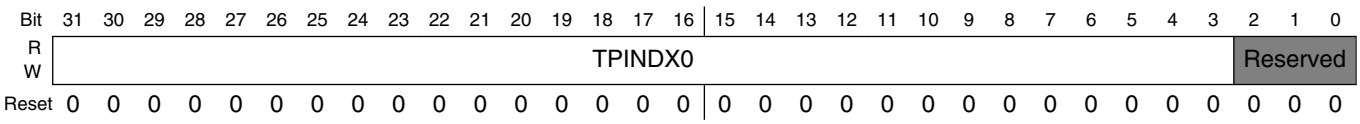
The size of buffer0 is the difference between BUF0IND+8 and 0.

It is the responsibility of the software to ensure that BUF0IND value is not greater than the overall size of the buffer. The hardware does not provide any protection against illegal programming.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 4005_A000h base + 30h offset = 4005_A030h



QuadSPIx_BUF0IND field descriptions

Field	Description
31–3 TPINDX0	Top index of buffer 0.
Reserved	This field is reserved. Reserved.

35.4.2.11 Buffer1 Top Index Register (QuadSPIx_BUF1IND)

This register specifies the top index of buffer1, which defines its size. Note that the 3 LSBs of this register are set to zero. This ensures that the buffer is 64-bit aligned as each buffer entry is 64 bits long.

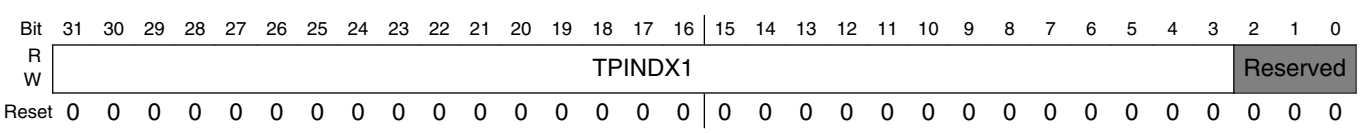
The size of buffer1 is the difference between BUF1IND and BUF0IND. The register value should be entered in bytes. For example, If BUF0IND = 0x100 then setting BUF1IND = 0x130 will set buffer1 size to 0x30 bytes.

It is the responsibility of the software to ensure that BUF1IND value is not greater than the overall size of the buffer. The hardware does not provide any protection against illegal programming.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 4005_A000h base + 34h offset = 4005_A034h



QuadSPIx_BUF1IND field descriptions

Field	Description
31–3 TPINDX1	Top index of buffer 1.
Reserved	This field is reserved.

35.4.2.12 Buffer2 Top Index Register (QuadSPIx_BUF2IND)

This register specifies the top index of buffer2, which defines its size. Note that that the 3 LSBs of this register are set to zero. This ensures that the buffer is 64-bit aligned as each buffer entry is 64 bits long.

The size of buffer2 is the difference between the BUF2IND and BUF1IND. The register value should be entered in bytes. For example, if BUF1IND = 0x130 then setting BUF2IND = 0x180 will set buffer2 size to 0x50 bytes.

It is the responsibility of the software to ensure that BUF2IND value is not greater than the overall size of the buffer. The hardware does not provide any protection against illegal programming.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 4005_A000h base + 38h offset = 4005_A038h

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 W	TPINDX2																													Reserved		
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

QuadSPIx_BUF2IND field descriptions

Field	Description
31–3 TPINDX2	Top index of buffer 2.
Reserved	This field is reserved.

35.4.2.13 Serial Flash Address Register (QuadSPIx_SFAR)

The module automatically translates this address on the memory map to the address on the flash itself. When operating in 24-bit mode, only bits 23-0 are sent to the flash. In 32-bit mode, bits 27-0 are used with bits 31-28 driven to 0 when QSPI_SFACR[CAS] is set to 0. Say, if QSPI_SFACR[CAS] is 3 then bits 26-3 are sent to flash as it page address in case flash is operating in 24-bit mode. Total number of address bits request by flash as it page and column address must not be more than 32 bit. Refer to Table 35-13 for the mapping between the access mode and the QSPI_SFAR content and to Normal Mode for details about the command triggering and command execution. The software should ensure that the serial flash address provided in the QSPI_SFAR register lies in the valid flash address range as defined in Table 35-13.

Write:

- *QSPI_SR[IP_ACC] = 0*

Address: 4005_A000h base + 100h offset = 4005_A100h

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 W	SFADR																															
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

QuadSPIx_SFAR field descriptions

Field	Description
SFADR	Serial Flash Address. The register content is used as byte address for all following IP Commands.

35.4.2.14 Serial Flash Address Configuration Register (QuadSPIx_SFACR)

This register contains the serial flash specific address requirements that must be configured according to the flash connected, for the controller to function properly. The module automatically translates the address QSPI_SFAR on the memory map or the incoming address on the AHB bus to the column address on the flash itself. Say, a flash needs 3 bits as its column address than only the lower 3 bits of QSPI_SFAR/AHB address are send to flash as its column address. The software should ensure that the serial flash address provided in the QSPI_SFAR register or the incoming AHB address lies in the valid flash address range.

Write:

- *QSPI_SR[IP_ACC] = 0*
- *QSPI_SR[AHB_ACC] = 0*

Address: 4005_A000h base + 104h offset = 4005_A104h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															WA
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 W	Reserved												CAS			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QuadSPIx_SFACR field descriptions

Field	Description
31–17 Reserved	This field is reserved.
16 WA	Word Addressable Defines whether the serial flash is a byte addressable flash or a word addressable flash. According to this bit configuration the address is re-mapped to the flash interface. Refer to Address scheme for details. 0 Byte addressable serial flash mode. 1 Word (2 byte) addressable serial flash mode.
15–4 Reserved	This field is reserved.
CAS	Column Address Space Defines the width of the column address. If the coulmn address is say [2:0] of QSPI_SFAR/AHB address, then CAS must be 3. If there is no column address separation in any serial flash this bit must be programmed to 0.

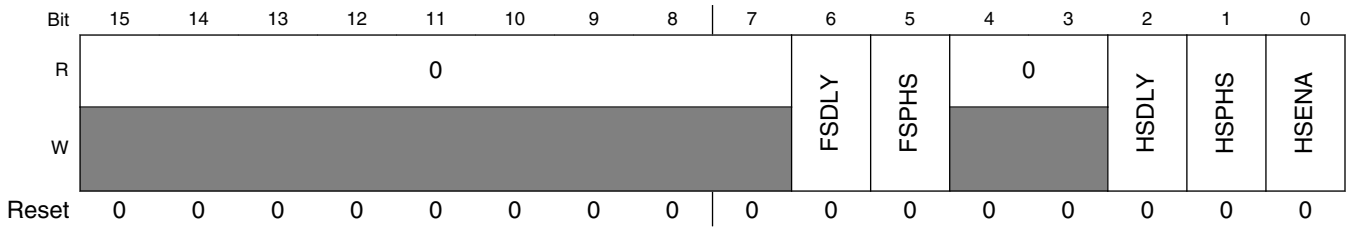
35.4.2.15 Sampling Register (QuadSPIx_SMPR)

The Sampling Register allows configuration of how the incoming data from the external serial flash devices are sampled in the QuadSPI module.

Write: Disabled Mode

Address: 4005_A000h base + 108h offset = 4005_A108h

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



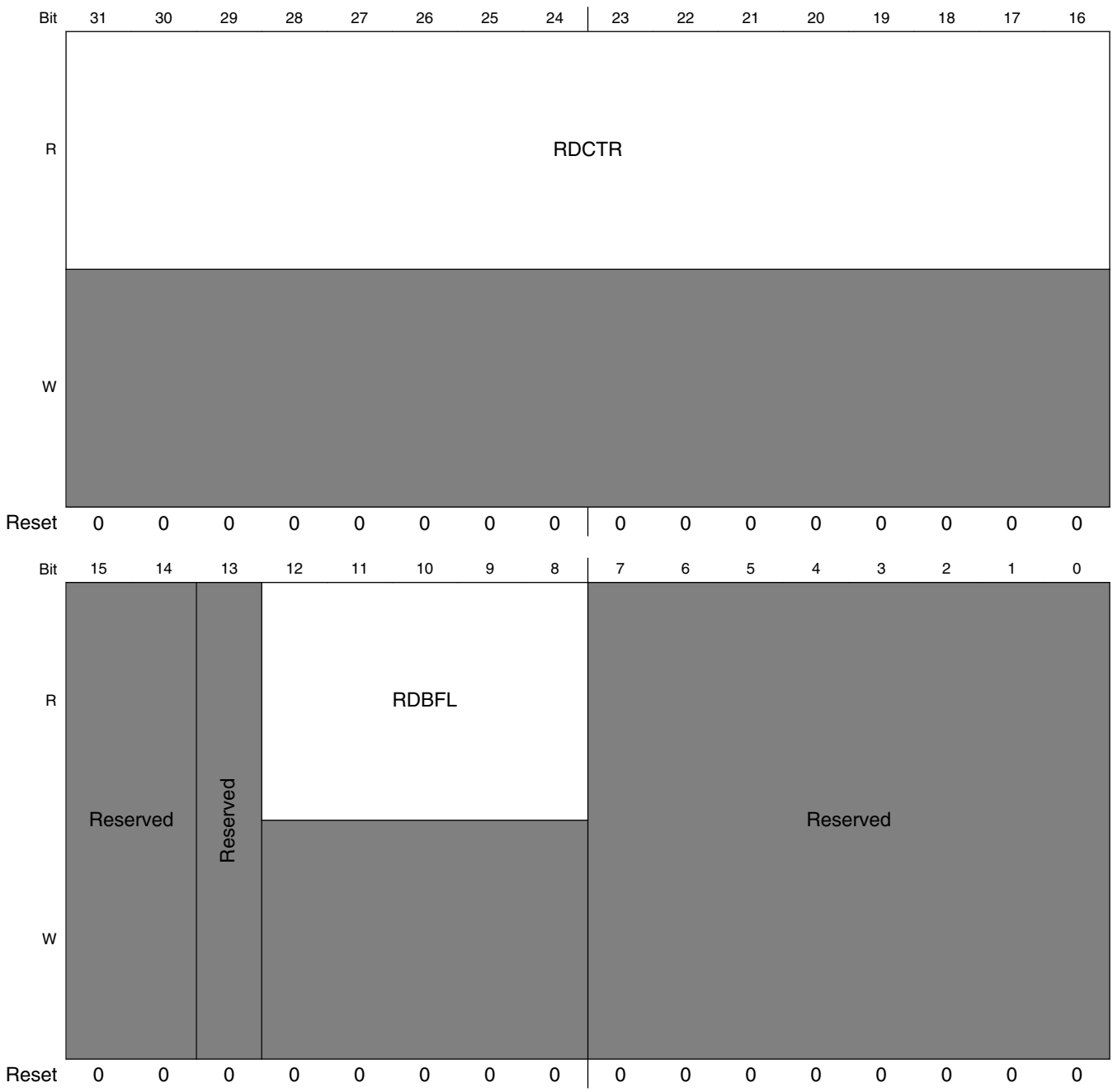
QuadSPIx_SMPR field descriptions

Field	Description
31–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–16 DDRSMP	DDR Sampling point Select the sampling point for incoming data when serial flash is executing a DDR instruction. Refer to Figure 35-13 for details on the sampling points.
15–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 FSDLY	Full Speed Delay selection for SDR instructions. Select the delay with respect to the reference edge for the sample point valid for full speed commands. 0 One clock cycle delay 1 Two clock cycles delay. **NOTE:** This bit is also used in DQS mode and ignored when using non-DQS DDR instructions.
5 FSPHS	Full Speed Phase selection for SDR instructions. Select the edge of the sampling clock valid for full speed commands. 0 Select sampling at non-inverted clock 1 Select sampling at inverted clock. **NOTE:** This bit is also used in DQS mode and ignored when using non-DQS DDR instructions.
4–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 HSDLY	Half Speed Delay selection for SDR instructions. Only relevant when HSENA bit is set. Select the delay with respect to the reference edge for the sample point valid for half speed commands. 0 One clock cycle delay 1 Two clock cycle delay
1 HSPHS	Half Speed Phase selection for SDR instructions. Only relevant when HSENA bit is set. Select the delay with respect to the reference edge for the sample point valid for half speed commands. 0 Select sampling at non-inverted clock 1 Select sampling at inverted clock
0 HSENA	Half Speed serial flash clock Enable This bit enables the divide by 2 of the clock to the external serial flash device for all commands, only in SDR. Refer to Serial Flash Clock Frequency Limitations for details. 0 Disable divide by 2 of serial flash clock for half speed commands 1 Enable divide by 2 of serial flash clock for half speed commands

35.4.2.16 RX Buffer Status Register (QuadSPIx_RBSR)

This register contains information related to the receive data buffer.

Address: 4005_A000h base + 10Ch offset = 4005_A10Ch



QuadSPIx_RBSR field descriptions

Field	Description
31–16 RDCTR	Read Counter. Indicates how many entries of 4 bytes have been removed from the RX Buffer. For example, a value of 0x2 would indicate 8 bytes have been removed. It is incremented by the number (QSPI_RBCT[WMRK] + 1) on RX Buffer POP event. The RX Buffer can be popped using DMA or pop flag QSPI_FR[RBDF]. The QSPI_RSER[RBDDE] defines which pop has to be done. For further details, refer to AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB15) and "Data Transfer from the QuadSPI Module Internal Buffers" section in Flash Read.
15–14 Reserved	This field is reserved.
13 Reserved	This field is reserved.
12–8 RDBFL	RX Buffer Fill Level. Indicates how many entries of 4 bytes are still available in the RX Buffer.For example, a value of 0x2 would indicate 8bytes are available.
Reserved	This field is reserved.

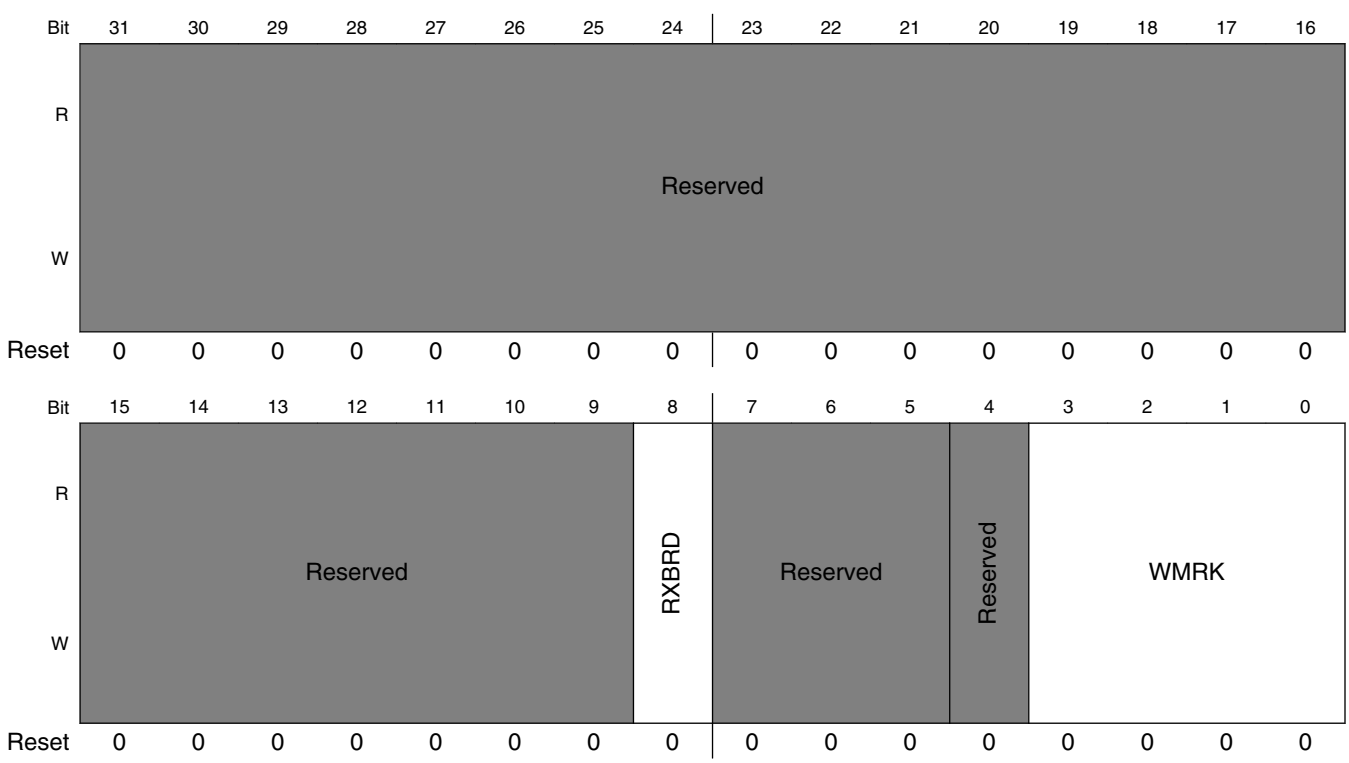
35.4.2.17 RX Buffer Control Register (QuadSPIx_RBCT)

This register contains control data related to the receive data buffer.

Write:

- *QSPI_SR[IP_ACC] = 0*

Address: 4005_A000h base + 110h offset = 4005_A110h



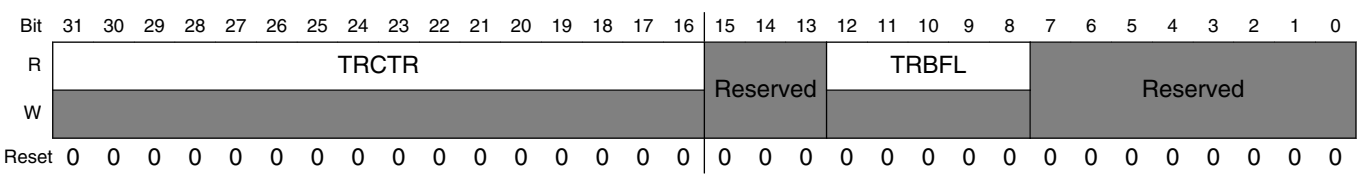
QuadSPIx_RBCT field descriptions

Field	Description
31–9 Reserved	This field is reserved.
8 RXBRD	RX Buffer Readout. This field specifies the access scheme for the RX Buffer readout. 0 RX Buffer content is read using the AHB Bus registers QSPI_ARDB0 to QSPI_ARDB15. For details, refer to Exclusive Access to Serial Flash for AHB Commands. 1 RX Buffer content is read using the IP Bus registers QSPI_RBDR0 to QSPI_RBDR15.
7–5 Reserved	This field is reserved.
4 Reserved	This field is reserved.
WMRK	RX Buffer Watermark. This field determines when the readout action of the RX Buffer is triggered. When the number of valid entries in the RX Buffer is equal to or greater than the number given by (WMRK+1) the QSPI_SR[RXWE] flag is asserted.The value should be entered as the number of 4-byte entries minus 1. For example, a value of 0x0 would set the watermark to 4 bytes, 1 to 8 bytes, 2 to 12 bytes, and so on. For details, refer to DMA Usage.

35.4.2.18 TX Buffer Status Register (QuadSPIx_TBSR)

This register contains information related to the transmit data buffer.

Address: 4005_A000h base + 150h offset = 4005_A150h



QuadSPIx_TBSR field descriptions

Field	Description
31–16 TRCTR	Transmit Counter. This field indicates how many entries of 4 bytes have been written into the TX Buffer by host accesses. It is reset to 0 when a 1 is written to QSPI_MCR[CLR_TXF]. It is incremented on each write access to the QSPI_TBDR register when another word has been pushed onto the TX Buffer. When it is not cleared the TRCTR field wraps around to 0. Refer to TX Buffer Data Register (QuadSPI_TBDR) for details.
15–13 Reserved	This field is reserved.
12–8 TRBFL	TX Buffer Fill Level. The TRBFL field contains the number of entries of 4 bytes each available in the TX Buffer for the QuadSPI module to transmit to the serial flash device.
Reserved	This field is reserved.

35.4.2.19 TX Buffer Data Register (QuadSPIx_TBDR)

The QSPI_TBDR register provides access to the circular TX Buffer of depth 64 bytes. This buffer provides the data written into it as write data for the page programming commands to the serial flash device. Refer to Table 35-22 for the byte ordering scheme. A write transaction on the flash with data size of less than 32 bits will lead to the removal of four data entry from the TX buffer. The valid bits will be used and the rest of the bits will be discarded.

Write:

- *QSPI_SR[TXFULL] = 0*

32-bit write access required

Address: 4005_A000h base + 154h offset = 4005_A154h

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 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

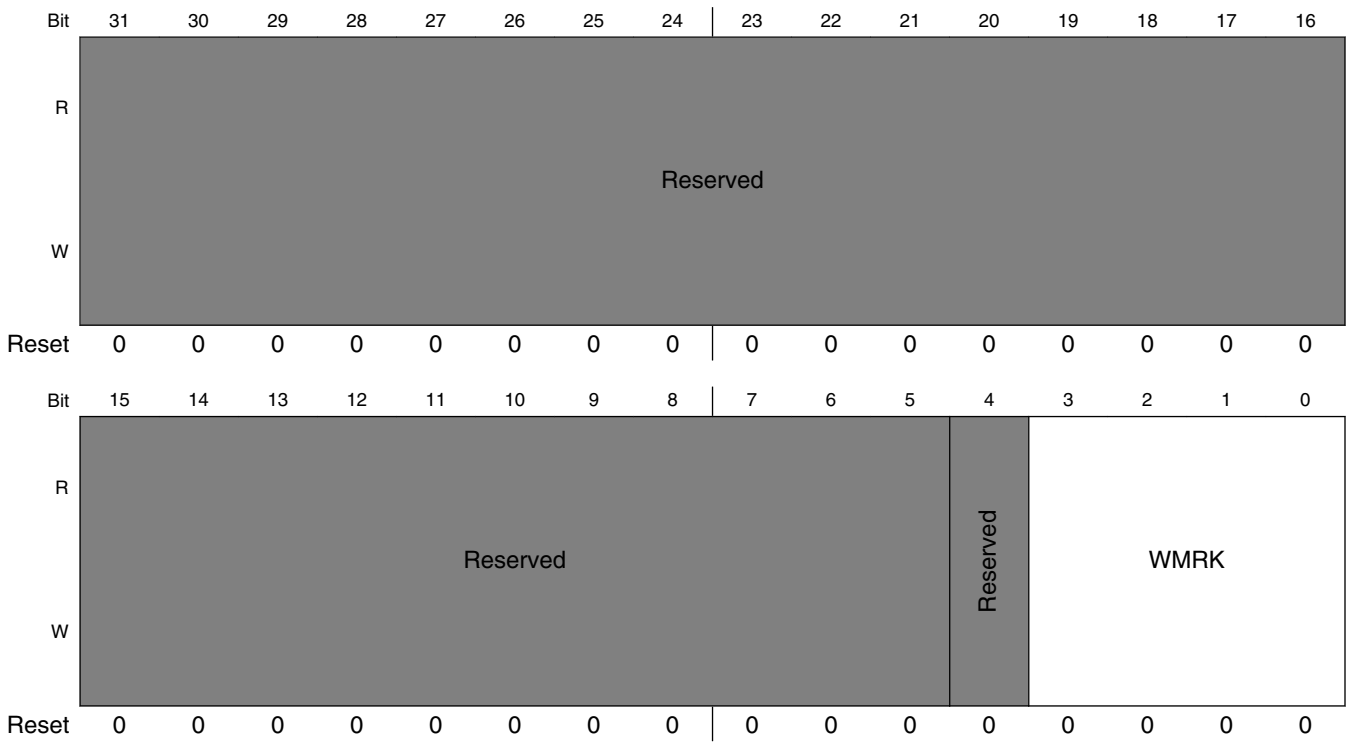
QuadSPIx_TBDR field descriptions

Field	Description
TXDATA	TX Data On write access the data is written into the next available entry of the TX Buffer and the QPSI_TBSR[TRBFL] field is updated accordingly. On a read access, the last data written to the register is returned.

35.4.2.20 Tx Buffer Control Register (QuadSPIx_TBCT)

This register contains control information for transmit data buffer.

Address: 4005_A000h base + 158h offset = 4005_A158h



QuadSPIx_TBCT field descriptions

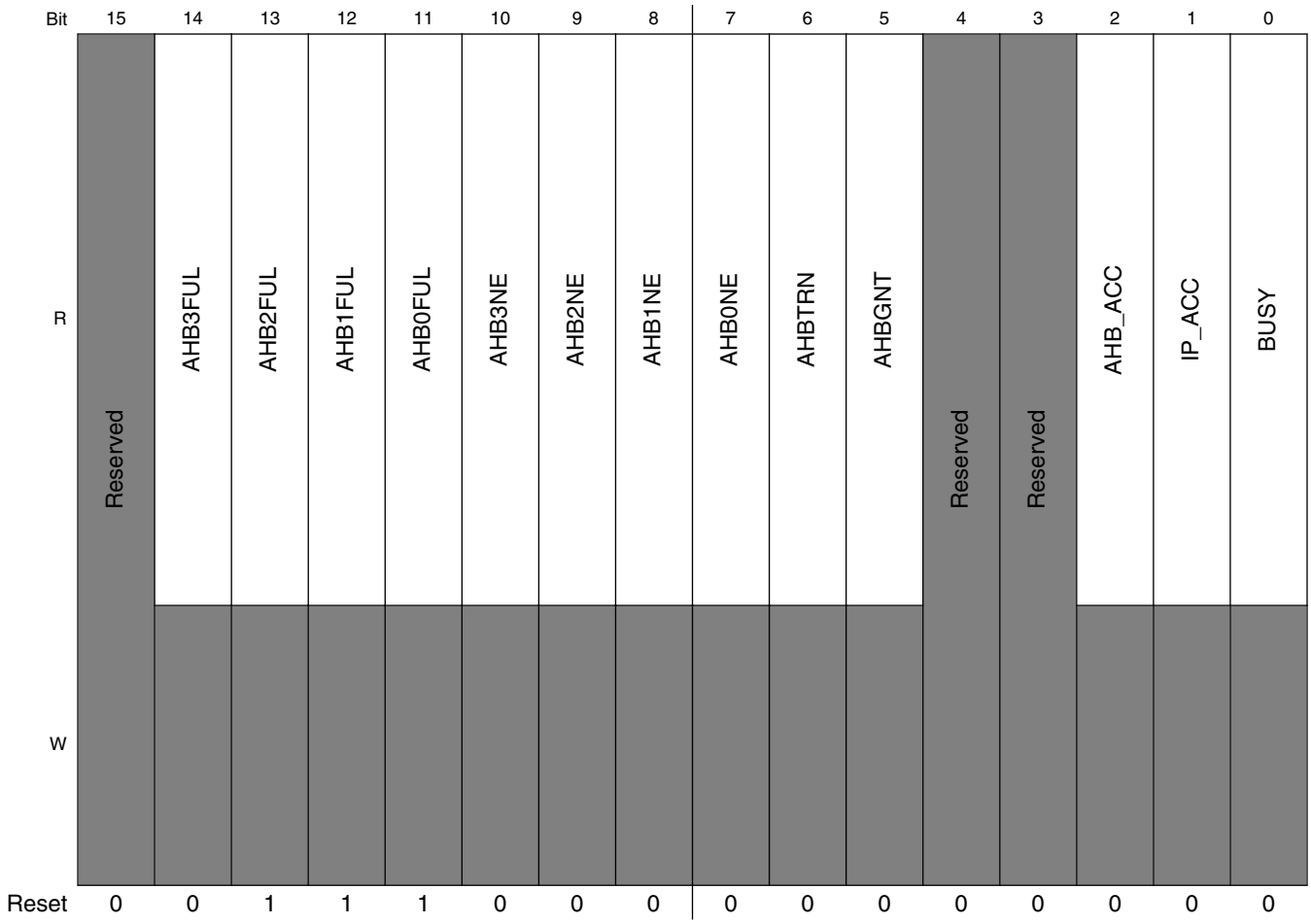
Field	Description
31–5 Reserved	This field is reserved.
4 Reserved	This field is reserved.
WMRK	Determines the watermark for the TX Buffer. When the number of available space in TX Buffer is greater than the number given by (WMRK+1), QSPI_SR[TXWA] is asserted. The values should be entered as the number of 4Bytes entries minus 1. For example, a value of 0x0 would set the watermark to 4 bytes, 1 to 8 bytes, 2 to 12 Bytes, and so on.For details, refer to DMA Usage.

35.4.2.21 Status Register (QuadSPIx_SR)

The QSPI_SR register provides all available status information about SFM command execution and arbitration, the RX Buffer, TX Buffer, and the AHB Buffer.

Address: 4005_A000h base + 15Ch offset = 4005_A15Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DLPSMP			Reserved	TXFULL	TXDMA	TXWA	TXEDA	RXDMA	Reserved			RXFULL	Reserved		RXWE
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0



QuadSPIx_SR field descriptions

Field	Description
31–29 DLPSMP	Data learning pattern sampling point. The sampling point found by the controller with the data learning pattern. • This is used for DDR only. • If the learning fails, this field will return garbage and DLPFF bit will be set. • In case of Data learning with DQS this field will return the reset value as sampling point match is not found in case of DQS. For details, refer to Data Learning.
28 Reserved	This field is reserved.
27 TXFULL	TX Buffer Full. Asserted when no more data can be stored.
26 TXDMA	TXDMA Asserted when TXFIFO fill via DMA is active i.e. DMA is requested or running
25 TXWA	TX Buffer watermark Available Asserted when the number of available spaces in TX buffer is greater than or equal to the value give by QSPI_TBCT[WMRK].

Table continues on the next page...

QuadSPIx_SR field descriptions (continued)

Field	Description
24 TXEDA	Tx Buffer Enough Data Available Asserted when TX Buffer contains enough data for any pop operation to take place. There must be at least 128 bit data available in TX FIFO for any pop operation; otherwise, QSPI_FR[TBUF] will be set.
23 RXDMA	RX Buffer DMA. Asserted when RX Buffer read out via DMA is active i.e DMA is requested or running.
22–20 Reserved	This field is reserved.
19 RXFULL	RX Buffer Full. Asserted when the RX Buffer is full, i.e. that QSPI_RBSR[RDBFL] field is equal to 32.
18–17 Reserved	This field is reserved.
16 RXWE	RX Buffer Watermark Exceeded. Asserted when the number of valid entries in the RX Buffer exceeds the number given in the QSPI_RBCT[WMRK] field.
15 Reserved	This field is reserved.
14 AHB3FUL	AHB 3 Buffer Full. Asserted when AHB 3 buffer is full.
13 AHB2FUL	AHB 2 Buffer Full. Asserted when AHB 2 buffer is full.
12 AHB1FUL	AHB 1 Buffer Full. Asserted when AHB 1 buffer is full.
11 AHB0FUL	AHB 0 Buffer Full. Asserted when AHB 0 buffer is full.
10 AHB3NE	AHB 3 Buffer Not Empty. Asserted when AHB 3 buffer contains data.
9 AHB2NE	AHB 2 Buffer Not Empty. Asserted when AHB 2 buffer contains data.
8 AHB1NE	AHB 1 Buffer Not Empty. Asserted when AHB 1 buffer contains data.
7 AHB0NE	AHB 0 Buffer Not Empty. Asserted when AHB 0 buffer contains data.
6 AHBTRN	AHB Access Transaction pending. Asserted when there is a pending request on the AHB interface. Refer to the AMBA specification for details.
5 AHBGNT	AHB Command priority Granted: Asserted when another module has been granted priority of AHB Commands against IP Commands. For details refer to Command Arbitration.
4 Reserved	This field is reserved.
3 RESERVED	This field is reserved.
2 AHB_ACC	AHB Access. Asserted when the transaction currently executed was initiated by AHB bus.
1 IP_ACC	IP Access. Asserted when transaction currently executed was initiated by IP bus.
0 BUSY	Module Busy. Asserted when module is currently busy handling a transaction to an external flash device.

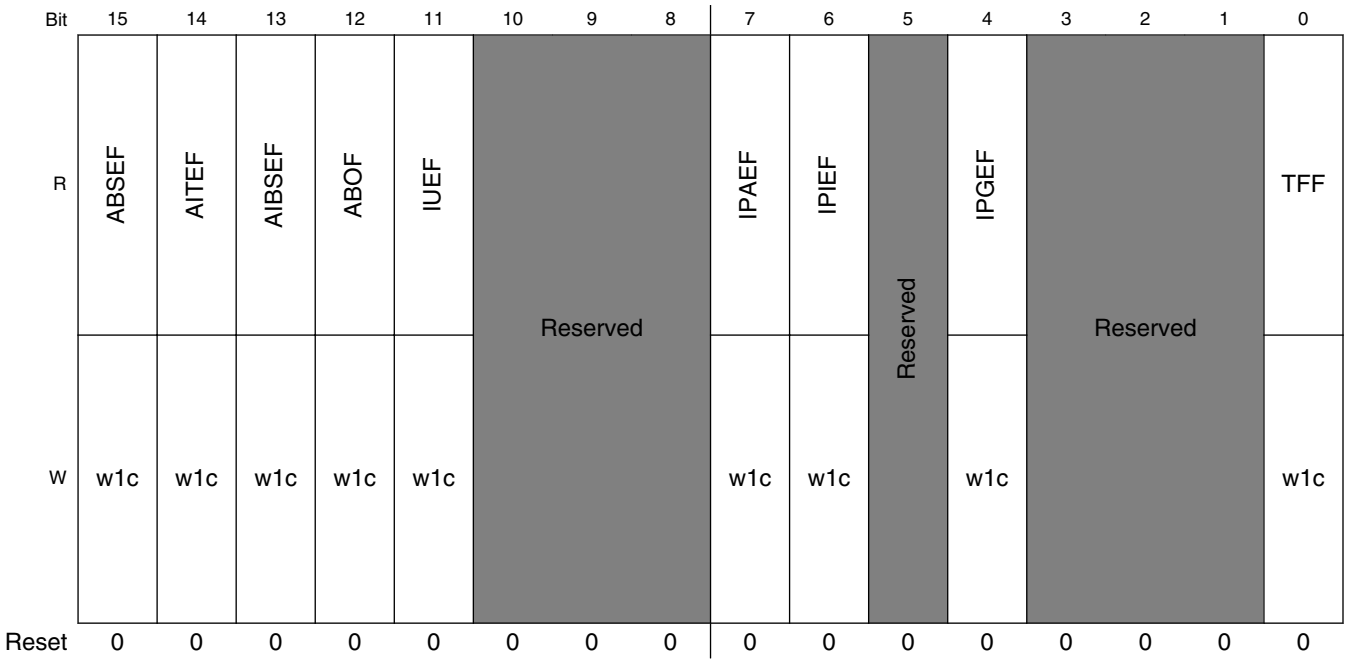
35.4.2.22 Flag Register (QuadSPIx_FR)

The QSPI_FR register provides all available flags about SFM command execution and arbitration which may serve as source for the generation of interrupt service requests. Note that the error flags in this register do not relate directly to the execution of the transaction in the serial flash device itself but only to the behavior and conditions visible in the QuadSPI module.

Write: Enabled Mode

Address: 4005_A000h base + 160h offset = 4005_A160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DLPFF	Reserved		Reserved	TBFF	TBUF	Reserved		ILLINE	Reserved					RBOF	RBDF
W	w1c				w1c	w1c			w1c						w1c	w1c
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0



QuadSPIx_FR field descriptions

Field	Description
31 DLPFF	Data Learning Pattern Failure Flag. Set when DATA_LEARN instruction was encountered in a sequence but no sampling point was found for the data learning patternin case only 8 bit data learning is requested for non DQS mode. The controller automatically starts sampling using the value in QSPI_SMPR[DDRSMP].If more than 8 bits data learning are requested with QSPI_MCR[DQS_EN] set to 0, and the sampling point found after first 8 bit match doesn't remain the same for the whole instruction duration, this flag is set. In case of Data learn with DQS this flag is set whenever the incoming data from flash on DQS edges doesn't match the pattern in QSPI_DLPR. For details, refer to Data Learning.
30–29 RESERVED	This field is reserved.
28 Reserved	This field is reserved.
27 TBFF	TX Buffer Fill Flag. Before writing to the TX buffer, this bit should be cleared. Then this bit has to be read back. If the bit is set, the TX Buffer can take more data. If the bit remains cleared, the TX buffer is full. Refer to Tx Buffer Operation for details.
26 TBUF	TX Buffer Underrun Flag. Set when the module tried to pull data although TX Buffer was emptyor the buffer contains less than 128 bits of data. The application must ensure that the buffer never goes empty during a transaction except for the last data fetch. The IP Command leading to the TX Buffer underrun is continued (data sent to the serial flash device is all F in case of valid TX underrun. The application must clear the TX Buffer in response to this event by writing a 1 to the QSPI_MCR[CLR_TXF] bit.
25–24 Reserved	This field is reserved.
23 ILLINE	Illegal Instruction Error Flag. Set when an illegal instruction is encountered by the controller in any of the sequences. Refer to Table 35-20 for a list of legal instructions.
22–18 Reserved	This field is reserved.
17 RBOF	RX Buffer Overflow Flag. Set when not all the data read from the serial flash device could be pushed into the RX Buffer.

Table continues on the next page...

QuadSPIx_FR field descriptions (continued)

Field	Description
	The IP Command leading to this condition is continued until the number of bytes according to the QSPI_IPCR[IDATSZ] field has been read from the serial flash device. The content of the RX Buffer is not changed.
16 RBDF	RX Buffer Drain Flag. Will be set if the QuadSPI_SR[RXWE] status bit is asserted. Writing 1 into this bit triggers one of the following actions: • If the RX Buffer has up to QuadSPI_RBCT[WMRK] valid entries then the flag is cleared. • If the RX Buffer has more than QuadSPI_RBCT[WMRK] valid entries and the QuadSPI_RSER[RBDDE] bit is not set (flag driven mode) a RX Buffer POP event is triggered. The flag remains set if the RX Buffer contains more than QuadSPI_RBCT[WMRK] valid entries after the RX Buffer POP event is finished. The flag is cleared if the RX Buffer contains less than or equal to QuadSPI_RBCT[WMRK] valid entries after the RX Buffer POP event is finished. Refer to "Receive Buffer Drain Interrupt or DMA Request" section in Normal Mode Interrupt and DMA Requests, for details.
15 ABSEF	AHB Sequence Error Flag. Set when the execution of an AHB Command is started with a WRITE or WRITE_DDR Command in the sequence pointed to by the QSPI_BUFxCR register. (QSPI_BUFxCR implies any one of QSPI_BUF0CR/QSPI_BUF1CR/QSPI_BUF2CR/QSPI_BUF3CR.) Communication with the serial flash device is terminated before the execution of WRITE/WRITE_DDR command by the QuadSPI module. The AHB bus request which triggered this command is answered with an ERROR response.
14 AITEF	AHB Illegal transaction error flag. Set whenever there is no response generated from QSPI to AHB bus in case of illegal transaction and the watchdog timer expires.The timer value is taken as parameter.
13 AIBSEF	AHB Illegal Burst Size Error Flag. Set whenever the total burst size (size x beat) of an AHB transaction is greater than the prefetch data size. The prefecth data size is defined by QSPI_BUFxCR[ADATSZ] or data size mentioned in the sequence pointed to by the SEQID field in case ADATSZ = 0. Refer to HBURST Support for more details on HBURST feature.
12 ABOF	AHB Buffer Overflow Flag. Set when the size of the AHB access exceeds the size of the AHB buffer. This condition can occur only if the QSPI_BUFxCR[ADATSZ] field is programmed incorrectly. The AHB Command leading to this condition is continued until the number of entries according to the QSPI_BUFxCR[ADATSZ] field has been read from the serial flash device. The content of the AHB Buffer is not changed.
11 IUEF	IP Command Usage Error Flag. Set when in parallel flash mode the execution of an IP Command is started with more than one pad enabled and the sequence pointed to by the sequence ID contains a WRITE or a WRITE_DDR command. Refer to Table 35-20 table for the related commands. Communication with the serial flash device is terminated before the execution of WRITE/WRITE_DDR command by the QuadSPI module.
10–8 Reserved	This field is reserved.
7 IPAEF	IP Command Trigger during AHB Access Error Flag. Set when the following condition occurs: • A write access occurs to the QSPI_IPCR[SEQID] field and the QSPI_SR[AHB_ACC] bit is set. Any command leading to the assertion of the IPAEF flag is ignored.
6 IPIEF	IP Command Trigger could not be executed Error Flag. Set when the QSPI_SR[IP_ACC] bit is set (i.e. an IP triggered command is currently executing) and any of the following conditions occurs: • Write access to the QSPI_IPCR register. Any command leading to the assertion of the IPIEF flag is ignored

Table continues on the next page...

QuadSPIx_FR field descriptions (continued)

Field	Description
	• Write access to the QSPI_SFAR register. • Write access to the QSPI_RBCT register.
5 Reserved	This field is reserved.
4 IPGEF	IP Command Trigger during AHB Grant Error Flag. Set when the following condition occurs: • A write access occurs to the QSPI_IPCR[SEQID] field and the QSPI_SR[AHBGNT] bit is set. Any command leading to the assertion of the IPGEF flag is ignored.
3–1 Reserved	This field is reserved.
0 TFF	IP Command Transaction Finished Flag. Set when the QuadSPI module has finished a running IP Command. If an error occurred the related error flags are valid, at the latest, in the same clock cycle when the TFF flag is asserted.

35.4.2.23 Interrupt and DMA Request Select and Enable Register (QuadSPIx_RSER)

The QuadSPI_RSER register provides enables and selectors for the interrupts in the QuadSPI module.

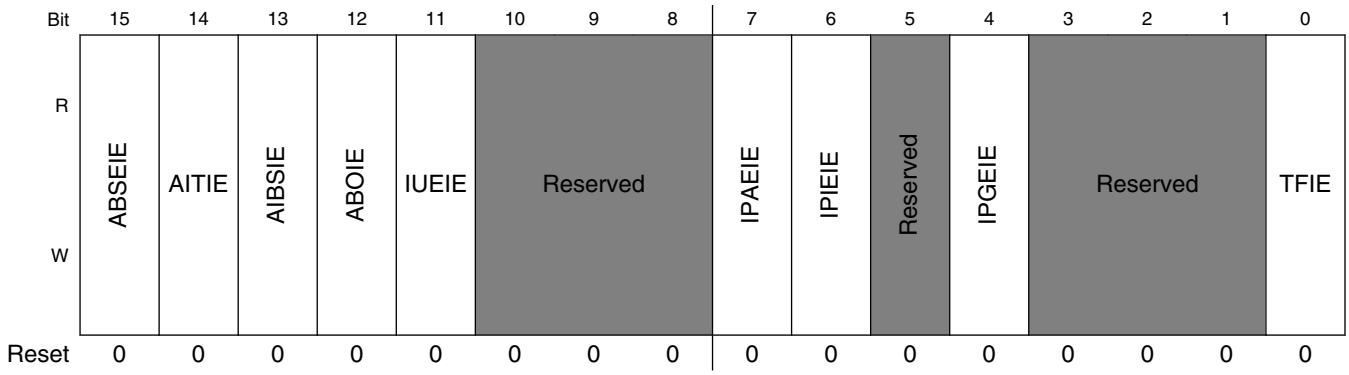
NOTE

Each flag of the QuadSPI_FR register enabled as source for an interrupt prevents the QuadSPI module from entering Stop Mode or Module Disable Mode when this flag is set.

Write: Anytime

Address: 4005_A000h base + 164h offset = 4005_A164h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DLPFIE	Reserved		Reserved	TBFIE	TBUIE	TBFDE	Reserved	ILLINIE	Reserved	RBDDE	Reserved			RBOIE	RBDIE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



QuadSPIx_RSER field descriptions

Field	Description
31 DLPFIE	Data Learning Pattern Failure Interrupt enable . Triggered by DLPFF flag in QSPI_FR register 0 No DLPFF interrupt will be generated 1 DLPFF interrupt will be generated
30–29 Reserved	This field is reserved.
28 RESERVED	This field is reserved.
27 TBFIE	TX Buffer Fill Interrupt Enable 0 No TBFF interrupt will be generated 1 TBFF interrupt will be generated
26 TBUIE	TX Buffer Underrun Interrupt Enable 0 No TBUF interrupt will be generated 1 TBUF interrupt will be generated
25 TBFDE	TX Buffer Fill DMA Enable Enables generation of DMA requests for TX Buffer fill. When this is set DMA requests are generated as long as the QSPI_SR[TXWA] status bit is set. 0 No DMA request will be generated 1 DMA request will be generated
24 Reserved	This field is reserved.
23 ILLINIE	Illegal Instruction Error Interrupt Enable. Triggered by ILLINE flag in QSPI_FR 0 No ILLINE interrupt will be generated 1 ILLINE interrupt will be generated
22 Reserved	This field is reserved.
21 RBDDE	RX Buffer Drain DMA Enable: Enables generation of DMA requests for RX Buffer Drain. When this bit is set DMA requests are generated as long as the QSPI_SR[RXWE] status bit is set. 0 No DMA request will be generated 1 DMA request will be generated

Table continues on the next page...

QuadSPIx_RSER field descriptions (continued)

Field	Description
20–18 Reserved	This field is reserved.
17 RBOIE	RX Buffer Overflow Interrupt Enable 0 No RBOF interrupt will be generated 1 RBOF interrupt will be generated
16 RBDIE	RX Buffer Drain Interrupt Enable: Enables generation of IRQ requests for RX Buffer Drain. When this bit is set the interrupt is asserted as long as the QuadSPI_SR[RBDF] flag is set. 0 No RBDF interrupt will be generated 1 RBDF Interrupt will be generated
15 ABSEIE	AHB Sequence Error Interrupt Enable: Triggered by ABSEF flags of QSPI_FR 0 No ABSEF interrupt will be generated 1 ABSEF interrupt will be generated
14 AITIE	AHB Illegal transaction interrupt enable. 0 No AITEF interrupt will be generated 1 AITEF interrupt will be generated
13 AIBSIE	AHB Illegal Burst Size Interrupt Enable 0 No AIBSEF interrupt will be generated 1 AIBSEF interrupt will be generated
12 ABOIE	AHB Buffer Overflow Interrupt Enable 0 No ABOF interrupt will be generated 1 ABOF interrupt will be generated
11 IUEIE	IP Command Usage Error Interrupt Enable 0 No IUEF interrupt will be generated 1 IUEF interrupt will be generated
10–8 Reserved	This field is reserved.
7 IPAEIE	IP Command Trigger during AHB Access Error Interrupt Enable 0 No IPAEF interrupt will be generated 1 IPAEF interrupt will be generated
6 IPIEIE	IP Command Trigger during IP Access Error Interrupt Enable 0 No IPIEF interrupt will be generated 1 IPIEF interrupt will be generated
5 Reserved	This field is reserved.
4 IPGEIE	IP Command Trigger during AHB Grant Error Interrupt Enable 0 No IPGEF interrupt will be generated 1 IPGEF interrupt will be generated
3–1 Reserved	This field is reserved. Reserved.

Table continues on the next page...

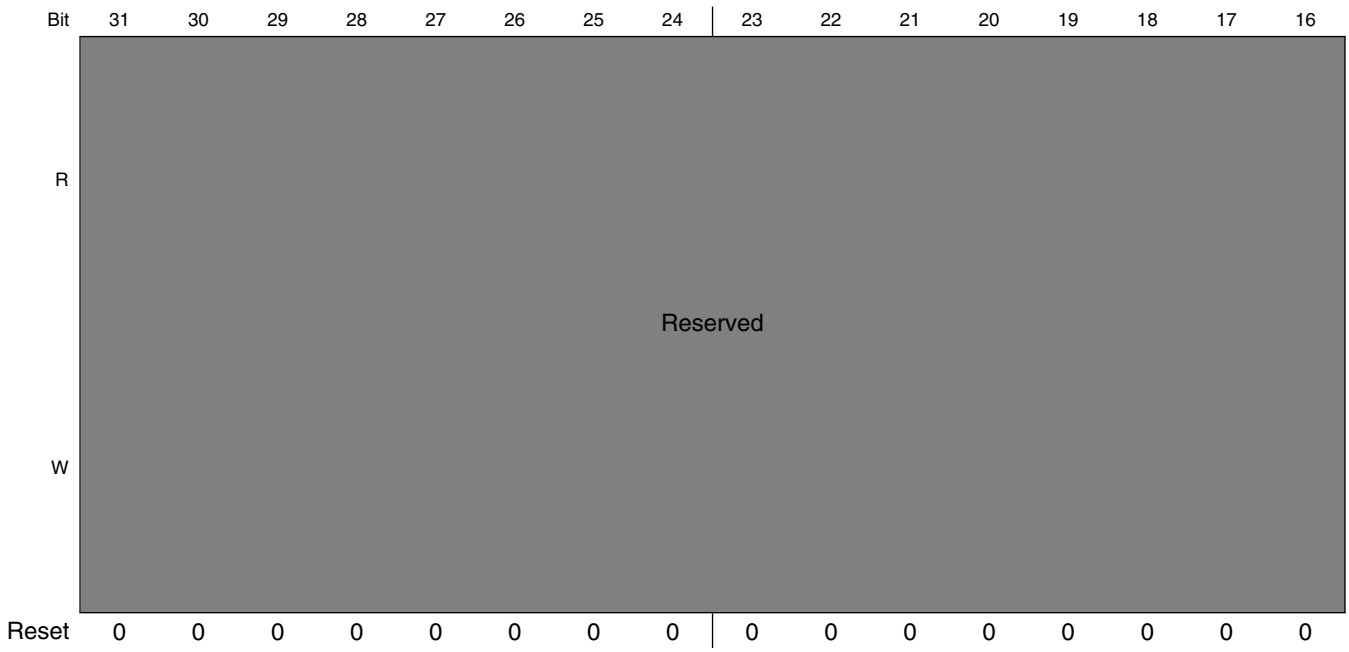
QuadSPIx_RSER field descriptions (continued)

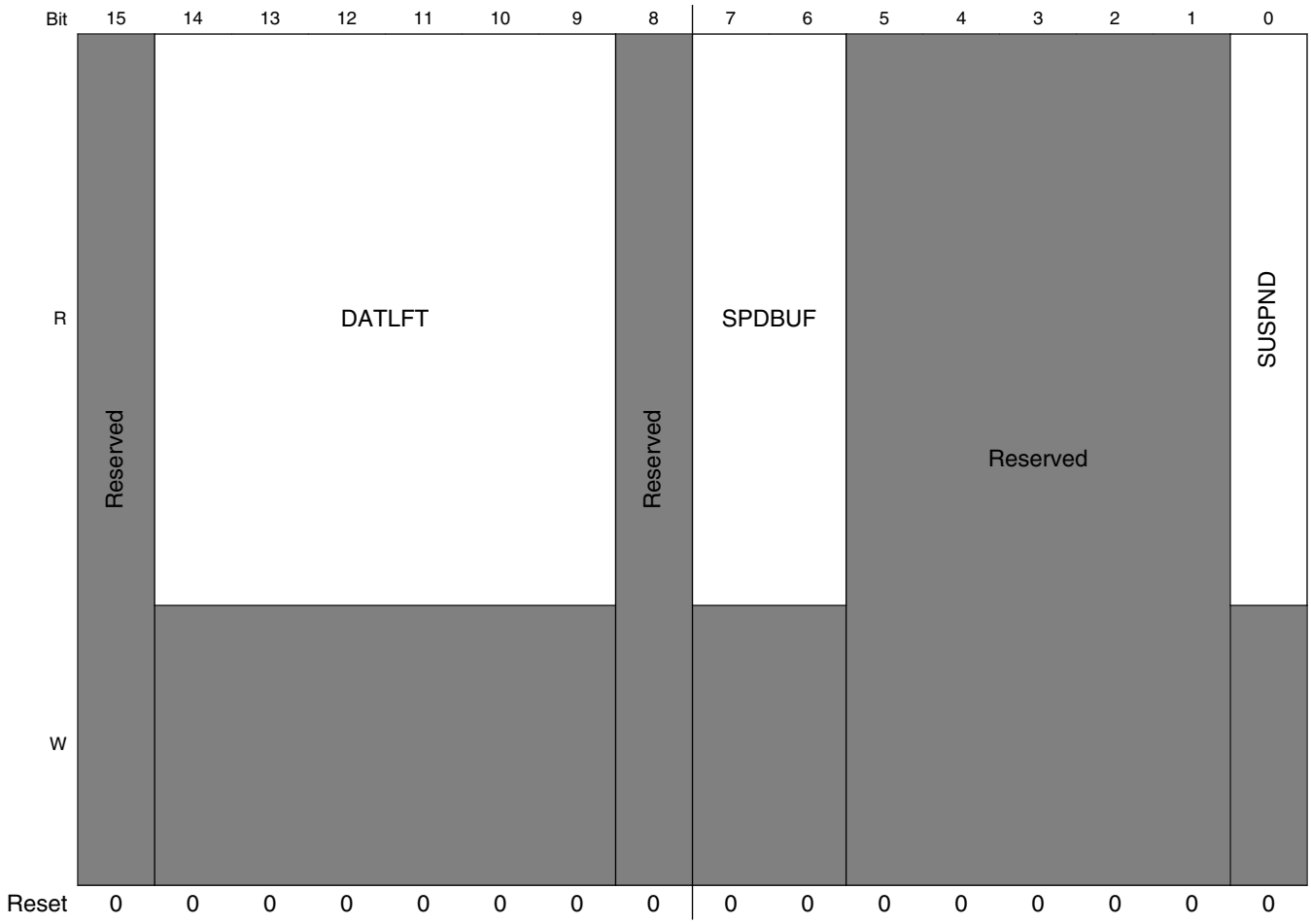
Field	Description
0 TFIE	Transaction Finished Interrupt Enable 0 No TFF interrupt will be generated 1 TFF interrupt will be generated

35.4.2.24 Sequence Suspend Status Register (QuadSPIx_SPNDST)

The sequence suspend status register provides information specific to any suspended sequence. An AHB sequence may be suspended when a high priority AHB master makes an access before the AHB sequence completes the data transfer requested.

Address: 4005_A000h base + 168h offset = 4005_A168h





QuadSPIx_SPNDST field descriptions

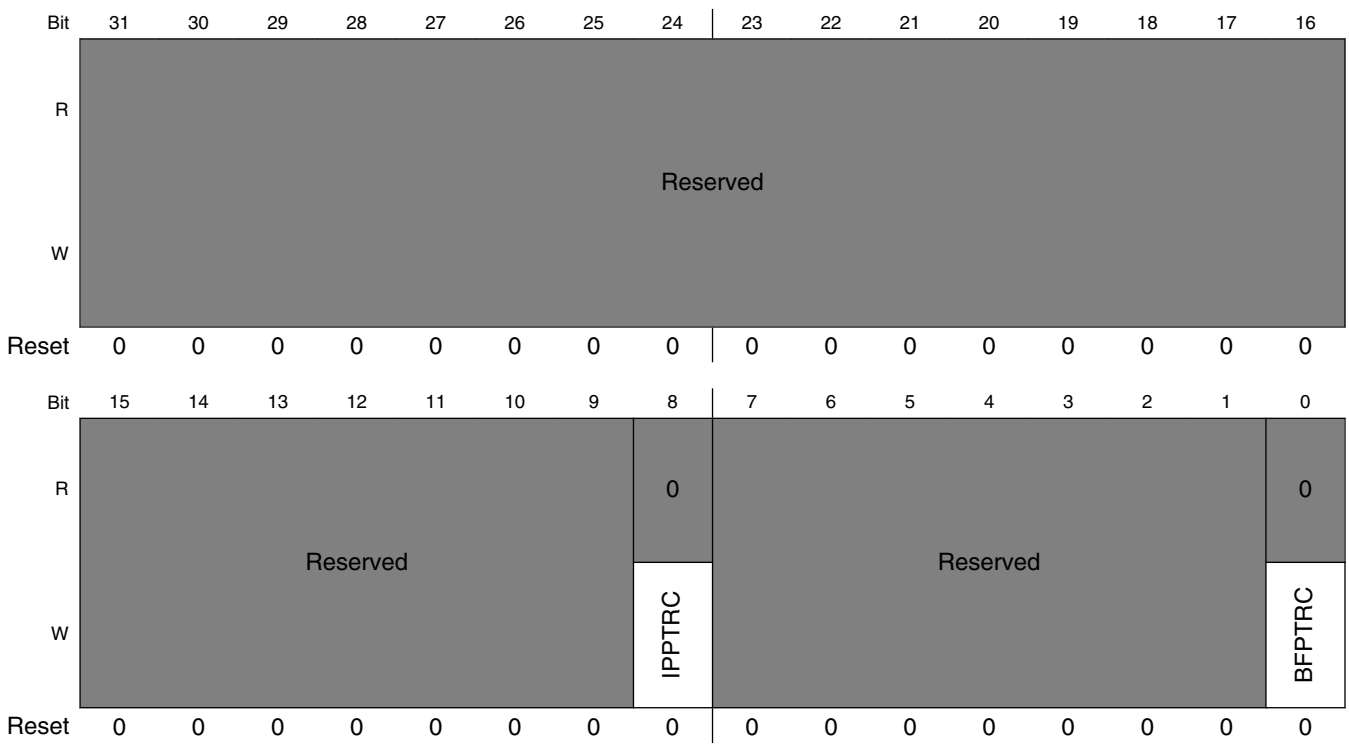
Field	Description
31–16 Reserved	This field is reserved.
15 Reserved	This field is reserved.
14–9 DATLFT	Data left: Provides information about the amount of data left to be read in the suspended sequence. Valid only when SUSPND is set to 1'b1. Value in terms of 64 bits or 8 bytes
8 Reserved	This field is reserved.
7–6 SPDBUF	Suspended Buffer: Provides the suspended buffer number. Valid only when SUSPND is set to 1'b1
5–1 Reserved	This field is reserved.
0 SUSPND	When set, it signifies that a sequence is in suspended state

35.4.2.25 Sequence Pointer Clear Register (QuadSPIx_SPTRCLR)

The sequence pointer clear register provides bits to reset the IP and Buffer sequence pointers. The sequence pointer contains the index of which instruction within the LUT entry is to be executed next. For example, if the LUT entry ends on a JMP_ON_CS value of 2, the index will be stored as 2.

The software should reset the sequence pointers whenever the sequence ID is changed by updating the SEQID field in QSPI_IPCR or QSPI_BFGENCR.

Address: 4005_A000h base + 16Ch offset = 4005_A16Ch



QuadSPIx_SPTRCLR field descriptions

Field	Description
31–9 Reserved	This field is reserved.
8 IPPTRC	IP Pointer Clear: 1: Clears the sequence pointer for IP accesses as defined in QuadSPI_IPCR This is a self-clearing field.
7–1 Reserved	This field is reserved. Reserved.
0 BFPTRC	Buffer Pointer Clear: 1: Clears the sequence pointer for AHB accesses as defined in QuadSPI_BFGENCR. This is a self-clearing field.

35.4.2.26 Serial Flash A1 Top Address (QuadSPIx_SFA1AD)

The QSPI_SFA1AD register provides the address mapping for the serial flash A1.The difference between QSPI_SFA1AD[TPADA1] and QSPI_AMBA_BASE defines the size of the memory map for serial flash A1.

Write:

- *QSPI_SR[IP_ACC] = 0*
- *QSPI_SR[AHB_ACC] = 0*

Address: 4005_A000h base + 180h offset = 4005_A180h

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 W	TPADA1																						Reserved									
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0

* Notes:

- TPADA1 field: See the module configuration for the device specific reset values.

QuadSPIx_SFA1AD field descriptions

Field	Description
31–10 TPADA1	Top address for Serial Flash A1. In effect, TPADxx is the first location of the next memory.
Reserved	This field is reserved.

35.4.2.27 Serial Flash A2 Top Address (QuadSPIx_SFA2AD)

The QSPI_SFA2AD register provides the address mapping for the serial flash A2.The difference between QSPI_SFA2AD[TPADA2] and QSPI_SFA1AD[TPADA1] defines the size of the memory map for serial flash A2.

Write:

- *QSPI_SR[IP_ACC] = 0*
- *QSPI_SR[AHB_ACC] = 0*

Address: 4005_A000h base + 184h offset = 4005_A184h

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 W	TPADA2																						Reserved									
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0

* Notes:

- TPADA2 field: See the module configuration for the device specific reset values.

QuadSPIx_SFA2AD field descriptions

Field	Description
31–10 TPADA2	Top address for Serial Flash A2. In effect, TPxxAD is the first location of the next memory.
Reserved	This field is reserved.

35.4.2.28 Serial Flash B1Top Address (QuadSPIx_SFB1AD)

The QSPI_SFB1AD register provides the address mapping for the serial flash B1.The difference between QSPI_SFB1AD[TPADB1] and QSPI_SFA2AD[TPADA2] defines the size of the memory map for serial flash B1.

Write:

- *QSPI_SR[IP_ACC] = 0*
- *QSPI_SR[AHB_ACC] = 0*

Address: 4005_A000h base + 188h offset = 4005_A188h

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 W	TPADB1																Reserved															
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0

* Notes:

- TPADB1 field: See the module configuration for the device specific reset values.

QuadSPIx_SFB1AD field descriptions

Field	Description
31–10 TPADB1	Top address for Serial Flash B1.In effect, TPxxAD is the first location of the next memory.
Reserved	This field is reserved.

35.4.2.29 Serial Flash B2Top Address (QuadSPIx_SFB2AD)

The QSPI_SFB2AD register provides the address mapping for the serial flash B2.The difference between QSPI_SFB2AD[TPADB2] and QSPI_SFB1AD[TPADB1] defines the size of the memory map for serial flash B2.

Write:

- *QSPI_SR[IP_ACC] = 0*
- *QSPI_SR[AHB_ACC] = 0*

Address: 4005_A000h base + 18Ch offset = 4005_A18Ch

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 W	TPADB2																						Reserved									
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0

* Notes:
- TPADB2 field: See the module configuration for the device specific reset values.

QuadSPIx_SFB2AD field descriptions

Field	Description
31–10 TPADB2	Top address for Serial Flash B2. In effect, TPxxAD is the first location of the next memory.
Reserved	This field is reserved.

35.4.2.30 Data Learn Pattern Register (QuadSPIx_DLPR)

The QSPI_DLPR register contains the information of the data to be used for Data Learning.

Write:
- *QSPI_SR[IP_ACC] = 0*
- *QSPI_SR[AHB_ACC] = 0*

Address: 4005_A000h base + 190h offset = 4005_A190h

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 W	DLPV																															
Reset	1	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	1	1

QuadSPIx_DLPR field descriptions

Field	Description
DLPV	Data Learning Pattern Value: This value is used for data learning in DDR and DQS mode. If programmer wants to do data learn for more than 32 bit than the same value in the register is repeated. Say if 64 bit data learning is requested by any flash and the value of DLPR is aa55_3443 then the 64 bit value will be aa55_3443_aa55_3443. If 8 bit data learning was enabled by programming in seq_operand fields of DATA_LEARN instruction to 1 , the bits [7:0] are used as data_learning pattern. For details, refer to Data Learning

35.4.2.31 RX Buffer Data Register (QuadSPIx_RBDR)

The QuadSPI_RBDR registers provide access to the individual entries in the RX Buffer. Refer to Table 35-22 for the byte ordering scheme.

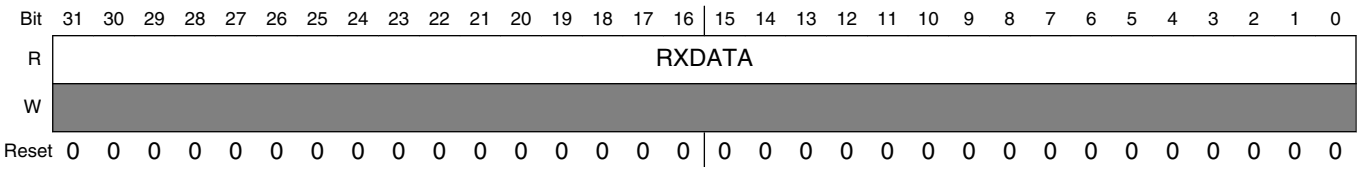
QuadSPI_RBDR0 corresponds to the actual position of the read pointer within the RX Buffer. The number of valid entries available depends from the number of RX Buffer entries implemented and from the number of valid buffer entries available in the RX Buffer.

Example 1, RX Buffer filled completely with 16 words: In this case the address range for valid read access extends from QuadSPI_RBDR0 to QuadSPI_RBDR15.

Example 2, RX Buffer filled with 5 valid words: RX Buffer fill level QuadSPI_RBSR[RDBFL] is 5. In this case an access to QuadSPI_RBDR4 provides the last valid entry.

Any access beyond the range of valid RX Buffer entries provides undefined results.

Address: 4005_A000h base + 200h offset + (4d × i), where i=0d to 15d



QuadSPIx_RBDR*n* field descriptions

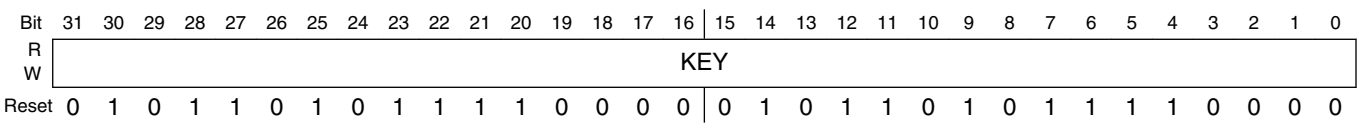
Field	Description
RXDATA	RX Data. The RXDATA field contains the data associated with the related RX Buffer entry. Data format and byte ordering is given in Byte Ordering of Serial Flash Read Data .

35.4.2.32 LUT Key Register (QuadSPIx_LUTKEY)

The LUT Key register contains the key to lock and unlock the Look-up-table. Refer to Look-up Table for details.

Write: Anytime

Address: 4005_A000h base + 300h offset = 4005_A300h



QuadSPIx_LUTKEY field descriptions

Field	Description
KEY	The key to lock or unlock the LUT. The KEY is 0x5AF05AF0. The read value is always 0x5AF05AF0

35.4.2.33 LUT Lock Configuration Register (QuadSPIx_LCKCR)

The LUT lock configuration register is used along with QSPI_LUTKEY register to lock or unlock the LUT. This register has to be written immediately after QSPI_LUTKEY register for the lock or unlock operation to be successful. Refer to Look-up Table for details. Setting both the LOCK and UNLOCK bits as "00" or "11" is not allowed.

Write: Just after writing the LUT Key Register

(QSPI_LUTKEY)

Address: 4005_A000h base + 304h offset = 4005_A304h

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														UNLOCK	LOCK
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

QuadSPIx_LCKCR field descriptions

Field	Description
31–2 Reserved	This field is reserved.
1 UNLOCK	Unlocks the LUT when the following two conditions are met: 1. This register is written just after the LUT Key Register (QuadSPI_LUTKEY) 2. The LUT key register was written with 0x5AF05AF0 key
0 LOCK	Locks the LUT when the following condition is met: 1. This register is written just after the LUT Key Register (QuadSPI_LUTKEY) 2. The LUT key register was written with 0x5AF05AF0 key

35.4.2.34 Look-up Table register (QuadSPIx_LUT*n*)

The LUT registers are a look-up-table for sequences of instructions. The programmable sequence engine executes the instructions in these sequences to generate a valid serial flash transaction. There are a total of 64 LUT registers. These 64 registers are divided into groups of 4 registers that make a valid sequence. Therefore, QSPI_LUT[0], QSPI_LUT[4], QSPI_LUT[8] ..... QSPI_LUT[60] are the starting registers of a valid sequence. Each of these sets of 4 registers can have a maximum of 8 instructions. Reset value of the register shown below is only applicable to LUT2 to LUT63. A maximum of 16 sequences can be defined at one time. Look-up Table describes the LUT registers in detail.

NOTE

The reset values for LUT0 and LUT1 are 0818_0403h and 2400_1C08h, respectively.

Write: Once the LUT is unlocked

Address: 4005_A000h base + 310h offset + (4d × i), where i=0d to 63d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INSTR1						PAD1		OPRND1							
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 W	INSTR0						PAD0		OPRND0							
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset values for LUT0 and LUT1 are 0818_0403h and 2400_1C08h respectively.

QuadSPIx_LUT*n* field descriptions

Field	Description
31–26 INSTR1	Instruction 1
25–24 PAD1	Pad information for INSTR1. 00 1 Pad 01 2 Pads 10 4 Pads 11 8 Pads
23–16 OPRND1	Operand for INSTR1.
15–10 INSTR0	Instruction 0

Table continues on the next page...

QuadSPIx_LUT*n* field descriptions (continued)

Field	Description
9–8 PAD0	Pad information for INSTR0. 00 1 Pad 01 2 Pads 10 4 Pads 11 8 Pads
OPRND0	Operand for INSTR0.

35.4.3 Serial Flash Address Assignment

The serial flash address assignment may be modified by writing into Serial Flash A1 Top Address (QuadSPI_SFA1AD) and Serial Flash A2 Top Address (QuadSPI_SFA2AD) for device A and into Serial Flash B1Top Address (QuadSPI_SFB1AD) and Serial Flash B2Top Address (QuadSPI_SFB2AD) for device B. The following table shows how different access modes are related to the address specified for the next SFM Command. Note that this address assignment is valid for both IP and AHB commands.

Table 35-13. Serial Flash Address Assignment

Parameter	Function	Access Mode
QSPI_AMBA_BASE ((31:10) - 22 bits)	QuadSPI AHB base address	
TOP_ADDR_MEMA1(TPADA1)	Top address for the external flash A1 (first device of the dual die flash A, or the first of the two independent flashes sharing the IOFA)	Any access to the address space between TOP_ADDR_MEMA1 and QSPI_AMBA_BASE will be routed to **Serial Flash A1**
TOP_ADDR_MEMA2(TPADA2)	Top address for the external flash A2 (second device of the dual die flash A, or the second of the two independent flashes sharing the IOFA).	Any access to the address space between TOP_ADDR_MEMA2 and TOP_ADDR_MEMA1 will be routed to **Serial Flash A2**
TOP_ADDR_MEMB1(TPADB1)	Top address for the external flash B1 (first device of the dual die flash B, or the first of the two independent flashes sharing the IOFB)	Any access to the address space between TOP_ADDR_MEMB1 and TOP_ADDR_MEMA2 will be routed to **Serial Flash B1**
TOP_ADDR_MEMB2(TPADB2)	Top address for the external flash B2 (second device of the dual die flash B or the second of the two independent flashes sharing the IOFB)	Any access to the address space between TOP_ADDR_MEMB2 and TOP_ADDR_MEMB1 will be routed to **Serial Flash A2**

35.4.4 AMBA Bus Register Memory Map

QSPI_AMBA_BASE defines the address to be used as start address of the serial flash device as defined by the system memory map..

Table 35-14. QuadSPI AMBA Bus Memory Map

Address	Register Name
Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A	
QSPI_AMBA_BASE to (TOP_ADDR_MEMA2 - 0x01)	Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A Refer to Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A for details and to Table 35-22 and Table 35-27 for information about the byte ordering.
Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B	
TOP_ADDR_MEMA2 to (TOP_ADDR_MEMB2 - 0x01)	Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B Refer to Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B for details and to Table 35-22 and Table 35-27 for information about the byte ordering.
Parallel Flash Mode	
QSPI_AMBA_BASE to (TOP_ADDR_MEMB2 - 0x01)	Parallel Flash Mode Refer to Parallel Flash Mode for details and to Table 35-26 and Table 35-27 for information about the byte ordering.
AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB15)	
QSPI_ARDB_BASE to… (32 * 4 Byte) QSPI_ARDB_BASE + 0x0000_01FF	AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB15) Refer to Table 35-22 and Table 35-24 for information about the byte ordering.

Note

Any read access to non-implemented addresses will provide undefined results.

In case single die flash devices, TOP_ADDR_MEMA2 and TOP_ADDR_MEMB2 should be initialized/programmed to TOP_ADDR_MEMA1 and TOP_ADDR_MEMB1 respectively- in effect, setting the size of these devices to 0. This would ensure that the complete memory map is assigned to only one flash device.

Parallel Flash Mode is valid only for commands related to data read and data write in single io mode from the serial flash. The first device of flash A has to be paired with the first device of flash B and the second device of flash A has to be paired with the second device of flash B in parallel mode. Parallel mode is selected via the QSPI_BFGENCR[PAR_EN] bit for all masters in AHB driven mode and via the QSPI_IPCR[PAR_EN] in IP driven mode. In parallel mode, the incoming address (SFAR

address in case of IP initiated transactions and the incoming AHB address in case of AHB initiated transactions) is divided by 2 and sent to the two flashes connected in parallel.

Any IP Command other than data read and write (through one pad) in Parallel Flash Mode will result in the assertion of the QSPI_FR[IUEF] flag and any AHB Command other than data read in Parallel Flash Mode will result in the assertion of the QSPI_FR[ABSEF] flag.

In the Individual Flash Modes, the 3/4 address bytes (as programmed in the instruction/operand in the sequence) available for the flash address is determined by SFADR [23:0] or SFADR [31:0]as given in the table above.

In Parallel Flash Mode, both flashes are read with the same starting address of 3/4 (as programmed in the instruction/ operand in the sequence) bytes in size. This address is derived from SFADR [24:1] or SFADR [31:1] as given in the table above. The LSB of the SFADR field is used to select the appropriate bits of both flash devices to combine the byte corresponding to the selected address.

35.4.5 AHB Bus Register Memory Map Descriptions

This chapter contains definitions of registers in the AMBA address space.

35.4.5.1 AHB Bus Access Considerations

It has to be noted that all logic in the QuadSPI module implementing the AHB Bus access is designed to read the content of an external serial flash device. Therefore the following restrictions apply to the QuadSPI module with respect to accesses to the AHB bus:

- Any write access is answered with the ERROR condition according to the AMBA AHB Specification. No write occurs.
- Any AHB Command resulting in the assertion of the QSPI_FR[ABSEF] flag is answered with the ERROR condition according to the AMBA_AHB specification. The resulting AHB Command is ignored.

- AHB Bus access types fully supported are NONSEQ and BUSY.
- AHB access type SEQ is treated in the same way like NONSEQ. Refer to the AMBA AHB Specification for further details.

35.4.5.2 Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A

Starting with address QSPI_AMBA_BASE the content of the first external serial flash devices is mapped into the address space of the device containing the QuadSPI module. Serial flash address byte address 0x0 corresponds to bus address QSPI_AMBA_BASE with increasing order. Assuming that a dual-die flash is connected on the first set of external pads, the address space is divided into two parts, one for each device of the dual die package. Refer to the following table for the address mapping. The byte ordering for 32 bit access is given in Table 35-22 and for 64 bit read access the byte ordering is given in Table 35-27.

Table 35-15. Memory Mapped Individual Flash Mode - Flash A Address Scheme

Memory Mapped Address 32 Bit Access	Memory Mapped Address 64 Bit Access	Serial Flash Byte Address	Flash Device
QSPI_AMBA_BASE + 0x00	QSPI_AMBA_BASE + 0x00	0x00_0000 to 0x00_0003	A1
QSPI_AMBA_BASE + 0x04		0x00_0004 to 0x00_0007	
…	…	…	
TOP_ADDR_MEMA1 - 0x08	TOP_ADDR_MEMA1 - 0x08	(TOP_ADDR_MEMA1- 0x08) to (TOP_ADDR_MEMA1 - 0x04 -0x01)	
TOP_ADDR_MEMA1 - 0x04		(TOP_ADDR_MEMA1 - 0x04) to (TOP_ADDR_MEMA1 - 0x01)	
TOP_ADDR_MEMA1 + 0x00	TOP_ADDR_MEMA1 + 0x00_0000	0x00_0000 to 0x00_0003	A2
TOP_ADDR_MEMA1 + 0x04		0x00_0004 to 0x00_0007	
…..	…	…	
TOP_ADDR_MEMA2 - 0x08	TOP_ADDR_MEMA2 - 0x08	(TOP_ADDR_MEMA2 - 0x08) to (TOP_ADDR_MEMA2 - 0x04 - 0x01)	
TOP_ADDR_MEMA2 - 0x04		(TOP_ADDR_MEMA2 - 0x04) to (TOP_ADDR_MEMA2 - 0x01)	

The available address range depends from the size of the external serial flash device. Any access beyond the size of the external serial flash provides undefined results.

For details concerning the read process refer to Flash Read.

35.4.5.3 Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B

Starting with address TOP_ADDR_MEMA2 the content of the first external serial flash devices is mapped into the address space of the device containing the QuadSPI module. Serial flash address byte address 0x0 corresponds to bus address TOP_ADDR_MEMA2 with increasing order. Assuming that a dual-die flash is connected on the first set of external pads, the address space is divided into two parts, one for each device of the dual die package. Refer the following table for the address mapping. The byte ordering for 32 bit access is given in Table 35-22 and for 64 bit read access the byte ordering is given in Table 35-27.

Table 35-16. Memory Mapped Individual Flash Mode - Flash B Address Scheme

Memory Mapped Address 32 Bit Access	Memory Mapped Address 64 Bit Access	Serial Flash Byte Address	Flash Device
TOP_ADDR_MEMA2 + 0x00	TOP_ADDR_MEMA2 + 0x00	0x00_0000 to 0x00_0003	B1
TOP_ADDR_MEMA2 + 0x04		0x00_0004 to 0x00_0007	
…	…	…	
TOP_ADDR_MEMB1 - 0x08	TOP_ADDR_MEMB1 - 0x08	(TOP_ADDR_MEMB1- TOP_ADDR_MEMA2 - 0x08) to (TOP_ADDR_MEMB1 - TOP_ADDR_MEMA2 - 0x04 -0x01)	
TOP_ADDR_MEMB1 - 0x04		(TOP_ADDR_MEMB1- TOP_ADDR_MEMA2 - 0x04) to (TOP_ADDR_MEMB1- TOP_ADDR_MEMA2- 0x01)	
TOP_ADDR_MEMB1 + 0x00	TOP_ADDR_MEMB1 + 0x00_0000	0x00_0000 to 0x00_0003	B2
TOP_ADDR_MEMB1 + 0x04		0x00_0004 to 0x00_0007	
…..	…	…	
TOP_ADDR_MEMB2 - 0x08	TOP_ADDR_MEMA2 - 0x08	(TOP_ADDR_MEMB2 - TOP_ADDR_MEMB1- 0x08) to (TOP_ADDR_MEMB2 - TOP_ADDR_MEMB1 - 0x04 - 0x01)	
TOP_ADDR_MEMB2 - 0x04		(TOP_ADDR_MEMB2- TOP_ADDR_MEMB1 - 0x04) to (TOP_ADDR_MEMB2 - TOP_ADDR_MEMB1 - 0x01)	

The available address range depends from the size of the external serial flash device. Any access beyond the size of the external serial flash provides undefined results.

For details concerning the read process refer to Flash Read.

35.4.5.4 Parallel Flash Mode

Any of the AHB flexible-buffers can be configured to work in parallel flash mode by programming the QSPI_BFGENCR[PAR_EN] bit to '1'. When parallel mode is set, Flash A1 is paired with Flash B1 and Flash A2 is paired with Flash B2. In parallel mode, software should ensure that the size of Flash A1(A2) is equal to the size of Flash B1(B2).

Reads from any even AHB bus address provides bits [7:4] of both serial flash devices and reads from any odd AHB bus address provides bits [3:0] of both flash devices. Refer to the following table for the address mapping. The byte ordering for 32 bit access is given in Table 35-24 and for 64 bit read access the byte ordering is given in Table 35-27.

Table 35-17. Memory Mapped Parallel Flash Mode Address Scheme

Memory Mapped Address 32 Bit Access	Memory Mapped Address 64 Bit Access	Serial Flash A Byte Address	Serial Flash B Byte Address
QSPI_AMBA_BASE + 0x0000_0000	QSPI_AMBA_BASE + 0x00 For details, please refer to Parallel mode and Dual Die Flashes.	0x00_0000 - 0x00_0001	0x00_0000 - 0x00_0001
QSPI_AMBA_BASE + 0x0000_0004		0x00_002 - 0x00_0003	0x00_0002 - 0x00_0003
QSPI_AMBA_BASE + 0x0000_0008	QSPI_AMBA_BASE + 0x08	0x00_0004 - 0x00_0005	0x00_0004 - 0x00_0005
QSPI_AMBA_BASE + 0x0000_000C		0x00_0006 - 0x00_0007	0x00_0006 - 0x00_0007
…	…	…	…
TOP_ADDR_MEMB2 - 0x08	TOP_ADDR_MEMB2 - 0x08	(TOP_ADDR_MEMB2 - QSPI_AMBA_BASE - 0x08)/2	(TOP_ADDR_MEMB2 - QSPI_AMBA_BASE - 0x08)/2
TOP_ADDR_MEMB2 - 0x04		(TOP_ADDR_MEMB2 - QSPI_AMBA_BASE - 0x04)/2 +0x01	(TOP_ADDR_MEMB2 - QSPI_AMBA_BASE- 0x04)/2 + 0x01

The available address range covers 27 address bits, corresponding to 128 MB per flash device. The usable space depends from the size of the external serial flash devices. Any access beyond the size of the external serial flash provides undefined results.

For details concerning the read process refer to Flash Read.

35.4.5.5 AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB15)

NOTE

See the System Memory map in this document for the base address of the QSPI AHB RX Data Buffer.

memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	AHB RX Data Buffer register (ARDB0)	32	R/W	0000_0000h	35.4.5.5.1/ 828
4	AHB RX Data Buffer register (ARDB1)	32	R/W	0000_0000h	35.4.5.5.1/ 828
8	AHB RX Data Buffer register (ARDB2)	32	R/W	0000_0000h	35.4.5.5.1/ 828
C	AHB RX Data Buffer register (ARDB3)	32	R/W	0000_0000h	35.4.5.5.1/ 828
10	AHB RX Data Buffer register (ARDB4)	32	R/W	0000_0000h	35.4.5.5.1/ 828
14	AHB RX Data Buffer register (ARDB5)	32	R/W	0000_0000h	35.4.5.5.1/ 828
18	AHB RX Data Buffer register (ARDB6)	32	R/W	0000_0000h	35.4.5.5.1/ 828
1C	AHB RX Data Buffer register (ARDB7)	32	R/W	0000_0000h	35.4.5.5.1/ 828
20	AHB RX Data Buffer register (ARDB8)	32	R/W	0000_0000h	35.4.5.5.1/ 828
24	AHB RX Data Buffer register (ARDB9)	32	R/W	0000_0000h	35.4.5.5.1/ 828
28	AHB RX Data Buffer register (ARDB10)	32	R/W	0000_0000h	35.4.5.5.1/ 828
2C	AHB RX Data Buffer register (ARDB11)	32	R/W	0000_0000h	35.4.5.5.1/ 828
30	AHB RX Data Buffer register (ARDB12)	32	R/W	0000_0000h	35.4.5.5.1/ 828
34	AHB RX Data Buffer register (ARDB13)	32	R/W	0000_0000h	35.4.5.5.1/ 828
38	AHB RX Data Buffer register (ARDB14)	32	R/W	0000_0000h	35.4.5.5.1/ 828
3C	AHB RX Data Buffer register (ARDB15)	32	R/W	0000_0000h	35.4.5.5.1/ 828

35.4.5.5.1 AHB RX Data Buffer register (ARDB)

The AHB RX Data Buffer register 0 to 15 can be used to read the buffer content of the RX Buffer from successive addresses. QSPI_ARDB0 corresponds to the RX Buffer register entry corresponding to the current value of the read pointer with increasing order.

The increment of the read pointer depends from the access scheme (DMA or flag-driven). Refer to "Data Transfer from the QuadSPI Module Internal Buffers" section in Flash Read section, RX Buffer, data read via register interface and AHB read, for the description of successive accesses to the RX Buffer content. Refer also to Byte Ordering of Serial Flash Read Data for the byte ordering scheme.

Valid address range accessible in the QSPI_ARDBn range depends from the number of RX Buffer entries implemented and from the number of valid buffer entries available in the RX Buffer.

- Example 1, RX Buffer filled completely with 16 words: In this case the address range for valid read access extends from QSPI_ARDB0 to QSPI_ARDB15.
- Example 2, RX Buffer filled with 5 valid words, RX Buffer fill level QSPI_RBSR[RDBFL] is 5. In this case an access to QSPI_ARDB4 provides the last valid entry.

Address: 0h base + 0h 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 W	ARXD																															
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

ARDB*n* field descriptions

Field	Description
ARXD	ARDB provided RX Buffer Data. Byte order (endianness) is identical to the RX Buffer Data Registers.

35.5 Interrupt Signals

The interrupt request lines of the QuadSPI module are mapped to the internal flags according to the following table.

Table 35-18. Assignment of Interrupt Request Lines

IRQ/DMA line	QSPI_FR Flag	Interrupt Description
ipi_int_tfff	TBFF	TX Buffer Fill

Table continues on the next page...

Table 35-18. Assignment of Interrupt Request Lines (continued)

IRQ/DMA line	QSPI_FR Flag	Interrupt Description
ipi_int_tcf	TFF	Peripheral Command Transaction Finished
ipi_int_rfdf	RBDF	RX Buffer Drain
ipi_int_overrun		Buffer Overflow/Underrun Error Logical OR from:
	RBOF	RX Buffer Overrun
	TBUF	TX Buffer Underrun
	ABOF	AHB Buffer Overflow
ipi_int_cerr		Serial Flash Command Error Logical OR from:
	IPAEF	Peripheral access while AHB busy Error
	IPIEF	Peripheral Command could not be triggered Error
	IPGEF	Peripheral access while AHB Grant Error
	IUEF	Peripheral Command Usage Error
ipi_int_ored	DLPFF, TBFF, TFF, ILLINE, RBDF, RBOF, TBUF, ABSEF, ABOF, IPAEF, IPIEF, IPGEF, IUEF, AITEF,AIBSEF	Logical OR from all the QSPI_FR flags mentioned

35.6 Functional Description

This section provides the functional information of the QuadSPI module.

35.6.1 Serial Flash Access Schemes

The following access schemes are possible, depending on the serial flash devices attached to the QuadSPI module.

Table 35-19. Access Schemes for Serial Flash Data Access

Access Scheme	One Flash Device on Port A	One Flash Device on Port B	Two identical Flash Devices connected on Port A and Port B
Individual Flash Mode: Access to Flash A	**Yes**	**N/a**	**Yes**
Individual Flash Mode: Access to Flash B	**N/a**	**Yes**	**Yes**
Parallel Flash Mode: Read from Flash A and Flash B	**N/a**	**N/a**	**Yes**

Note

If two flash devices are accessed in Parallel Flash Mode, they are accessed with identical control signals. Special alignment on per-flash basis is **not** possible. It is within the responsibility of the application to ensure that the identical signals are applicable to both flash devices.

In Parallel Flash Mode, both external serial flash devices appear logically as one single memory doubled in size with respect to one individual flash device.

If two different flash devices are attached, they can be operated only in Individual Flash Mode.

In the Parallel Flash Mode, only data read commands and write command (in x1 mode) are supported. Any other IP Command will result in an error condition signaled by the assertion of the QSPI_FR[IUEF] flag and any other AHB Command will result in the assertion of the QSPI_FR[ABSEF] flag.

In the Individual Flash Mode, all supported commands are available.

Unless explicitly noted, all the following descriptions relate to the Individual Flash Mode.

35.6.2 Modes of Operation

Refer to QuadSPI Modes of Operation for an overview over the possible operational modes of the QuadSPI block.

- Normal Mode can be used for write or read accesses to an external serial flash device.
  - Serial Flash Write: Data can be programmed into the flash via the IP interface only. Refer to Flash Programming for further details.
  - Serial Flash Read: Read the contents of the serial flash device. Two separate read channels are available via RX Buffer and AHB Buffer, see Flash Read.

- Stop Mode: The mode is used for power management. When a request is made to enter Stop Mode, the QuadSPI block acknowledges the request and completes the SFM Command in progress, then the system clocks to the QuadSPI block may be shut off
- Module Disable Mode: The mode is used for power management. The clock to the non-memory mapped logic in the QuadSPI can be stopped while in Module Disable Mode. The module enters the mode by setting QSPI_MCR[MDIS].

35.6.3 Normal Mode

This mode is used to allow communication with an external serial flash device. Compared to the standard SPI protocol, this communication method uses up to 4 bidirectional data lines operating at high data rates. The communication to the external serial flash device consists of an instruction code and optional address, mode, dummy and data transfers. The flexible programmable core engine described below is immune to a wide variety of command/protocol differences in the serial flash devices provided by various flash vendors.

35.6.3.1 Programmable Sequence Engine

The core of the QuadSPI module is a programmable sequence engine that works on "instruction-operand" pairs. The core controller executes each programmed instruction sequentially. The complete list of instructions and the corresponding operands is given in the following table.

Table 35-20. Instruction set

Instruction	Instruction encoding	Pins	Operand	Action on Serial Flash(es)
CMD	6'd1	N=2'd{0,1,2,3} 2'd0 - One pad 2'd1 - Two pads 2'd2 - Four pads 2'd3- Eight pads	8 bit command value	Provide the serial flash with operand on the number of pads specified
ADDR	6'd2		Number of address bits to be sent (for example, 8'd24 => 24 address bits required)	Provide the serial flash with address cycles according to the operand on the number of pads specified. The actual address to be provided will be derived from the incoming address in case of AHB initiated transactions and the value of SFAR in case of IPS initiated transactions , if QSPI_SFACR[CAS] is set to 0, else the actual address will take CAS into consideration.

Table continues on the next page...

Table 35-20. Instruction set (continued)

Instruction	Instruction encoding	Pins	Operand	Action on Serial Flash(es)
DUMMY	6'd3		Number of dummy clock cycles (should be <= 64 cycles)	Provide the serial flash with dummy cycles as per the operand. The PAD information defines the number of pads in input mode. (for example, one pad implies that pad 1 is not driven, rest all are driven)
MODE	6'd4		8 bit mode value	Provide the serial flash with 8 bit operand on the number of pads specified
MODE2	6'd5	N=2'd{0,1}	2 bit mode value	Provide the serial flash with 2 bit operand on the number of pads[1] specified
MODE4	6'd6	N=2'd{0,1,2}	4 bit mode value	Provide the serial flash with 4 bit operand on the number of pads[2] specified
READ	6'd7	N=2'd{0,1,2,3} 2'd0 - One pad 2'd1 - Two pads 2'd2 - Four pads 2'd3- Eight pads	Read data size in bytes (the user's application should ensure that data size is a multiple of 8 bytes)	Read data from flash on the number of pads specified. The data size may be overwritten by writing to the ADATSZ field of the QSPI_BUFxCR registers for AHB initiated transactions and IDATSZ field of IP Configuration Register (QuadSPI_IPCR) for IP initiated transactions.
WRITE	6'd8		Write data size in bytes	Write data on number of pads sepcified. The data size may be overwritten by writing to the IDATSZ field of IP Configuration Register (QuadSPI_IPCR) register
JMP_ON_CS	6'd9	NA	Instruction number	Every time the CS is deasserted, jump to the instruction pointed to by the operand. This instruction allows the programmer to specify the behavior of the controller when a new read transaction is initiated following a CS deassertion.
ADDR_DDR	6'd10	N=2'd{0,1,2,3} 2'd0 - One pad 2'd1 - Two pads 2'd2 - Four pads 2'd3- Eight pads	Number of address bits to be sent (for example, 8'd24 => 24 address bits required)	Provide the serial flash with address cycles according to the operand on the number of pads specified at each clock edge of serial flash clock. The actual address to be provided will be derived from the incoming address in case of AHB initiated transactions and the value of QSPI_SFAR in case of IPS initiated transactions , if QSPI_SFACR[CAS] is set to 0, else the actual address will take CAS into consideration.
MODE_DDR	6'd11		8 bit mode value	Provide the serial flash with 8 bit operand on the number of pads specified at each clock edge of serial flash.
MODE2_DDR	6'd12	N=2'd{0}	2 bit mode value	Provide the serial flash with 2 bit operand on the number of pads specified at each clock edge of serial flash[3]
MODE4_DDR	6'd13	N=2'd{0,1}	4 bit mode value	Provide the serial flash with 4 bit operand on the number of pads specified at each clock edge of serial flash[4].
READ_DDR	6'd14	N=2'd{0,1,2,3} 2'd0 - One pad 2'd1 - Two pads 2'd2 - Four pads	Read data size in bytes (the user's application should ensure that data size is in multiple of 8 bytes)	Read data from flash on the number of pads specified at each clock edge of serial flash. The data size may be overwritten by writing to the ADATSZ field of the QSPI_BUFxCR registers for AHB initiated transactions and IDATSZ field of IP Configuration Register (QuadSPI_IPCR) for IP initiated transactions

Table continues on the next page...

Table 35-20. Instruction set (continued)

Instruction	Instruction encoding	Pins	Operand	Action on Serial Flash(es)
WRITE_DDR	6'd15	2'd3- Eight pads	Write data size in bytes	Write data on the number of pads specified at each clock edge of serial flash. The data size may be overwritten by writing to the IDATSZ field of IP Configuration Register (QuadSPI_IPCR) register
DATA_LEARN	6'd16		Number of data in bytes to be used for data learning. Say if operand is 1 than 8 bit data_learning is enabled.	Finds the correct sampling point in case of only DDR operations. When this instruction is encountered, the QSPI_SMPR[DDRSMP] values are ignored and the controller finds the correct sampling point on its own by data learning. But this feature of sampling point is valid for only DDR modes. If DQS mode is enabled, then data learn instruction just matches the incoming data from flash and if it does not matches the QSPI_DLPV then Flag in QSPI_FR[DLPFF] is set.
CMD_DDR	6'd17	N=2'd{0,1,2,3} 2'd0 - One pad 2'd1 - Two pads 2'd2 - Four pads 2'd3- Eight pads	8 bit command value	Provide the serial flash with the operand with number of pads specified at each clock edge of the serial flash
CADDR	6'd18		Number of Column address bits to be sent(8'd8 means 8 bits of column address is to be sent)	Provide the serial flash with column address cycles according to the operand on the number of pads specified. The actual address to be provided to flash will depend on value of QSPI_SFACR[CAS]. For example, if QSPI_SFACR[CAS] is 3, then the address to flash will be [2:0] of incoming address in case of AHB and the value of QSPI_SFAR in case of IP. This will be appended with zero if QSPI_SFACR[CAS] is less than number of pads for a Flash.
CADDR_DDR	6'd19		Number of Column address bits to be sent(8'd8 means 8 bits of column address is to be sent)	Provide the serial flash with column address cycles according to the operand on the number of pads specified at each clock edge of the serial flash. The actual address to be provided to flash will depend on value of QSPI_SFACR[CAS]. For example, if CAS is 3, then the address to flash will be [2:0] of incoming address in case of AHB and the value of QSPI_SFAR in case of IP. This will be appended with zero if CAS is less than number of pads for a Flash.
STOP	8'd0	NA	NA	Stop execution; deassert CS

1. For a one pad instruction, MODE2 will take 2 serial flash clock cycles on the flash interface.
2. For a one pad instruction, MODE4 will take 4 serial flash clock cycles on the flash interface. For a 4 pad instruction, MODE4 will take 1 serial flash clock cycle on the flash interface.
3. For a one pad instruction, MODE2_DDR will take 1 serial flash clock cycle on the flash interface.
4. For a one pad instruction, MODE4_DDR will take 2 serial flash clock cycles on the flash interface. For a 4 pad instruction MODE4_DDR will take half a cycle on the serial flash interface.

A sequence of such instruction-operand pairs may be pre-populated in the LUT according to the device connected on board. Each instruction-operand pair is of 16 bits (2 bytes) each. Every sequence pre-programmed in the LUT is referred to by its index.

The programmable sequence engine allows the user to configure the QuadSPI module according to the serial flash connected on board. The flexible structure is easily adaptable to new command/protocol changes from different vendors.

35.6.3.2 Flexible AHB buffers

In order to reduce the latency of the reads for AHB masters, the data read from the serial flash is buffered in flexible AHB buffers. There are four such flexible buffers. The size of each of these buffers is configurable with the minimum size being 0 Bytes and maximum size being the size of the complete buffer instantiated.The size of buffer 0 is defined as being from 0 to QSPI_BUF0IND. The Size of buffer 1 is from QSPI_BUF0IND to QSPI_BUF1IND, buffer2 is from QSPI_BUF1IND to QSPI_BUF2IND and buffer 3 is from QSPI_BUF2IND to the size of the complete buffer, which is given in the chip-specific QuadSPI information.

Each flexible AHB buffer is associated with the following

1. An AHB master. Optionally, buffer3 may be configured as an "all master" buffer by setting the QSPI_BUF3CR[ALLMST] bit. When buffer3 is configured in such a way, any access from a master not associated with any other buffer is routed to buffer3.
2. A datasize field representing the amount of data to be fetched from the flash on every "missed" access.

The master port number of every incoming request is checked and the data is returned/ fetched into the corresponding associated buffer. Every "missed" access to the buffer causes the controller to clear the buffer and fetch QSPI_BUFxCR[ADATSZ] amount of data from the serial flash.As such, there is no benefit in configuring a buffer size of greater than ADATSZ, as the locations greater than ADATSZ will never be used. For any AHB access, the sequence pointed to by the QSPI_BFGENCR[SEQID] field is used for the flash transaction initiated. The data is returned to the master as soon as the requested amount is read from the serial flash. The controller however, continues to prefetch the rest of the data in anticipation of a next consecutive request. Figure 35-4 shows the flexible AHB buffers.

The QSPI_BFGENCR[SEQID] field points to an index of the LUT. Refer to Look-up Table for details.

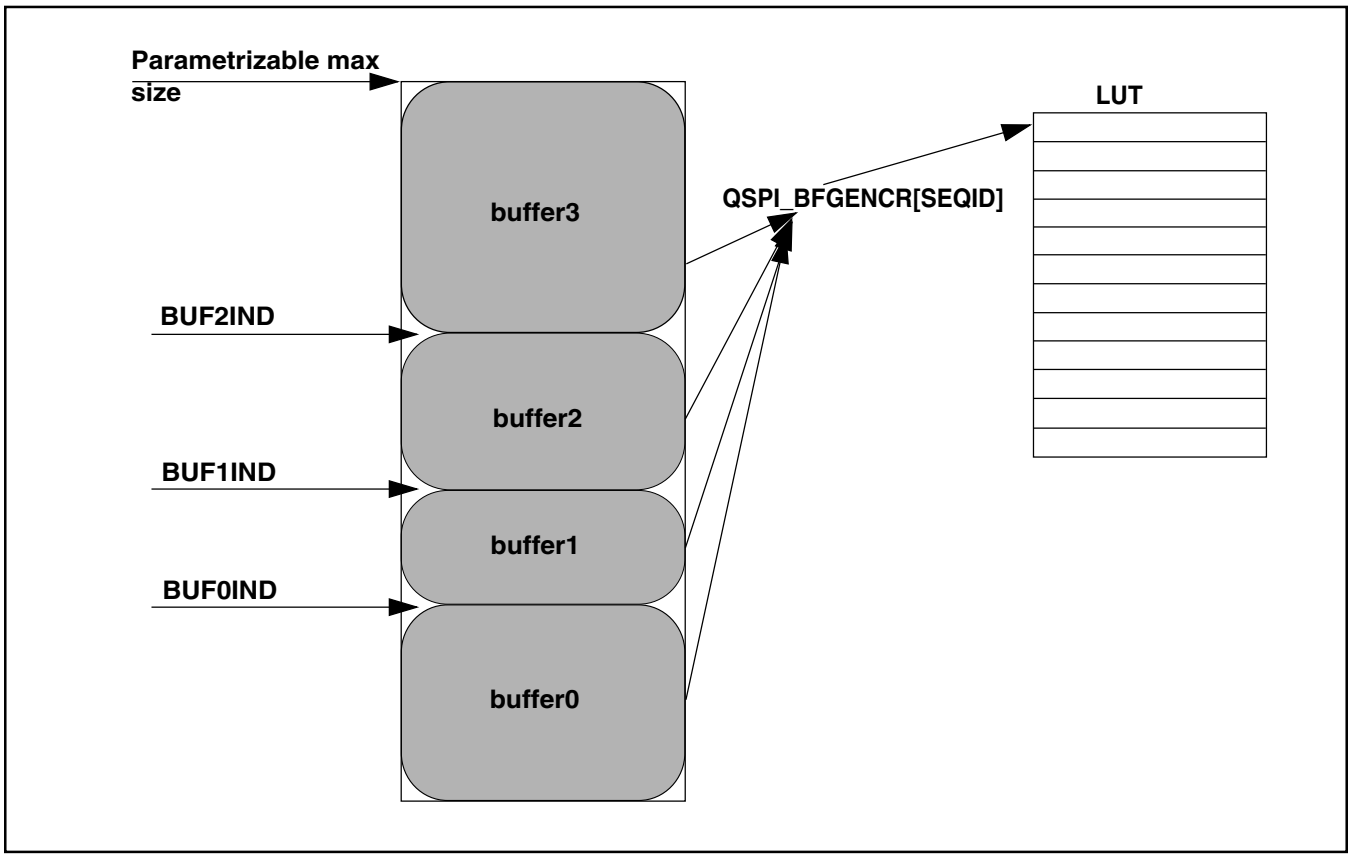


Figure 35-4. Flexible AHB Buffers

Buffer0 may optionally be configured to be associated with a high priority master by setting the QSPU_BUF0CR[HP_EN] bit. An access by a high priority master suspends any ongoing prefetch to any of the other buffers. The ongoing prefetch is suspended and the high priority master is serviced first. Once the high priority masters access completes, the suspended transaction is resumed (before any other AHB access is entertained). The status of the suspended buffer can be read from Sequence Suspend Status Register (QuadSPI_SPNDST).

35.6.3.3 Suspend-Abort Mechanism

Any low priority AHB access can be suspended by a high priority AHB master request. The ongoing transaction is suspended at 64 bit boundary. The suspended transaction is restarted after the high priority master is served and the high priority transaction including data prefetch is completed. While a transaction is in suspended state, it may be aborted if a transaction by the same suspended master is made to a location which is different from the location of the suspended transaction.

Any ongoing transaction is aborted if a request from the same master arrives for a location other than the location at which the transaction is going on. The abort can happen at any point of time.

35.6.3.4 HBURST Support

QuadSPI controller supports HBURST and HSIZE on the AHB interface. HBURST indicates if the transfer forms part of a burst. Four, eight and sixteen beat bursts are supported and the burst may be either incrementing or wrapping. HSIZE indicates the size of the transfer. 8, 16, 32 and 64 bit data size are supported. In case of WRAP accesses, QuadSPI generates aligned accesses to Serial Flash if there is no buffer hit for any incoming non-sequential AHB access. In case there is a buffer hit, the incoming address in the haddr line is latched as it is. If the total burst size is more than the data prefetch size an error response is generated and QSPI_FR[AIBSEF] is set. The data perfetch size can be defined by QSPI_BUFxCR[ADATSZ] or data size mentioned in the sequence pointed to by the SEQID field when ADATSZ is programmed as zero. A few examples are shown in the figure below:

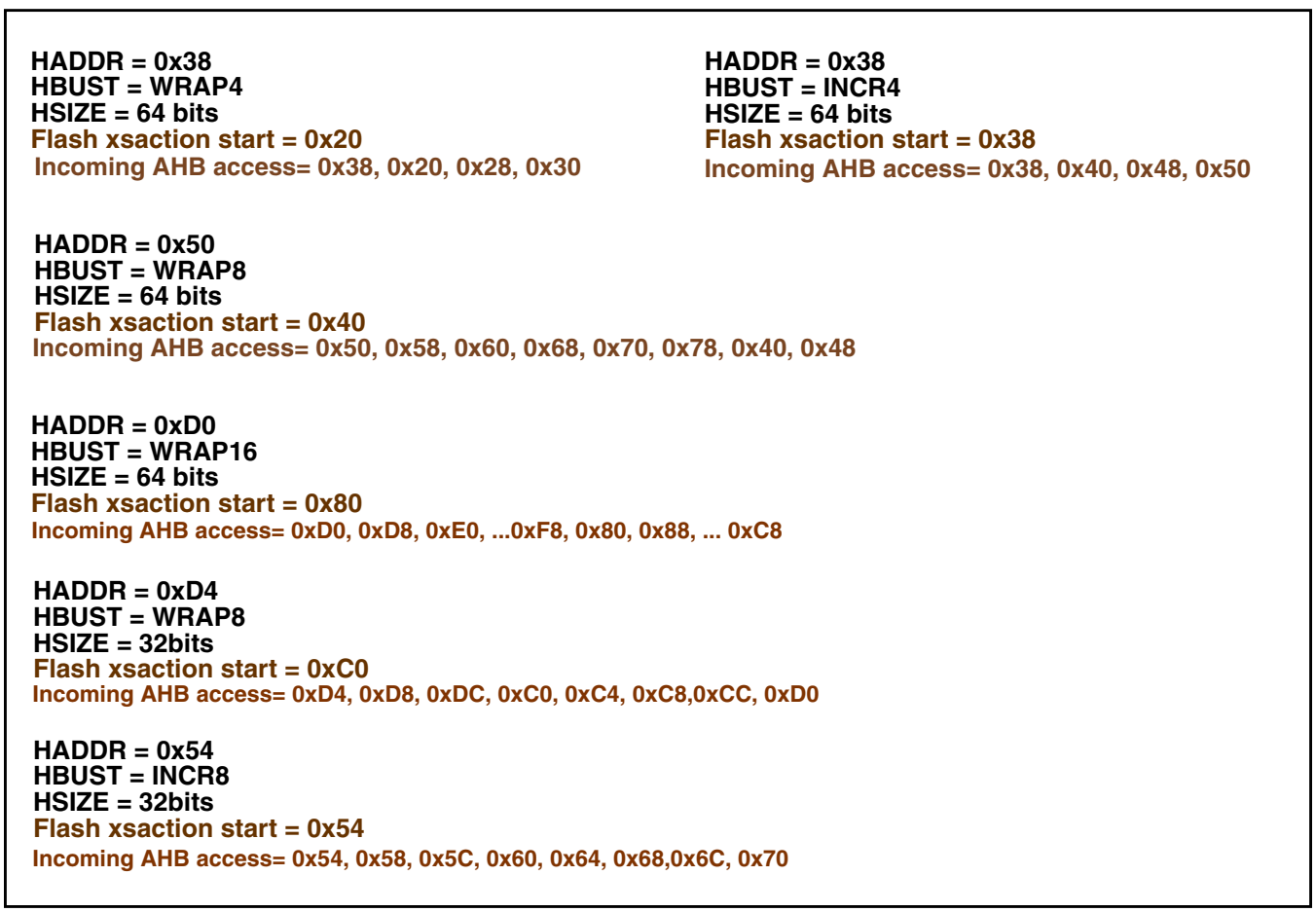


Figure 35-5. QuadSPI HBURST support

NOTE

The software must take care that the prefetch size should never be set less than the minimum data needed by any external interface to start processing.

35.6.3.5 Look-up Table

The Look-up-table or LUT consists of a number of pre-programmed sequences. Each sequence is basically a sequence of instruction-operand pairs which when executed sequentially generates a valid serial flash transaction. Each sequence can have a maximum of 8 instruction-operand pairs. The LUT can hold a maximum of 16 sequences. The figure below shows the basic structure of the sequence in the LUT.

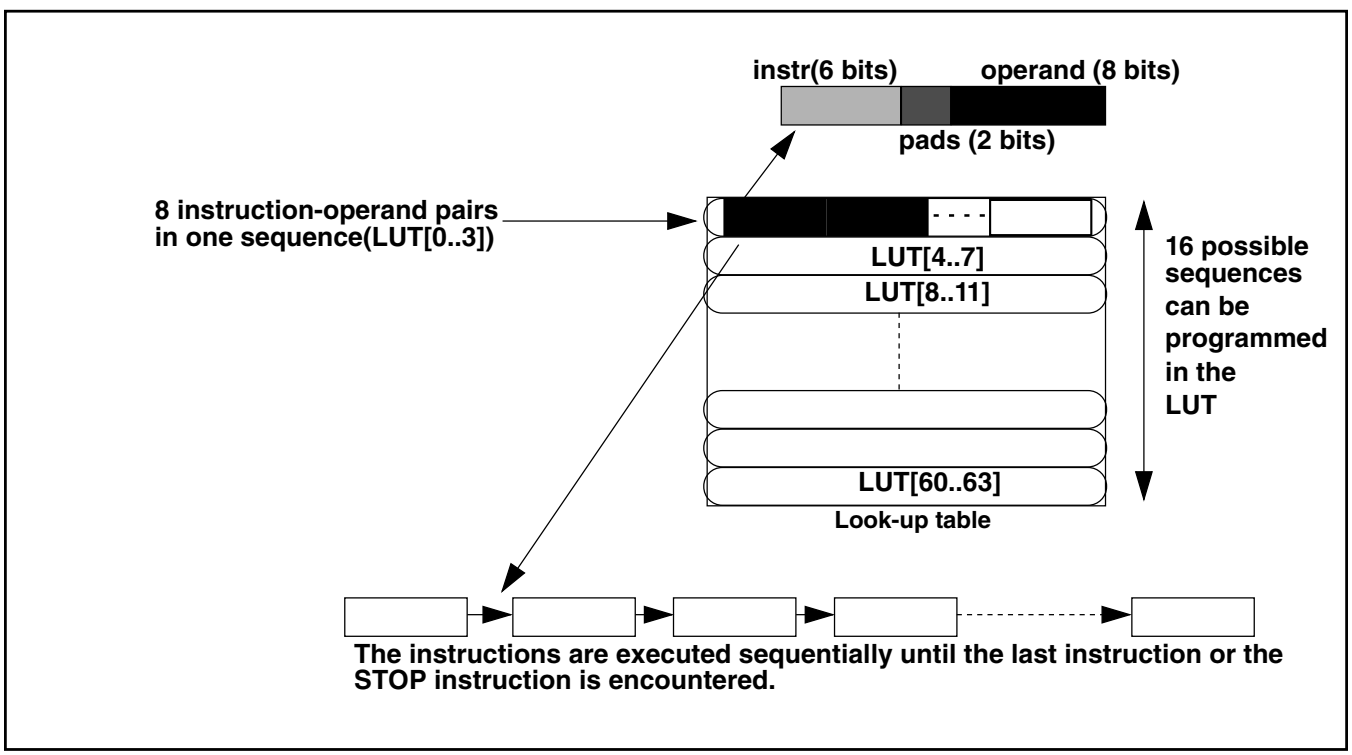


Figure 35-6. LUT and sequence structure

At reset, the index 0 of the look-up-table (LUT[0..3]) is programmed with a basic read sequence as given in Table 35-21. After reset the complete LUT may be reprogrammed according to the device connected on board.In order to protect its contents during a code runover the LUT may be locked, after which a write to the LUT will not be successful until it has been unlocked again.The key for locking or unlocking the LUT is **0x5AF05AF0**. The process for locking and un-locking the LUT is as follows:

Locking the LUT

1. Write the key **(0x5AF05AF0)** in to the LUT Key Register (QuadSPI_LUTKEY).
2. Write 0b01 to the LUT Lock Configuration Register (QuadSPI_LCKCR). Note that this IPS transaction should immediately follow the above IPS transaction (no other IPS transaction can be issued in between). A successful write into this register locks the LUT.

Unlocking the LUT

1. Write the key **(0x5AF05AF0)** into the LUT Key Register (QuadSPI_LUTKEY)
2. Write 0b10 to the LUT Lock Configuration Register (QuadSPI_LCKCR). Note that this IPS transaction should immediately follow the above IPS transaction (no other IPS transaction can be issued in between). A successful write into this register unlocks the LUT.

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

Some example sequences are defined in Example Sequences. The reset sequence at LUT index 0 is given in the following table.

Table 35-21. Reset sequence

Instruction	Pad	Operand	Comment
CMD	0x00	0x03	Read Data byte command on one pad
ADDR	0x00	0x18	24 Addr bits to be sent on one pad
READ	0x00	0x08	Read 64 bits
JMP_ON_CS	0x00	0x00	Jump to instruction 0 (CMD)

35.6.3.6 Issuing SFM Commands

Each access to the external device follows the same sequence:

1. The user must pre-populate the LUT with the serial flash command sequences that are required for the flash device being used.
2. The QuadSPI module starts executing the instructions in the sequence one by one. The transaction starts and the status bit QSPI_SR[BUSY] is set.
3. Communication with the external serial flash device is started and the transaction is executed.

4. When the transaction is finished (all transmit- and receive operations with the external serial flash device are finished) the status bit QSPI_SR[BUSY] is reset. In case of an IP Command the QSPI_FR[TFF] flag is asserted.

Further details are given in below in Flash Programming and Flash Read.

You can trigger the processing of SFM commands in the QuadSPI module in one of the following ways:

- **Using IP commands**

  For IP Commands the required components need to be written into the following registers:

  - Write the serial flash address to be used by the instruction into QSPI_SFAR, refer to Serial Flash Address Register (QSPI_SFAR). For IP Commands not related to specific addresses, the base address of the related flash need to be programmed.For example, for an instruction which does not require an address (i.e. write enable instruction) the SFAR should be programmed with the base address of the memory the command is to be sent to.
  - Write the sequence ID and data size details in the IP Configuration Register (QSPI_IPCR).
  - Note that the write into the QSPI_IPCR[SEQID] field must be the last step of the sequence. It is possible to combine all fields of the QSPI_IPCR into one single write. Refer to IP Configuration Register (QSPI_IPCR) for details.

  Note that there are some conditions where no IP Command is executed after writing the QSPI_IPCR[SEQID] field and the write operation itself is ignored. They are described in Command Arbitration.

- **Using AHB commands**

  Any AHB memory mapped access is routed to one of the buffers depending on the master port number of the request. If the access is a "miss", a new serial flash transaction is started. The transaction is based on the sequence pointed to by the BFGENCR[SEQID] field as described in Flexible AHB buffers.

  An AHB access is termed memory mapped when the access is to the memory mapped serial flashes, as described in Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A and Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B.

  Again the possible error conditions are described in Command Arbitration .

35.6.3.7 Flash Programming

In all cases the memory sector to be written needs to be erased first. The programming sequence itself is then initiated in the following way:

1. Check that the TX Buffer is empty. If the QSPI_SR[TXEDA] bit is set then the TX Buffer must be cleared by writing 1 into the QSPI_MCR[CLR_TXF] bit.
2. Program the address related to the command in the QSPI_SFAR register. Program the QSPI_SFACR[CAS] if required to desired value else to 0. Program the QSPI_SFACR[WA] to 1 if the serial flash is a word addressable flash else to 0 in case serial flash is byte addressable.
3. Provide initial data for the program command into the circular buffer via register TX Buffer Data Register (QSPI_TBDR) . At least four word of data must be written into the TX Buffer up to a maximum of 16.
4. Program the QSPI_IPCR register to trigger the command. The QSPI_IPCR[SEQID] should point to an index of the LUT which has the flash program sequence pre-programmed.The IDATSZ field should be set to denote the size of the write.
5. Depending on the amount of data required, step 3 must be repeated until all the required data have been written into the QSPI_TBDR register.The QSPI_SR[TXFULL] can be used to check if the buffer is ready to receive more data. At any time, the QSPI_TBSR[TRCTR] field can be read to check how many words have been written actually into the TX Buffer.

Upon writing the QSPI_IPCR[SEQID] field (refer to step 4) the QuadSPI module will start to execute the programmed sequence. It is the responsibility of the software to ensure that a correct sequence is programmed into the LUT in accordance with the flash memory connected to the module. The data is fetched from the TX Buffer. It consists of **16** entries of 32-bits and is organized as a circular FIFO, whose read pointer is incremented by four after each fetch. When all data are transmitted, the QuadSPI module will return from 'busy' to 'idle'. However, this is not true for the external device since the internal programming is still ongoing. It is up to the user to monitor the relevant status information available from the serial flash device and to ensure that the programming is finished properly.

35.6.3.8 Flash Read

Host access to the data stored in the external serial flash device is done in two steps. First, the data must be read into the internal buffers and in the second step these internal buffers can be read by the host.

1. **Reading Serial Flash Data into the QuadSPI Module Internal Buffers**

   A read access to the external serial flash device can be triggered in two different ways:

   - **IP Command Read**: For **reading flash data into the RX Buffer** the user must provide the correct sequence ID in the QuadSPI_IPCR[SEQID] register. The sequence ID points to a sequence in the LUT. It is the responsibility of the software to ensure that a correct read sequence is programmed in the LUT in accordance with the serial flash device connected on board. The user should program the Serial Flash Address Register (QSPI_SFAR) , QSPI_SFACR[CAS] and the IP Configuration Register (QSPI_IPCR) registers. All available read commands supported by the external serial flash are possible.

     Optionally it is possible to clear the RX Buffer pointer prior to triggering the IP Command by writing a 1 into the QuadSPI_MCR[CLR_RXF] field.

     From these inputs, the complete transaction is built when the QSPI_IPCR[SEQID] field is written. The transaction related to the read access starts and the requested number of bytes is fetched from the external serial flash device into the RX Buffer. Since the read access is triggered by an IP command, the IP_ACC status bit and the BUSY bit are both set (both are located in the Status Register (QSPI_SR)). A count of the number of entries currently in the Rx Buffer can be obtained from QSPI_RBSR[RDBFL].

     The communication with the external serial flash is stopped when the specified number of bytes has been read (successful completion of the transaction).

   - **AHB Command Read**: For **reading flash data into the AHB Buffer** the user must set up a read access by a master to the address range in the system memory map which the external serial flash devices are mapped to. The user should program the QSPI_SFACR[CAS], if required, to desired value else to 0.The user should also program the buffer registers corresponding to the AHB master initiating the request, this is depends on the configuration of the QSPI_RBCT[RXBRD]. The user should provide the correct sequence ID into the buffer generic configuration register (QSPI_BFGENCR). It is the responsibility of the software to ensure that a correct read sequence in programmed in the LUT in accordance with the serial flash device connected on board. Flash device selection and access mode are determined by the address

accessed in the AHB address space associated to the QuadSPI module (refer to Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A, Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B and Parallel Flash Mode.)

On each AHB read access to the memory mapped area the valid data in the AHB Buffer is checked against the address requested in the actual read. When the AHB read request can't be served from the content of the AHB Buffer, the buffer is flushed and the sequence pointed to by the sequence ID is executed by the controller. The requested number of buffer entries defined in the QSPI_BUFxCR[ADATSZ] field is then fetched from the external serial flash device into the internal AHB Buffer. Since the read access is triggered via the AHB bus, the QSPI_SR[AHB_ACC] status bit is set driving in turn the QSPI_SR[BUSY] bit until the transaction is finished. The communication with the external serial flash is stopped when the specified number of entries has been filled.

2. **Data Transfer from the QuadSPI Module Internal Buffers**

   The data read out from the external serial flash device by the QuadSPI module is stored in the internal buffers.The means of accessing the data from the buffer differs depending on which buffer the data has been loaded to. Refer to Block Diagram for details about the two available buffers, the RX Buffer and the AHB Buffer, in the QuadSPI module:

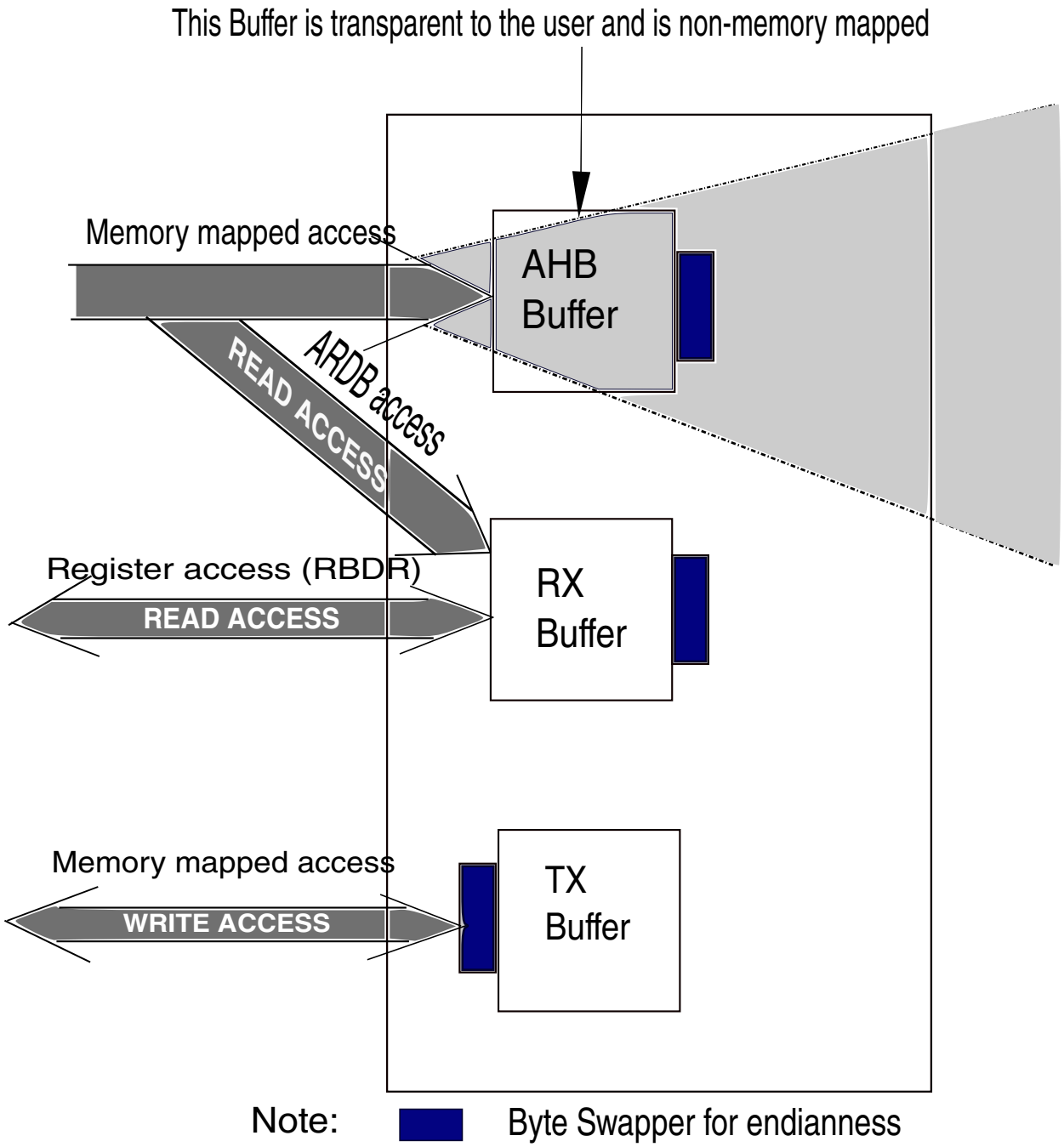


Figure 35-7. QuadSPI memory map

- The RX Buffer is implemented as FIFO of depth 16 entries of 4 bytes. Its content is accessible in two different address areas both referring to the identical data and the same physical memory.

  In the IPS address space in the area associated to QSPI_RBDR0 to QSPI_RBDR15

  In the AHB address space in the area associated to QSPI_ARDB0 to QSPI_ARDB15. Two successive entries are accessed with one single 64 bit AHB read operation.

RX Buffer operation can be summarized as follows: The QSPI_RBCT[WMRK] field determines at which fill level the RXWE bit is asserted and how many entries are removed from the RX Buffer on each Buffer POP operation. So the QSPI_SR[RXWE] bit indicates that the configured number of data entries is available in the RX Buffer and the QSPI_RBSR[RDBFL] field indicates how many valid entries are available in total. Note that the first entry (QSPI_RBDR0 or QSPI_ARDB0) always corresponds to the first valid entry in the RX Buffer. The software needs to manage the number of valid data bytes itself.

Further details can be found in RX Buffer Data Register (QuadSPI_RBDR) and in AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB15).

- **Flag-based Data Read of the RX Buffer** is done by polling the QSPI_SR[RXWE] bit. When it is asserted the valid entries can be read either via the IPS address space (QSPI_RBDRn) or the AHB address space (QSPI_ARDBn). A Buffer POP operation must be triggered by the application by writing a 1 into the QSPI_FR[RBDF] bit - this automatically updates the FIFO to point to the next entry as defined by RBCT[WMRK]. For example, if WMRK is set to 3, then the buffer will discard 16 bytes of data.
- **DMA controlled Data Read of the RX Buffer** is done by using the DMA module.The application must ensure that the DMA controller of the related device is programmed appropriately like it is described in DMA Usage.

  DMA controlled read out is triggered fully automatically by the assertion of the QSPI_SR[RXWE] bit. The related Buffer POP operation is also handled completely inside the QuadSPI module. Like in the case above, accessing the RX Buffer content either on QSPI_RBDRn or QSPI_ARDBn related addresses is equivalent.

- **AHB Buffer data read via memory mapped access:** This kind of access is done by reading one of the addresses assigned to the external serial flash device(s) within the range given in Table 35-14 table *under the condition that the data requested are already present in the AHB Buffer or it is currently being read from the serial flash device by the instruction in progress.* If this is not the case a memory mapped AHB command read is triggered as described above. If the requested data is already available in the AHB Buffer they are provided directly to the host.

  When AHB access are made to the flash memory mapped address, the data will be fetched and returned to the AHB interface. Till the data is being fetched the AHB interface would be stalled. As soon as the data from the requested address has been read by the QuadSPI module the AHB read access is served. So it is possible to run sequential reads from the AHB buffer at arbitrary speed without

the need to monitor any information about the availability of the data. Nevertheless this access scheme stalls the AHB bus for the time required to read the data from the serial flash device. A better way is (when it is known the access is sequential) to have a prefetch enabled (by programming the ADATSZ field) such that before the next sequential AHB access come, the data is already fetched into the buffer.

As long as the host restricts its accesses to the data already in the buffer and the data currently fetched from the serial flash, it is possible to run the host read from the AHB Buffer in parallel to the serial flash read into the AHB Buffer.

35.6.3.9 Byte Ordering of Serial Flash Read Data

In this paragraph the byte ordering of the serial flash data is given. The basic scheme is that the **first** byte read out of the serial flash device - which is addressed by the QSPI_SFAR[SFADR] field - corresponds to bit position QSPI_RBDR0[31:24] register for IP Command read. In contrast to that for AHB Command read the bytes are always positioned according to the byte ordering of the AHB bus.

- **Byte Ordering in Individual Flash Mode**

  The following table gives the byte ordering scheme of how the byte oriented data space of the serial flash device is mapped into one single 32 bit entry of the RX Buffer or the AHB Buffer. The table is valid within the following context:

  - Flash A or Flash B in Individual Flash Mode
  - All AHB data read commands with access size of 32 bit

Table 35-22. Byte Ordering in Individual Flash Mode

Serial Flash Byte Numbering	3	2	1	0
Buffer Entry Bit Position [31:0] (32 Bit data width)	[31:24]	[23:16]	[15:8]	[7:0]

Note

For IP Commands the read size can be given in number of bytes. If this number is not a multiple of 4, then the last buffer entry is not completely filled with the missing higher numbered bytes at undefined values.

For AHB Commands, reads, starting from an address not aligned to 32 bit boundaries, the requested bytes are given at the appropriate positions according to the AMBA AHB specification.

- **Byte Ordering in Parallel Flash Mode**

  In Parallel Flash Mode each byte is combined out of 2 half bytes which are read or written in parallel from the two serial flash devices. The following tables shows how the flash content is separated into the half bytes and how the half bytes are assembled to the content of the QSPI_RBDR0 register.

Table 35-23. Serial Flash Device Half Byte Ordering

Serial Flash Device Byte #	Flash A Bit Position		Flash B Bit Position	
	[7:4]	[3:0]	[7:4]	[3:0]
0	fah0	fal0	fbh0	fbl0
1	fah1	fal1	fbh1	fbl1
2	fah2	fal2	fbh2	fbl2
3	fah3	fal3	fbh3	fbl3
4	fah4	fal4	fbh4	fbl4
5	fah5	fal5	fbh5	fbl5
6	fah6	fal6	fbh6	fbl6
7	fah7	fal7	fbh7	fbl7
8	fah8	fal8	fbh8	fbl8

The table entry naming reflects the half byte positioning in the serial flash devices:

- **<fa>**h0 means **Flash A**, <fb>h0 means Flash B.
- fa**<h>**0 means half byte in **high position**, fa<l>o means half byte in low position.
- fah**<0>** means **physical byte address 0** in the serial flash device, fal<1> means physical byte address 1 in the serial flash device.

Table 35-24. Byte Ordering in Parallel Flash Mode - RX Buffer

QSPI_SFAR[SFADR] set to 0x000_0000

Table continues on the next page...

Table 35-24. Byte Ordering in Parallel Flash Mode - RX Buffer (continued)

QSPI_RBDR0 **QSPI_ARDB0**	fal1	fbl1	fah1	fbh1	fal0	fbl0	fah0	fbh0
QSPI_RBDR1 **QSPI_ARDB1**	fal3	fbl3	fah3	fbh3	fal2	fbl2	fah2	fbh2
QSPI_SFAR[SFADR] set to 0x000_0001								
QSPI_RBDR0 **QSPI_ARDB0**	fal2	fbl2	fah2	fbh2	fal1	fbl1	fah1	fbh1
QSPI_RBDR1 **QSPI_ARDB1**	fal4	fbl4	fah4	fbh4	fal3	fbl3	fah3	fbh3

Note

For IP Commands the read size can be given in number of bytes. If this number is not a multiple of 4 the last buffer entry is not completely filled with the missing higher numbered bytes at undefined values.

Table 35-25. Byte Ordering in Parallel Flash Mode - TX Buffer*

QSPI_SFAR[SFADR] set to 0x000_0000								
QSPI_TBDR0	fal1	fbl1	fah1	fbh1	fal0	fbl0	fah0	fbh0
QSPI_TBDR1	fal3	fbl3	fah3	fbh3	fal2	fbl2	fah2	fbh2

* Applicable only for single I/O mode.

Table 35-26. Byte Ordering in Parallel Flash Mode - AHB Buffer

AHB Address **(32 Bit Access)**	fal1	fbl1	fah1	fbh1	fal0	fbl0	fah0	fbh0
AHB Address 0x800_0004 **(32 Bit Access)**	fal3	fbl3	fah3	fbh3	fal2	fbl2	fah2	fbh2

Note

For AHB Command read starting from an address not aligned to 32 bit boundaries or AHB access size smaller than 32 bit the requested bytes are given at the appropriate positions according to the AMBA AHB specification.

- **Buffer Entry Ordering for 64 Bit Read Access**

  For read access via the AHB interface 64 bit access is possible. Each 64 bit access reads 2 32 bit entries simultaneously. The ordering of these 32 bit entries within the 64 bit word is given in the following table.

Table 35-27. 64 Bit Read Access Buffer Entry Ordering

AHB Read Data Bit Position [63:0]	[63:32]	[31:0]
Buffer Entry #	Odd (1, 3, 5, ...)	Even (0, 2, 4, ...)

35.6.3.10 Normal Mode Interrupt and DMA Requests

The QuadSPI module has different flags that can only generate interrupt requests and one flag that can generate interrupt as well as DMA requests. The following table lists the eight conditions. Note that the flags mentioned in the table are related to the Flag Register (QSPI_FR).

Table 35-28. Interrupt and DMA Request Conditions

Condition	Flag(QSPI_FR)	DMA
Data Learn patttern Failure	DLPFF	-
TX Buffer Fill	TBFF	-
TX Buffer Underrun	TBUF	-
Illegal Instruction Error	ILLINE	-
RX Buffer Drain	RBDF	X
RX Buffer Overflow	RBOF	-
AHB Buffer Overflow	ABOF	-
AHB Sequence Error	ABSEF	-
AHB Illegal Transaction Error	AITEF	-
AHB Illegal Burst Size Error	AIBSEF	-
IP Command Usage Error	IUEF	-

Table continues on the next page...

Table 35-28. Interrupt and DMA Request Conditions (continued)

Condition	Flag(QSPI_FR)	DMA
IP Command Trigger during AHB Access Error	IPAEF	-
IP Command Trigger could not be executed Error	IPIEF	-
IP Access during AHB Grant Error	IPGEF	-
IP Command related Transaction Finished	TFF	-

Each condition has a flag bit in the Flag Register (QSPI_FR) and a Request Enable bit in the DMA Request Select and Enable Register (QSPI_RSER) . The RX Buffer Drain Flag (RBDF) has separate enable bits for generating IRQ and DMA requests. Note that not all flags have an individual IRQ line. Check the devices Interrupt Vector Table for more details.

- Transmit Buffer Fill Interrupt Request:

  The Transmit Buffer Fill IRQ indicates that the TX Buffer can accept new data. It is asserted if the QSPI_FR[TBFF] flag is asserted and if the corresponding enable bit (QSIP_RSER[TBFIE]) is set. Refer to TX Buffer Operation, for details about the assertion of the QSPI_FR[TBFF] flag.

  Aside from IRQ it is possible to handle TX Buffer fill by DMA. If the QSPI_RSER[TBFDE] bit is set, a DMA request will be triggered when the number of available space in the TX Buffer is more than the QSPI_TBCT[WMRK] valid entries and QSPI_SR[TXWA] is set. The application must set the environment appropriately (eg. the DMA controller) for the DMA transfer.

- Receive Buffer Drain Interrupt or DMA Request:

  The Receive Buffer Drain IRQ derived from the QSPI_FR[RBDF] flag indicates that the RX Buffer of the QuadSPI module has data available from the serial flash device to be read by the host. It remains set as long as the QSPI_RBSR[RXWE] bit is set. The QSPI_RSER[RBDIE] bit enables the related IRQ.

  Aside from the IRQ it is possible to handle RX Buffer drain by DMA. If the QSPI_RSER[RBDDE] bit is set, a DMA request will be triggered when the RX Buffer contains more than QSPI_RBCT[WMRK] valid entries.The application must set the environment appropriately (for example, the DMA controller) for the DMA transfers.

- Buffer Overflow/Underrun Interrupt Request:

The Buffer Overflow/Underrun IRQ is a combination of the following flags (all located in the QSPI_FR register with the related enable bits in the QSPI_RSER register):

- TBUF - TX Buffer Underrun, enabled by TBUIE
- RBOF - RX Buffer Overflow, enabled by RBOIE
- ABOF - AHB Buffer Overflow, enabled by ABOIE

  The Transmit Buffer Underrun indicates that an underrun condition in the TX Buffer has occurred. It is generated when a write instruction is triggered whilst the Tx Buffer is empty and the QSPI_RSER[TBUIE] bit is set.

  The Receive Buffer Overflow indicates that an overflow condition in the RX Buffer has occurred. It is generated when the RX Buffer is full, an additional read transfer attempts to write into the RX Buffer and the QSPI_RSER[RBOIE] bit is set.

  The AHB Buffer Overflow indicates that an overflow condition in the AHB Buffer has occurred. It is generated when the AHB Buffer is full, an additional read transfer attempts to write into the AHB Buffer and the QSPI_RSER[ABOIE] bit is set.

  The data from the transfers that generated the individual overflow conditions is ignored.

- Serial Flash Command Error Interrupt Request

  If the IPAEF, IPIEF, IPGEF or IUEF flags in the QSPI_FR are set, and the related interrupt enable bits in the QSPI_RSER are also set, then an interrupt is requested.

- Transaction Finished Interrupt Request

  The IP Command Transaction Finished IRQ indicates the completion of the current IP Command.It is triggered by the QSPI_FR[TFF] flag and is masked by the QSPI_RSER[TFIE] bit.

35.6.3.11 TX Buffer Operation

The TX Buffer provides the data used for page programming. For proper operation it is required to provide at least four entry in the TX Buffer prior to starting the execution of the page programming command. The application must ensure that the required number of data bytes is written into the TX Buffer fast enough as long as the command is executed without a TX Buffer overflow or underrun.

The QuadSPI module sets the QSPI_FR[TBFF] flag so long as the TX Buffer is not full and can accept more data.

When the QuadSPI module tries to pull data out of an empty TX Buffer the TX Buffer underrun is signaled by the QSPI_FR[TBUF] flag.The TX buffer underrun flag is also asserted when TX buffer contains less than 128 bits of data and QuadSPI module tries to pull out data from it. The current IP Command leading to the underrun condition is continued until the specified number of bytes has been sent to the serial flash device, in the underrun condition when QuadSPI module tries to pull out data of empty TX buffer, the data transferred is all F's i.e. once the underrun flag is set under this condition, it will return F's until the required number of bytes are not sent. This has been done to ensure that the software need not to erase whole sector after underrun, just reprogramming from failure point will serve the purpose. When this Sequence Command is finished, the QSPI_FR[TBFF] flag is asserted indicating that the Tx Buffer is ready to be written again.

The TX Buffer overflow isn't signaled explicitly, but the TX Buffer fill level can be monitored by the QSPI_TBSR[TRBFL] field.

Refer to TX Buffer Status Register (QuadSPI_TBSR) and Flag Register (QuadSPI_FR) for details about the TX Buffer related registers.

35.6.3.12 Address scheme

Earlier serial flash memories supported only 24-bit address space hence restricting the maximum memory size of the serial flash as 16 MB. The new memory specification supports two types of 32-bit addressing mode in addition to legacy 24-bit address mode. It also supports segregation of address programmed, into Row address and Column address of the Flash, as per requirement.

- **Extended Address Mode**

  In this mode, the legacy 24-bit commands are converted to accept 32-bit address commands. The flash memory needs to be configured for 32-bit address mode. Also, while programming the LUT sequence in QuadSPI for 32-bit mode, the ADDR and ADDR_DDR command should be programmed with 8’d32 as the operand value with QSPI_SFACR[CAS] programmed to 0. If a flash needs some bits of the address as its column address, then it must always considered that the total bits required by the Flash should not exceed 32, as maximum address supported by QuadSPI is 32 bits. Each of the memory vendors have a different way of enabling this mode (Refer to the memory specification from memory vendors). For example, the command B7h sent to Macronix flash will enable it for 32-bit address mode.

- **Extended Address register**

  In this mode, the upper 8-bit of the 32-bit address is provided by the Extended address register in the memory itself. The memory provides a specific register which is updated according to the address to be accessed. This effectively converts the legacy 24-bit address command into 32-bit address commands. The memories greater in size than 16 MB, consists of banks of 16 MB. The 8-bit written in the extended address register effectively enables a bank. For example in Spansion memory, when the extended address register is updated with a value of 0x01 with the help of the command 17h, it will open Bank1 of the memory. The consequent 24-bit address commands will lead to Bank1. The extended address register needs to be update with the respective value for access to other banks. This effectively converts the legacy 24-bit address command into 32-bit address commands.

- **Separation of address into row and column address**

  This mode has been introduced for flashes which needs addresses segregated into Row and Column. The value in QSPI_SFACR[CAS] defines the width of the column address required by a flash. The actual address to be provided will be derived from the incoming address in case of AHB initiated transactions and the value of SFAR in case of IPS initiated transactions , if QSPI_SFACR[CAS] is set to 0, else the actual address will take CAS into consideration. If QSPI_SFACR[CAS] is 3 then bits 26-3 of the address programmed are sent to flash as it page address in case flash is operating in 24bit mode and bits 2-0 are sent as its column address. If a flash requirement for column address is less than the number of pads in which address has to be sent than the remaining bits are appended with 0 by QuadSPI. The user must program the operand value in CADDR and CADDR_DDR command accordingly. It must be ensured that the total number of address bits request by flash as its page and column address must not be more than 32 bits.

- **Word addressable mode for Flash**

  This mode has been introduced for flashes which has word addressable memory i.e. each address of the flash contains one word (two bytes) of data. The QSPI_SFACR [WA] is set to 1 to enter this mode. QuadSPI internally divides the incoming address in the AHB bus or the address in the QSPI_SFAR to map it to a valid flash location. For example, if the incoming address is 0x2004, the controller re-maps this address to access the flash location 0x1002. If not in this mode, the incoming address 0x2004 will be mapped to flash location 0x2004.

35.7 Initialization/Application Information

This section provides the initialization and application information of the QuadSPI module.

35.7.1 Power Up and Reset

Note that the serial flash devices connected to the QuadSPI module may require special voltage characteristics of their inputs during power up or reset. It is the responsibility of the application to ensure this.

35.7.2 Available Status/Flag Information

This paragraph gives an overview of the different status and flag information available and their interdependencies for different use cases. Related registers are QSPI_SR and QSPI_FR. Refer to the related descriptions how to set up the QuadSPI module appropriately.

35.7.2.1 IP Commands

Refer to IP Configuration Register (QuadSPI_IPCR) for additional details not explicitly covered in this paragraph.

- **IP Commands - Normal Operation**

  Writing the QSPI_IPCR[SEQID] field triggers the execution of a new IP Command. Given that this is a legal command the QSPI_SR[IPACC] and the QSPI_SR[BUSY] bits are asserted simultaneously, immediately after the execution is started.

  When the instruction on the serial flash device has been finished these bits are de-asserted and the QSPI_FR[TFF] flag is set.

- **IP Commands - Error Situations**

  Refer to Table 35-29 below.

35.7.2.2 AHB Commands

Refer to Section 1, Reading Serial Flash Data into the QuadSPI Module, in Flash Read for additional details not explicitly covered in this paragraph.

- **AHB Commands - Normal Operation**

  Memory mapped read access to a serial flash address not contained in the AHB Buffer, triggers the execution of an AHB Command. Given that this is a legal command the QSPI_SR[AHBACC] and the QSPI_SR[BUSY] bits are asserted simultaneously immediately after the execution is started. When the instruction on the serial flash device has been finished these bits are de-asserted.

- **IP Commands - Error Situations**

  Refer to Table 35-29 below.

35.7.2.3 Overview of Error Flags

The following table gives an overview of the different error flags in the QSPI_FR register and additional error-related details.

Table 35-29. Overview of QSPI_FR Error Flags

Error Category	Error Flag in QSPI_FR	Command Execution on Serial Flash Device TFF Behavior (in case of IP commands only)	Description
AHB Error Flag	ABSEF	Flash transaction is aborted	AHB sequence contains • WRITE instruction • WRITE_DDR instruction
AHB Error Flag	ABOF	Flash transaction continues until it finishes	Set when the module tried to push data into the AHB buffer that exceeded the size of the AHB buffer. Only occurs due to wrong programming of the QSPI_BUFxCR[ADATSZ].
AHB Error Flag	AIBSEF	Flash transaction is aborted	Total burst size of AHB transaction is greater than prefetch data size
AHB Error Flag	AITEF	Flash transaction is aborted	No response generated from QSPI to AHB bus in case of illegal transaction and the watchdog timer expires
Miscellaneous Error Flag	DLPFF	Flash transaction continues until it finishes	Set when DATA_LEARN instruction was encountered in a sequence but no sampling point was found for the data learning pattern.

Table continues on the next page...

Table 35-29. Overview of QSPI_FR Error Flags (continued)

Error Category	Error Flag in QSPI_FR	Command Execution on Serial Flash Device TFF Behavior (in case of IP commands only)	Description
Miscellaneous Error Flag	ILLINE	Flash transaction aborted	Illegal instruction Error Flag – Set when an illegal instruction is encountered by the controller in any of the sequences.
Command Arbitration Error	IPIEF	TFF not asserted in conjunction with that command	IP Command Error - caused when IP access is currently in progress (IP_ACC set) and • write attempt to QSPI_IPCR register. • write attempt to QSPI_SFAR register. • write attempt to QSPI_RBCT register.
Command Arbitration Error	IPAEF		• AHB Command already running, another IP Command could not be executed. • AHB Command already running, write attempt to QSPI_IPCR[SEQID] field.
Command Arbitration Error	IPGEF		• Exclusive access to the serial flash granted for AHB Commands, write attempt to QSPI_IPCR[SEQID] field.
IP Command Error	IUEF	—	• IP Command Usage Error
Buffer Related Error	RBOF	TFF is asserted on completion	• RX Buffer Overrun
Buffer Related Error	TBUF		• TX Buffer Underrun

Note that only the buffer related errors are related to a transaction on the external serial flash. All the other errors do not trigger an actual transaction.

35.7.2.4 IP Bus and AHB Access Command Collisions

There are two flags related to this topic, the QSPI_FR[IPAEF] and QSPI_FR[IPIEF]. Refer to sub-section "Reading Serial Flash Data into the QuadSPI Module" of Flash Read section, for a description of the flags and Command Arbitration , for details about possible command collisions.

35.7.3 Exclusive Access to Serial Flash for AHB Commands

It is possible that several masters need to access the serial flash device connected to the QuadSPI module separately, one master by triggering IP Commands and reading the RX Buffer (via RBDR*n* register) and the other masters by triggering AHB Commands (via ARDB*n* Registers). These two set of buffer (RBDR and ARDB Buffer) points to the same physical buffer.Refer to Figure 35-7 To avoid command collisions resulting in excessive latencies the QuadSPI module implements a request-handshake mechanism between the master triggering AHB Commands and the QuadSPI module allowing this specific master to request exclusive access to the serial flash device for AHB Commands. If this exclusive access is granted the execution of IP Commands is blocked. This resolves command collisions and excessive times where the AHB interface may be blocked.

If this capability is used in the device there is additional status and flag information available related to this mechanism. The QSPI_SR[AHBGNT] bit reflects the module-internal state that the exclusive access mentioned above is granted, any attempt to trigger an IP Command is rejected and results in the assertion of the QSPI_FR[IPGEF] flag. Refer to the descriptions of the related bit and flag for details.

It is within the responsibility of the application to set up the master using this mechanism appropriately, if used incorrectly no IP Commands at all can be triggered.

Two different cases can be distinguished:

35.7.3.1 RX Buffer Read via QSPI_ARDB Registers

In this case all masters share the AHB bus for RX Buffer as well as for AHB Buffer read. In this case the access to the AHB interface by the master triggering AHB Commands must be deferred until any pending IP Command has been finished **and** the RX Buffer readout has been finished as well. The QSPI ARDB Buffers access the Rx buffer i.e the data from the Rx Buffer is returned and no data from AHB Buffer is touched. This is the conservative use case, corresponding to the reset value 0 of the QSPI_RBCT[RXBRD] bit.

In this case the QSPI_SR[AHBGNT] bit is asserted not earlier than any running IP Command has been finished (QSPI_SR[IP_ACC] is 0), the RX Buffer has been read out completely (QSPI_RBSR[RDBFL] equal to 0) or no DMA read is pending (QSPI_SR[RXDMA] equal to 0 and Rx Buffer readout is via AHB(QSPI_RBCT[RXBRD]) equal to 1.

35.7.3.2 RX Buffer Read via QSPI_RBDR Registers

This is the preferred use case as an access to the AHB buffer (memory mapped flash) does not interfere with any IPS access to read the RBDR buffer. It is not possible that a pending AHB bus access triggered by an AHB Command stalls the AHB bus and blocks the RX Buffer readout since the RX Buffer is read via the IP bus based registers QSPI_RBDR0 to QSPI_RBDR15.

For this case it is recommended to program the QSPI_RBCT[RXBRD] bit to 1. The QSPI_SR[AHBGNT] bit is asserted immediately after any running IP Command has been finished (QSPI_SR[IP_ACC] is 0), the RX Buffer has been read out completely (QSPI_RBSR[RDBFL] equal to 0) or no DMA read is pending (QSPI_SR[RXDMA] equal to 0, allowing the master triggering AHB Commands to trigger AHB Commands as soon as possible without the need to wait for the RX Buffer readout to be finished.

35.7.4 Command Arbitration

In case of overlapping commands, the arbitration scheme is described in the following paragraphs under the assumption that the priority mechanism described in Exclusive Access to Serial Flash for AHB Commands is **not** used:

- During the execution of an IP Command, the running IP Command can't be terminated by issuing another IP Command or AHB Command. The QSPI_FR[IPIEF] flag is asserted when the host tries to write into the QSPI_IPCR register. When the host triggers an AHB Command (refer to sub-section "Reading Serial Flash Data into the QuadSPI Module" of Flash Read section, for details), this command is stalled until the currently running IP Command is finished.
- During the execution of an AHB Command, the running AHB Command can't be terminated by issuing an IP Command. The command is ignored and the QSPI_FR[IPAEF] flag is asserted. Refer to Flag Register (QuadSPI_FR) for the description of these flags.

  When another AHB Command is triggered the address of the memory mapped access is considered. If the requested address is currently read from the serial flash device, the running command is continued. If this is not the case the currently running command is terminated and another AHB Command related to the requested address is executed. Refer to sub-section "Reading Serial Flash Data into the QuadSPI Module" of Flash Read section, for further details.

In case of coinciding commands the IP Command is triggered and the AHB Command is stalled until the IP Command has been finished (QSPI_SR[IP_ACC] has been deasserted).

The IP Commands ignored in case of command collision will not result in the assertion of the QSPI_FR[TFF] flag.

35.7.5 Flash Device Selection

Regardless of the SFM Command (IP or AHB) the access mode is selected by specifying the 32 bit address value for the following SFM Command.

For IP Commands the access mode is selected with the address programmed into the QSPI_SFAR register. Refer to Serial Flash Address Register (QuadSPI_SFAR) for details.

For AHB Commands the access mode is determined by the memory mapped address which is accessed Refer to AMBA Bus Register Memory Map for details.

35.7.6 DMA Usage

For the complete description of the DMA module refer to the related DMA Controller chapter. In this paragraph only the details specific to the DMA usage related to the QuadSPI module are given.

35.7.6.1 DMA Usage in Normal Mode

35.7.6.1.1 Bandwidth considerations

Careful consideration of the throughput rate of the entire chain (serial flash -> AHB bus / IP Bus -> DMA controller) involved in the read/write data process is essential for proper operation. Such analysis must take into account not only the data rate provided by the serial flash but also the data rate of the AHB bus and the performance of the DMA controller in reading/writing data from/to the RX/TX buffer.

Two figures must match for proper operation, that means that the data rate provided by the serial flash device must not exceed the average RX Buffer readout data rate. Otherwise, the longer this state persists, a RX Buffer overflow will result. Similarly, the data consumed by the serial flash device must not exceed the average TX buffer fill rate. If this persists, it would eventually lead to an underrun.

AHB Bus Side (data read):

The total number of bus cycles for each DMA Minor Loop completion is added from the following components:

- Overhead for each minor loop, given by DMA controller: Assume 10 cycles
- Overhead due to clock domain crossing: Assume 2 cycles
- Number of bus clock cycles required for 8 bytes (64 bit read size): Assume 2 cycles (read/write sequence of DMA controller)

Note that the size of the minor loop is determined by the size of the QSPI_RBCT[WMRK] field, therefore the overhead given above distributes among (QSPI_RBCT[WMRK]+1)/2 read accesses of 64 bit each.

The following table gives some examples for typical use cases:

Table 35-30. Access Duration Examples - Bus Clock Side

QSPI_RBCT[WMRK]	Number of Bytes per DMA Loop [1]	Number of Bus Clock cycles for DMA Minor Loop	Time Duration of DMA Minor Loop for 120Mhz Bus clock Frequency
0	4	12+2 = 14	~117ns
1	8	12+2 = 14	~117ns
3	16	12+4 = 16	~133ns
7	32	12+8 = 20	~167ns
11	48	12+12 = 24	~200ns

1. DMA Loop means one Minor Loop Completion which is equivalent to one.

NOTE

The table figure represents ideal scenario, actual performance will depend on how the system is integrated.

Serial Flash Device Side (data read):

The number of serial flash cycles can be determined in the following way:

- Number of serial flash clock cycles required to read 4 bytes, corresponding to one RX Buffer entry (setup of command and address not considered): 2 cycles for Quad DDR mode instructions in Parallel Flash Mode, 2 cycles for Octal DDR mode (Hyperflash) in individual flash mode, 4 cycles for Quad (SDR) mode instruction in parallel flash mode or Dual IO DDR mode instruction in parallel flash mode, 8 cycles for Quad Mode (SDR) instructions in Individual Flash Mode etc.
- Overhead due to clock domain crossing : 1 cycle.

The following table lists the number of clock cycles required to read the data from the serial flash corresponding to the different settings of the QSPI_RBCT[WMRK] field:

Table 35-31. Access Duration Examples - Serial Flash side

QSPI_RBCT[WMRK] setting	Num Bytes per DMA Loop [1]	Num SCKFx for 80MHz SCKFx			Time duration of Flash data readout for 80MHz SCKFx (~12.5ns period)		
		IFM [2] Quad	IFM Quad DDR	PFM [3] Quad DDR/ IFM Octal DDR	IFM Quad	IFM Quad DDR	PFM Quad DDR
0	4	8	4	2	~100ns	~50ns	~25ns
1	8	16	8	4	~200ns	~100ns	~50ns
3	16	32	16	8	~400ns	~200ns	~100ns
7	32	64	32	16	~800ns	~400ns	~200ns
11	48	96	48	24	~1200ns	~600ns	~300ns

1. DMA Loop means one Minor loop completion which is equivalent to one Major Loop iteration.
2. Individual flash mode.
3. Parallel flash mode.

NOTE

The table figure represents ideal scenario, actual performance will depend on how the system is integrated.

From the examples given in the two tables above, it can be seen that depending on the relationship between the Bus clock and Serial flash clock frequencies, there are settings possible where the serial flash provides the read data faster than the AHB bus can read out the RX buffer. In the above tables, it is the case of PFM Quad DDR mode with Watermark up to 3 and other cases. In these cases, the RX buffer data keeps accumulating over time and will eventually overflow. To avoid RX Buffer overflow, the data transaction size should be small enough.

A complementary example would be when the watermark is set to be too high. In such a case, the time taken by the DMA to read out the RX buffer entries should be smaller than the time taken by the controller to push in the remaining entries in the buffer.

IPS Bus Side (data write):

The total number of bus cycles for each DMA Minor Loop completion is added from the following components:

- Overhead for each minor loop, given by DMA controller: Assume 10 cycles
- Overhead due to clock domain crossing: Assume 2 cycles
- Number of bus clock cycles required for 16 bytes (128 bit write size): Assume 4 cycles (read/write sequence of DMA controller)

Note that the size of the minor loop is determined by the size of the QSPI_TBCT[WMRK] field, therefore the overhead given above distributes among (QSPI_TBCT[WMRK]+1) write accesses of 32 bit each.

The following table gives some examples for typical use cases:

Table 35-32. Access Duration Examples - Bus Clock Side

QSPI_TBCT[WMRK]	Number of Bytes per DMA Loop [1]	Number of Bus Clock cycles for DMA Minor Loop	Time Duration of DMA Minor Loop for 80Mhz Bus clock Frequency
3	16	12+4 = 16	~200ns
7	32	12+8 = 20	~250ns
11	48	12+12 = 24	~300ns
15	64	12+16 = 28	~350ns

1. DMA Loop means one Minor Loop Completion which is equivalent to one.

NOTE

The table figure represents ideal scenario, actual performance will depend on how the system is integrated.

Serial Flash Device Side (data write):

The number of serial flash cycles can be determined in the following way:

- Number of serial flash clock cycles required to write 16 bytes, corresponding to four TX Buffer entry(setup of command and address not considered): 8 cycles for Octal DDR mode (Hyperflash) instructions in Individual Flash Mode, 32 cycles for Quad SDR writes in individual flash mode, 64 cycles in single io SDR writes in parallel flash mode.
- Overhead due to clock domain crossing : 1 cycle.

The following table lists the number of clock cycles required to read the data from the serial flash corresponding to the different settings of the QSPI_TBCT[WMRK] field:

Table 35-33. Access Duration Examples - Serial Flash side

QSPI_TBCT[WMRK] setting	Num Bytes per DMA Loop [1]	Num SCKFx			Time duration for consuming data at Flash interface 100MHz SCKFx (10ns period)[2]			Time for fifo to get empty[3]		
		IFM [4] Quad	IFM Octal DDR	[5] Single IO SDR Mode	IFM Quad	IFM Octal DDR	PFM Single IO SDR	IFM Quad	IFM Octal DDR	PFM Single IO SDR
3	16	32	8	64	320ns	80ns	640ns	2240ns	560ns	4480ns
7	32	64	16	128	640ns	160ns	1280ns	1920ns	480ns	3840ns
15	64	128	32	256	1280ns	320ns	2560ns	1280ns	320ns	2560ns

1. DMA Loop means one Minor loop completion which is equivalent to one Major Loop iteration.
2. Not all flash devices support writes at 100Mhz. Please refer to the flash datasheet for the actual page program frequency supported.
3. The assumption for these timings is that the TX Fifo was full when the transaction was initiated
4. Individual flash mode.
5. Parallel flash mode.

NOTE

The tables mentioned above are only examples which must be correlated with the DMA in the system.

From the examples given in the two tables above for TX fifo, it can be seen that depending on the relationship between the Bus clock and Serial flash clock frequencies, there are settings possible where the serial flash consumes data faster than the IPS bus can write data in TX buffer. In these cases, a TX buffer underrun situation will occur. To avoid TX Buffer underrun, the data transaction size should be large enough.

35.7.7 Parallel mode

QuadSPI can access two flashes in parallel. This increases the throughput of the QuadSPI by two times. Only read operations and x1 mode write are allowed in parallel mode. In case a write transaction is initiated in with mode than one pad in parallel mode, QSPI_FR[IUEF] is set. When dual die flashes are accessed in parallel mode, it is mandatory for flash A1 to be of the same size as B1 and A2 to be of the same size as B2. The following figure shows how QuadSPI maps the incoming addresses to the different flashes connected on board.

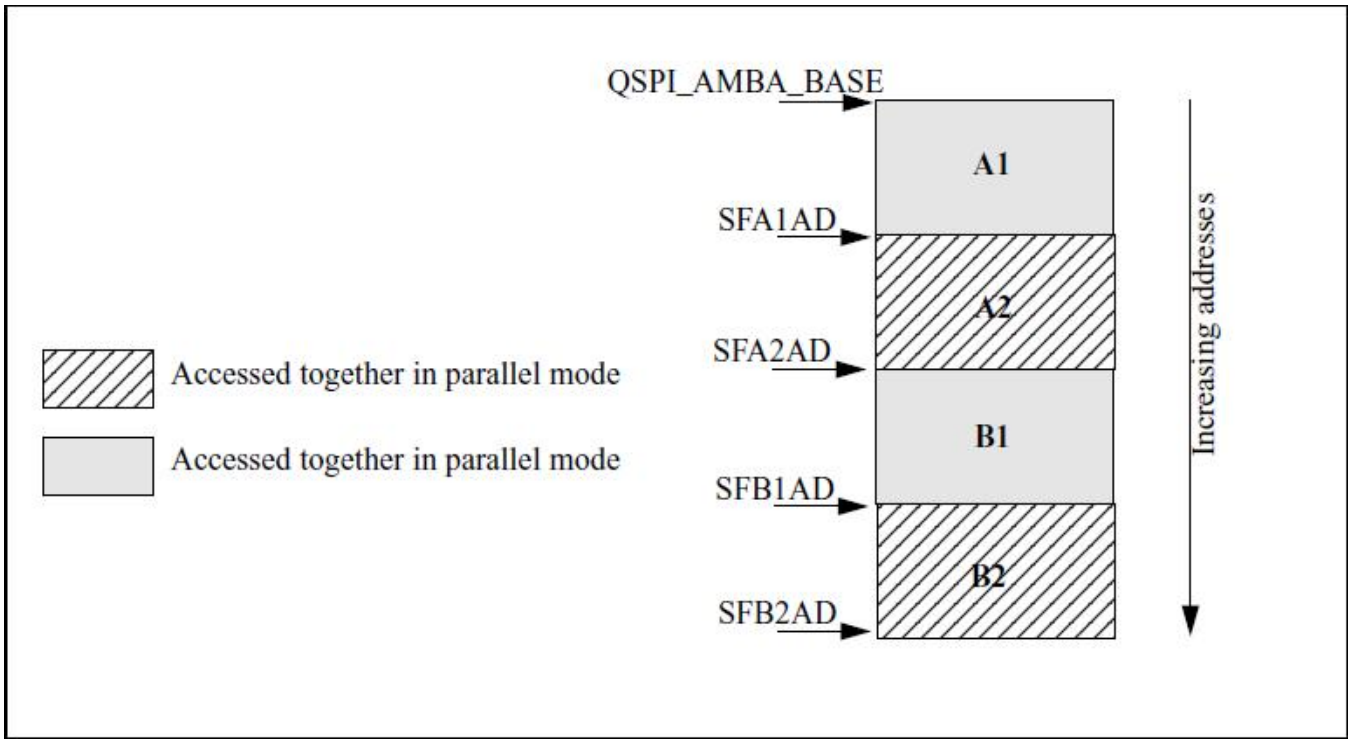


Figure 35-8. Flash addressing

An example programming for parallel mode access is given below (flash sizes are assumed to be 256MB):

- QSPI_AMBA_BASE - 0x10000000
- QSPI_SFA1AD[TPADA1] - 0x20000000
- QSPI_SFA2AD[TPADA2] - 0x30000000
- QSPI_SFB1AD[TPADB1] - 0x40000000
- QSPI_SFB2AD[TPADB2] - 0x50000000

In order to access the first location of A1/B1 pair, the incoming address should be 0x10000000. QSPI_AMBA_BASE is subtracted from this address and the result is divided by two. Therefore, address provided to flash A1 and B1

Flash Address = (Memory mapped address - QSPI_AMBA_BASE)/2

For Memory Mapped address:

- 0x10000000, flash address: 0x0 (Or, the first address of flash A1 and B1)
- 0x10000004, flash address: 0x2
- 0x10000008, flash address: 0x4 etc.

Similarly, in order to access the first location of A2/B2 pair, the incoming address should be 0x30000000.

Flash Address = (Memory mapped address - SFA2AD)/2

For Memory Mapped address:

- 0x30000000, flash address: 0x0 (Or, the first address of flash A2 and B2)
- 0x30000004, flash address: 0x2
- 0x30000008, flash address: 0x4 etc.

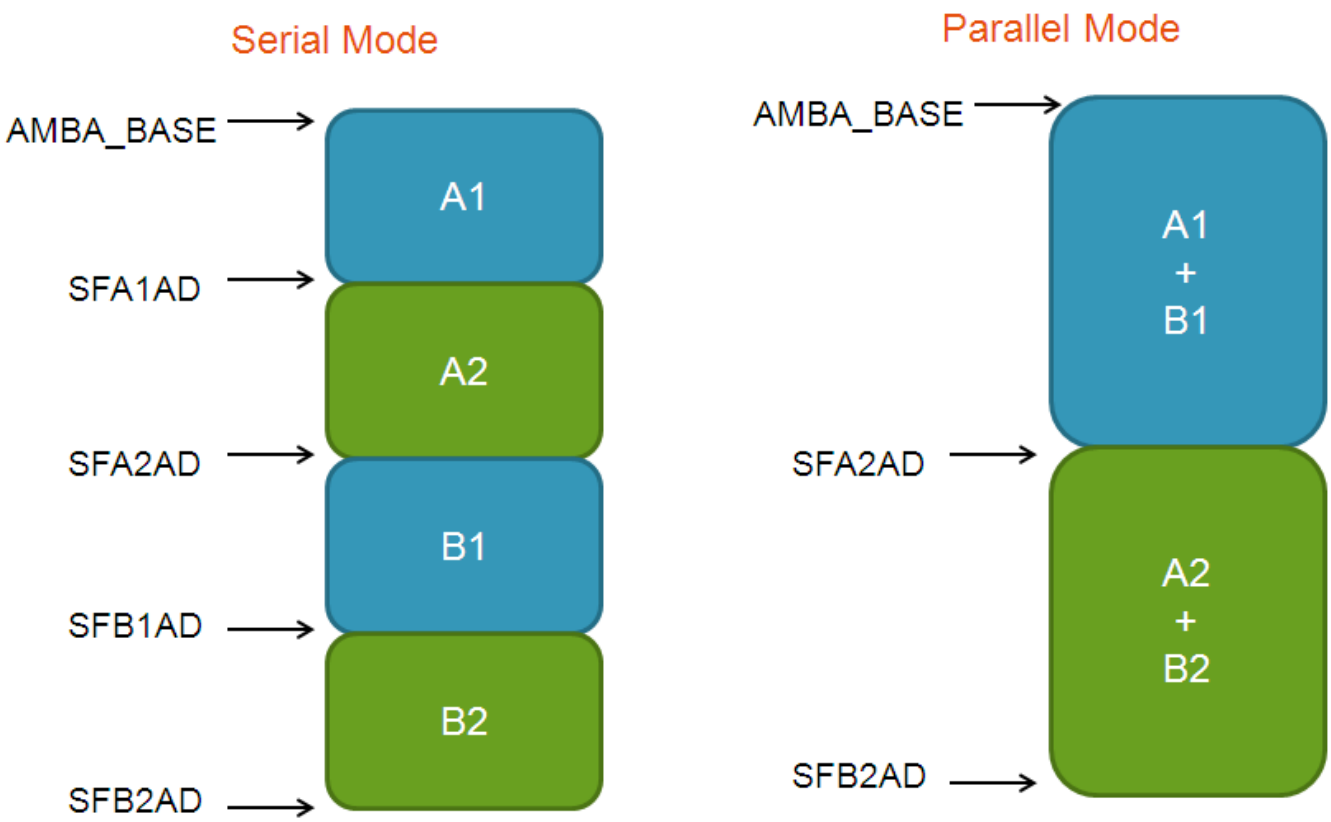


Figure 35-9. Memory map - Serial and Parallel

Software must ensure that when multiple flashes are used, accesses should never cross a boundary between separate flash devices. For example, in the above figure, for serial mode, accesses must not cross the A1 to A2, A2 to B1 or B1 to B2 boundaries. For parallel mode, accesses must not cross the A1+B1 to A2+B2 boundary. Software can manage this via the following methods:

- Reduce the data fetch amount of the AHB buffers to 64 bit so that no prefetch occurs. This will prevent any legal access from crossing the boundary
- If prefetch is enabled then reserve a memory region, the maximum size of the prefetch, prior to the boundary. So long as no accesses are made to this region of memory, no access will cross the boundary
- Ensure that cache is not prefetching across the boundary

35.8 Byte Ordering - Endianness

QuadSPI provides support for swapping the flash read/write data based on the configuration of the QSPI_MCR[END_CFG]. By default the data is always returned in 64 bit LE format on the AHB bus and 32 bit LE format on the IPS interface when read via the RX buffer and written in 32 bit LE format when written via the TX buffer.

The table(QSPI_MCR[END_CFG]) below shows the complete bit ordering. BE signifies Big Endian which means the high order bits of the associated data vectors are associated with low order address positions. LE signifies Little Endian which means the lower order bits of the associated data vectors are associated with low order address positions.Refer to figure Figure 35-7

Table 35-34. QSPI_MCR[END_CFG]

00	64 bit BE
01	32 bit LE
10	32 bit BE
11	64 bit LE

The tables below (Byte ordering configuration in AHB) and (Byte ordering configuration in IPS) show how this configuration is implemented in QSPI AHB and IPS interfaces respectively. B in the table signifies Byte and the index 1-8 refers to the byte position i.e. 1 refer to bits[7:0], 8 refer to bits[63:56] and so on.

Table 35-35. Byte ordering configuration in AHB

64 bit BE	B1	B2	B3	B4	B5	B6	B7	B8
64 bit LE	B8	B7	B6	B5	B4	B3	B2	B1
32 bit BE	B5	B6	B7	B8	B1	B2	B3	B4
32 bit LE	B4	B3	B2	B1	B8	B7	B6	B5

Table 35-36. Byte ordering configuration in IPS

32BE	B1	B2	B3	B4
32LE	B4	B3	B2	B1

The examples below show the byte ordering in 64 bit BE configuration for AHB Buffer and 32 bit BE for TX/RX Buffer:

35.8.1 Programming Flash Data

CPU write instructions to the QSPI_TBDR register like

- Write QSPI_TBDR -> 0x01_02_03_04
- Write QSPI_TBDR -> 0x05_06_07_08

result in the following content of the TX Buffer:

Table 35-37. Example of QuadSPI TX Buffer

TX Buffer Entry	Content
0	32'h01_02_03_04
1	32'h05_06_07_08

Programming the TX Buffer into the external serial flash device results in the following byte order to be sent to the serial flash:

- 01…02…03…04…05…06…07…08

35.8.2 Reading Flash Data into the RX Buffer

Reading the content from the same address provides the following sequence of bytes, identical to the write case:

- 01…02…03…04…05…06…07…08

This results in the RX Buffer filled with:

Table 35-38. Resulting RX Buffer Content

RX Buffer Entry	Content
0	32'h01_02_03_04
1	32'h05_06_07_08

35.8.2.1 Readout of the RX Buffer via QSPI_RBDRn

The RX Buffer content appears at CPU read access via the Peripheral bus interface in the following order:

- Read QSPI_RBDR0 <- 0x01_02_03_04
- Read QSPI_RBDR1 <- 0x05_06_07_08

35.8.2.2 Readout of the RX Buffer via ARDBn

The RX Buffer content appears at read access on the AMBA AHB interface at the QuadSPI module boundary:

- (1a): 32 Bit Access: Read QSPI_ARDB0 <- 0x01_02_03_04
- (2a): 32 Bit Access: Read QSPI_ARDB1 <- 0x05_06_07_08
- (1b/2b): 64 Bit Access: Read QSPI_ARDB0 <- 0x01_02_03_04_05_06_07_08

35.8.3 Reading Flash Data into the AHB Buffer

Reading the content from the same address as it was written to provides the following sequence of bytes, identical to the write case:

- 01…02…03…04…05…06…07…08

This results in the AHB Buffer filled with:

Table 35-39. Resulting AHB Buffer Content

AHB Buffer Entry	Content
0	64'h01_02_03_04_05_06_07_08

35.8.3.1 Readout of the AHB Buffer via Memory Mapped Read

The AHB Buffer content appears at read access on the AMBA AHB interface at the QuadSPI module boundary:

- (1a): 32 Bit Read Access: <- 0x01_02_03_04
- (2a): 32 Bit Read Access: <- 0x05_06_07_08
- (1/2): 64 Bit Read Access: <- 0x01_02_03_04_05_06_07_08

35.9 Serial Flash Devices

Several different vendors make flash devices with a QuadSPI interface. At present there is no set standard for the QuadSPI instruction set. Most common commands currently have the same instruction code for all vendors, however some commands are unique to specific vendors. Some example sequences are provided below.

35.9.1 Example Sequences

This section provides the example sequences of the QuadSPI module.

Table 35-40. Exit 4 x I/O Read Enhance Performance Mode (XIP) (Macronix) and Read Status

INSTR	PAD	OPERAND	COMMENT
CMD	0x0	0xEB	4xIO Read Command
ADDR	0x2	0x18	24 Bit address to be send on 4 pads
MODE	0x2	0x00	2 mode cycles (exit XIP)
DUMMY	0x0	0x04	4 dummy cycles
READ	0x2	0x08	Read 64 bits
CMD	0x0	0x05	Read Status register
READ	0x0	0x01	Status register data
STOP	0x0	0x00	STOP, Instruction over

35.9.1.1 Read Command (Spansion Hyperflash)

This section provides the read command sequences (Spansion Hyperflash) of the QuadSPI module.

Table 35-41. Read Command (Spansion Hyperflash)

INSTR	PAD	OPERAND	COMMENT
CMD_DDR	0x3	0xA0	Read command with continuous burst type
ADDR_DDR	0x3	0x18	24 bit row address
CADDR_DDR	0x3	0x10	16 bit column address with lower 3 bits valid rest 0
DUMMY	0x3	0x0F	15 dummy cycles
READ_DDR	0x3	0x4	32 bit data read on 8 pads
STOP	0x3	0x00	STOP, Instruction over

Hyperflash is a word addressable flash i.e. each address accesses a word wide (2 bytes) data value, the software should ensure that when Hyperflash is connected to the controller, the QSPI_SFACR [WA] bit must be set. If this bit is set the controller remaps a byte addressable access to a word addressable access.

35.9.1.2 Read Status Register(Spansion Hyperflash)

This section provides the read status register of the QuadSPI module.

Table 35-42. Read Status register (Spansion Hyperflash)

INSTR	INSTR SEQUENCE	PAD	OPERAND	COMMENT
CMD_DDR	Read Pre Comman	0x3	0x00	Write command with wrapped burst type.
ADDR_DDR		0x3	0x18	24 bit row address(0000AAh)
CADDR_DDR		0x3	0x10	16 bit column address with lower 3 bits valid rest 0(0005h),treated as command
CMD_DDR		0x3	0x00	Write command with wrapped burst type
CMD_DDR		0x3	0x70	Write data to be sent to flash as pre-command
CMD_DDR	Command phase (fourth/final chip select phase)	0x3	0xA0	Read command with continuous burst type
ADDR_DDR		0x3	0x18	24 bit row address
CADDR_DDR		0x3	0x10	16 bit column address with lower 3 bits valid rest 0
DUMMY		0x3	0x0F	15 dummy cycles
READ_DDR		0x3	0x4	32 bit data read on 8 pads
STOP		0x3	0x00	STOP, Instruction over

35.9.1.3 Word Program (Spansion Hyperflash)

This section provides the Word Program (Spansion Hyperflash) of the QuadSPI module.

Table 35-43. Word Program (Spansion Hyperflash)

INSTR	INSTR SEQUENCE	PAD	OPERAND	COMMENT
CMD_DDR	Unlock Sequence 1 (first chip select phase)	0x3	0x00	Write command with wrapped burst type

Table continues on the next page...

Table 35-43. Word Program (Spansion Hyperflash) (continued)

CMD_DDR		0x3	0x00	8 bit address 00h treated as command
CMD_DDR		0x3	0x00	8 bit address 00h treated as command
CMD_DDR		0x3	0xAA	8 bit address AAh treated as command
CADDR_DDR		0x3	0x10	16 bit column address with lower 3 bits valid rest 0(0005h),treated as command
CMD_DDR		0x3	0x00	Write command with wrapped burst type.
CMD_DDR		0x3	0xAA	Write data to be sent to flash as pre-command.
CMD_DDR	Unlock Sequence 2 (second chip select phase)	0x3	0x00	Write command with wrapped burst type
CMD_DDR		0x3	0x00	8 bit address 00h treated as command
CMD_DDR		0x3	0x00	8 bit address 00h treated as command
CMD_DDR		0x3	0x55	8 bit address 55h treated as command
CADDR_DDR		0x3	0x10	16 bit column address with lower 3 bits valid rest 0(0002h),treated as command
CMD_DDR		0x3	0x00	Write command with wrapped burst type
CMD_DDR		0x3	0x55	Write data to be sent to flash as pre-command
CMD_DDR	Program setup phase (third chip select phase)	0x3	0x00	Write command with wrapped burst type
CMD_DDR		0x3	0x00	8 bit address 00h treated as command
CMD_DDR		0x3	0x00	8 bit address 00h treated as command
CMD_DDR		0x3	0xAA	8 bit address AAh treated as command
CADDR_DDR		0x3	0x10	16 bit column address with lower 3 bits valid rest 0(0005h),treated as command
CMD_DDR		0x3	0x00	Write command with wrapped burst type
CMD_DDR		0x3	0xA0	Write data to be sent to flash as pre-command
CMD_DDR	Command phase (fourth/final chip select phase)	0x3	0x00	Write command with wrapped burst type

Table continues on the next page...

Table 35-43. Word Program (Spansion Hyperflash) (continued)

ADDR_DDR		0x3	0x18	24 bit row address
CADDR_DDR		0x3	0x10	16 bit column address with lower 3 bits valid rest 0
WRITE_DDR		0x3	0x2	2 bytes data written on 8 pads (D1D2)
STOP		0x3	0x00	STOP, Instruction over

35.9.1.4 Fast Read Sequence (Macronix/Numonyx/Spansion/ Winbond)

The following table shows the fast read sequence for Macronix/Numonyx/Spansion/ Winbond flashes.

Table 35-44. Fast Read sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0x0B	Fast Read command = 0x0B
ADDR	0x0	0x18	24 Addr bits to be sent on one pad
DUMMY	0x0	0x08	8 Dummy cycles
READ	0x0	0x04	Read 32 Bits on one pad
JMP_ON_CS	0x0	0x00	Jump to instruction 0 (CMD)

35.9.1.5 Fast Dual I/O DT Read Sequence (Macronix)

The following table shows the Fast Dual I/O DT read sequence for Macronix flashes.

Table 35-45. Fast Dual I/O DT Read sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0xBD	Fast Dual I/O DT read command = 0xBD
ADDR_DDR	0x1	0x18	24 Addr bits to be sent on 2 pads in DDR mode
MODE4_DDR	0x1	0x00	P2=P0 or P3=P1 is necessary. Refer to Macronix datasheet for details. One clock cycle for mode.
DUMMY	0x0	0x06	6 Dummy cycles
READ_DDR	0x1	0x04	Read 32 Bits on 2 pads in DDR mode
JMP_ON_CS	0x0	0x00	Jump to instruction 0 (CMD)

35.9.1.6 Fast Read Quad Output (Winbond)

The following table shows the Fast read quad output sequence for Winbond memories

Table 35-46. Fast Read Quad output sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0x6B	Fast read quad output command = 0x6B
ADDR	0x0	0x18	24 Addr bits to be sent on 1 pad
DUMMY	0x0	0x08	8 Dummy cycles
READ	0x2	0x04	Read 32 Bits on 4 pads
JMP_ON_CS	0x0	0x00	Jump to instruction 0 (CMD)

35.9.1.7 4 x I/O Read Enhance Performance Mode (XIP) (Macronix)

The following table shows the 4 x I/O Read Enhance Performance Mode for Macronix flashes. The enhanced performance mode is also known as XIP mode.

Table 35-47. Fast Read Quad output sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0xEB	4xI/O Read command = 0xEB
ADDR	0x2	0x18	24 Addr bits to be sent on 4 pads
MODE	0x2	0xA5	2 mode cycles
DUMMY	0x0	0x04	4 Dummy cycles
READ	0x2	0x04	Read 32 Bits on 4 pads
JMP_ON_CS	0x0	0x01	Jump to instruction 1 (ADDR)

When in XIP mode the software should ensure that all the flashes connected to the controller are in XIP mode. As a part of initializing the controller, all the flashes may be enabled with XIP by carrying out dummy reads.

35.9.1.8 Dual Command Page Program (Numonyx)

The following table shows the Dual command page program sequence for Numonyx flashes.

Table 35-48. Dual Command Page Program sequence

Instruction	Pad	Operand	Comment
CMD	0x1	0x02	Dual command page program = 0x02 on 2 pads

Table continues on the next page...

Table 35-48. Dual Command Page Program sequence (continued)

Instruction	Pad	Operand	Comment
ADDR	0x1	0x18	24 Addr bits to be sent on 2 pads
WRITE	0x1	0x20	Write 32 Bytes on 2 pads
STOP	0x0	0x00	STOP, Instruction over

35.9.1.9 Sector Erase (Macronix/Spansion/Numonyx)

The following table shows the Sector erase sequence for Macronix/Spansion/Numonyx flashes

Table 35-49. Sector Erase sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0xD8	Sector erase command = 0xD8
ADDR	0x0	0x18	24 Addr bits to be sent on 1 pad
STOP	0x0	0x00	STOP, Instruction over

35.9.1.10 Read Status Register (Macronix/Spansion/Numonyx/ Winbond)

The following table shows the Read status register sequence for Macronix/Spansion/ Numonyx/Winbond flashes.

Table 35-50. Read Status Register Sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0x05	Read status register command = 0x05
READ	0x0	0x01	Read status register data
STOP	0x0	0x00	STOP, Instruction over

35.9.1.11 Data Learn Instruction Sequence

The following table shows the data learn sequence for 4 I/O flash.

Table 35-51. Data learn Instruction sequence (4 I/O)

Instruction	Pad	Operand	Comment
CMD	0x0	0x6B	Fast Read Quad command = 0x6B
ADDR_DDR	0x2	0x18	24 address bits in DDR mode to be sent on four pads
DUMMY	0x2	0x08	8 Dummy cycle
DATA_LEARN	0x2	0x01	1 Byte data learn
READ_DDR	0x2	0x06	Read 6 bytes in 4 pads
JMP_ON_CS	0x0	0x00	Jump to Inst 0 (CMD)

The following table shows the data learn sequence for 8 I/O (Spansion Hyperflash) flash.

Table 35-52. Data learn Instruction sequence (8 I/O)

Instruction	Pad	Operand	Comment
CMD_DDR	0x3	0xAO	Read command for Hyperflash
ADDR_DDR	0x3	0x18	24 bit row address[1]
CADDR_DDR	0x3	0x10	16 bit column address
DUMMY	0x3	0x0F	15 cycle dummy
DATA_LEARN	0x3	0x01	1 byte data learn
READ_DDR	0x3	0x08	Read 8 bytes in 8 pads
STOP	0x3	0x00	STOP, instruction over

1. Address needs to be aligned i.e. No latency should be there between the RWDS edges.

35.9.2 Dual Die Flashes

Certain serial flash vendors provide dual-die packages which are essentially two devices (dies) stacked within the same package to increase the memory capacity of a single package. These two devices within a package share the same data and clock pins, but have individual Chip Selects. QuadSPI controller provides support for two dual-die packages to be connected simultaneously. The figure below shows the two dual-die packages and the naming conventions used in this document. For simplicity, the data pins are shown to be unidirectional.

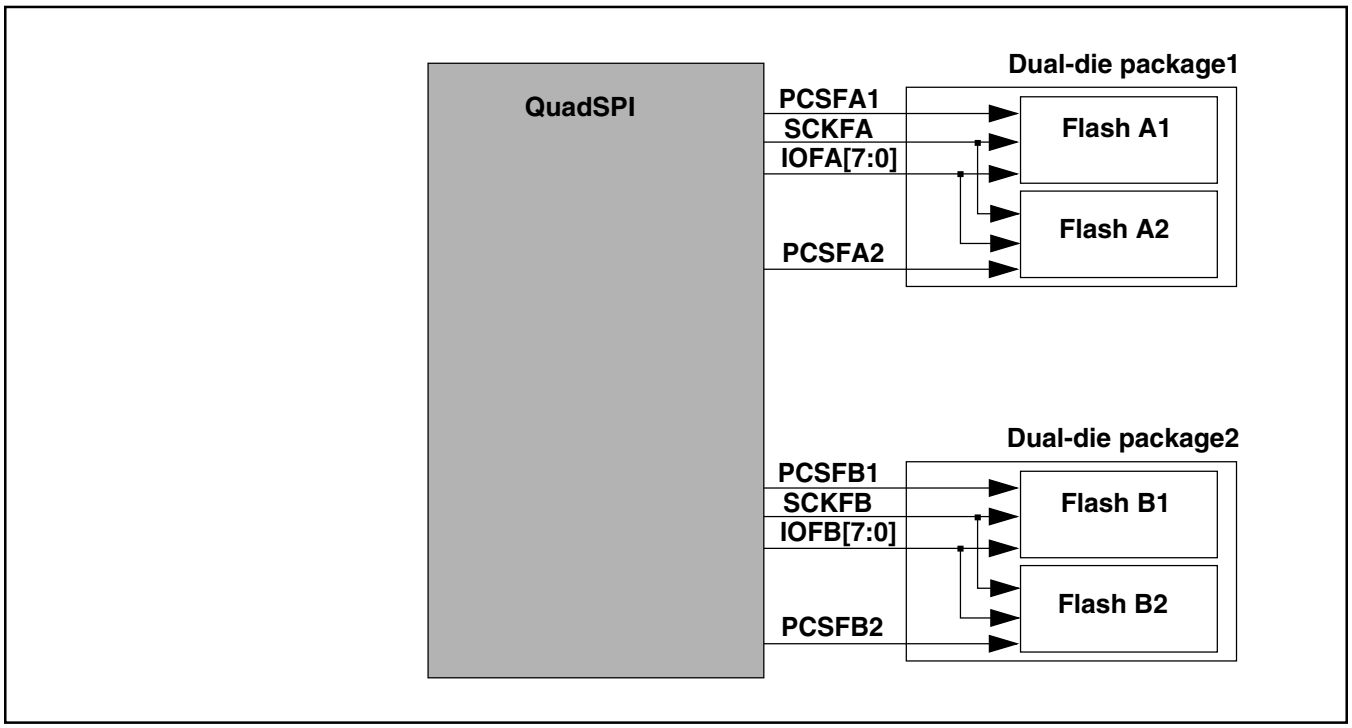


Figure 35-10. Dual-die support

Since the two devices within one package share the same i/o pads, they cannot function in parallel mode. Software should ensure that when QuadSPI is configured in parallel mode the two selected flash devices are from different dual-die packages.

35.9.3 Boot initialization sequence

The following are the recommended sequence of steps for booting from QuadSPI:

- System out of reset and flash available (300us)
- Clocks still at very low frequency. Clock tree configured, I/O pins configured. First request sent to QuadSPI for address 0x0 of flash.
- The reset command sequence in QuadSPI has 0x03 (basic read command) which is applicable to all flashes at < 50MHz serial flash clock
- The first few bytes of data is read from the flash which contains the following information:
  - The total sizes of all the flashes connected on board
  - Whether DDR mode supported
  - Frequency of DDR operation

    - Continuous mode entry sequence
    - 24bit or 32bit addressing (assuming 24bit for first accesses)
- All the serial flashes are configured
    - Quad Mode enabled
    - Dummy reads to enter into XIP
- QuadSPI is configured
    - Parallel enable set
    - LUT configured for highest performance reads
    - DDR mode enabled (if applicable)
    - Buffers configured
- Serial flash clock frequency increased.
- Boot reads happen in parallel, DDR enabled, quad output mode @66MHz.

35.9.4 Serial Flash Clock Frequency Limitations

Certain commands of some serial flash devices are limited in the frequency applied to the serial flash device on command execution. In order to support these commands without having to recalibrate the module clocks, the the serial flash device clock can be divided by 2 (half speed) by setting the QSPI_SMPR[HSENA] bit. The SCLK will return to full speed once the the QSPI_SMPR[HSENA] bit is cleared.

35.10 Sampling of Serial Flash Input Data

35.10.1 Basic Description

QuadSPI is used to read data from the serial flash device. Depending on the actual implementation, there is a delay between the internal clocking in the QuadSPI module and the external serial flash device. Refer to the following figure for an overview of this scheme.

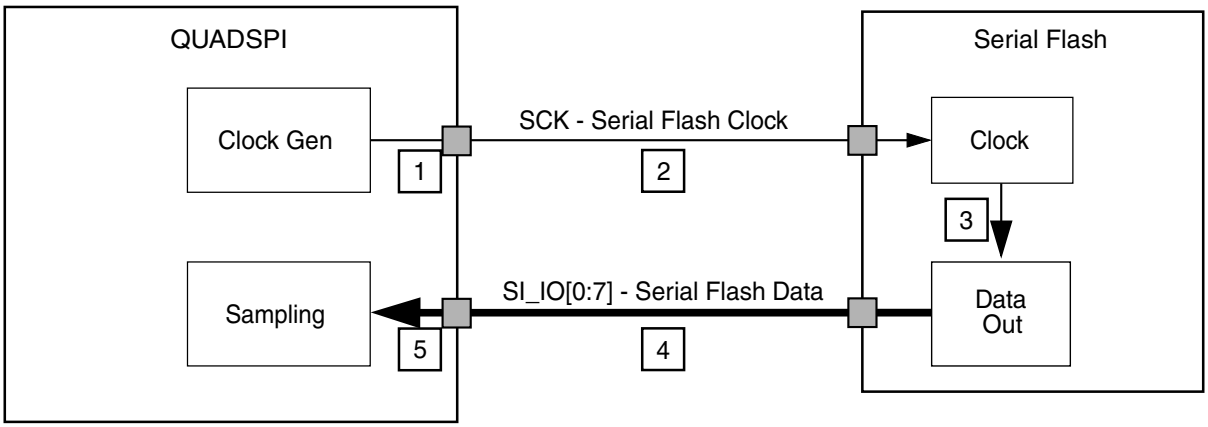


Figure 35-11. Serial Flash Sampling Clock Overview

The rising edge of the internal reference clock is taken as timing reference for the data output of the serial flash. After a time of $t_{Del,total}$ the data arrives at the internal sampling stage of the QuadSPI module. According to the Serial Flash Sampling Clock Overview figure, the following parts of the delay chain contribute to $t_{Del,total}$:

1. Output delay of the serial flash clock output of the device containing the QuadSPI module
2. Wire delay of application/PCB from the device containing the QuadSPI module to the external serial flash device
3. Clock to data out delay of the external serial flash device, including input and output delays
4. Wire delay of application/PCB from the external serial flash device to the device containing the QuadSPI module
5. Device delay corresponding to the input data

NOTE

The amount of total delay $t_{Del,total}$ is specific to the characteristics of the actual implementation. Also, the serial flash device clock (SCK) is inverted with respect to the QuadSPI internal reference clock.

35.10.2 Supported read modes

The table below provides an overview of QuadSPI read modes.

Table 35-53. QuadSPI read modes

Read modes			QuadSPI_MCR[DDR_EN]	QuadSPI_MCR[DQS_EN]	Data learning support	For more information
SDR mode	Internal sampling (N/1, I/1)		0	0	No	Internal sampling
	DQS sampling method	Internally generated DQS	0	1	No	Internally generated DQS
		External DQS	0	1	No	External DQS
DDR mode	Internal sampling (4x method)		1	0	Yes	Internal sampling (4x sampling method)
	DQS sampling method	Internally generated DQS	1	1	Yes	Internally generated DQS
		External DQS	1	1	Yes	External DQS

35.10.2.1 SDR mode

Most flash memories operate in single data rate (SDR) mode. In SDR mode, the data is transferred only on one edge of the clock signal. The SDR serial flash memories sample the incoming data on the rising edge of serial flash clock and drive the output data on the falling edge of the serial flash clock.

35.10.2.1.1 Internal sampling

QuadSPI uses different edges of the internal reference clock for sampling the input data in SDR mode.

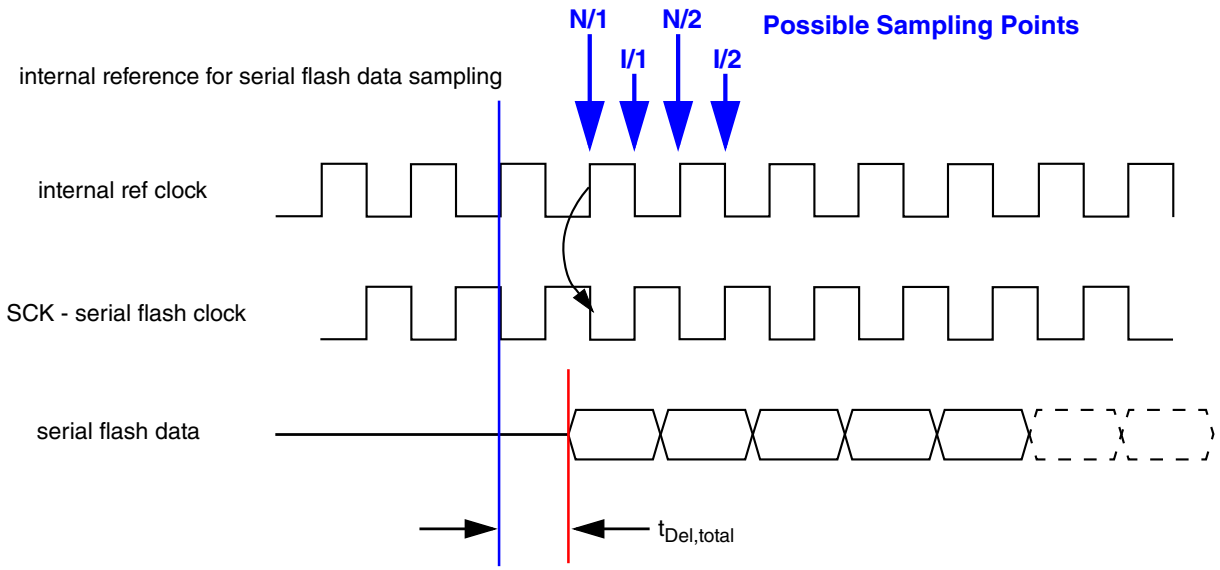


Figure 35-12. Internal sampling in SDR mode

The possible points in time for sampling incoming data are denoted as N/1, I/1, N/2 and I/2 above. The sampling point relevant for the internal sampling is configured in the QSPI_SMPR register. Refer to Sampling Register (QuadSPI_SMPR) for details. The following table gives an overview of the available configurations for the commands running at regular (full) speed:

Table 35-54. Sampling Configuration

Sampling Point	Description	Delay [FSDLY] [HSDLY]	Phase [FSPHS] [HSPHS]	QSPI_SMPR for Full Speed Setting[1]
N/1	sampling with non-inverted clock, 1 sample delay	0	0	0x0000000x
I/1	sampling with inverted clock, 1 sample delay	0	1	0x0000002x
N/2	sampling with non-inverted clock, 2 samples delay	1	0	0x0000004x
I/2	sampling with inverted clock, 2 samples delay	1	1	0x0000006x

1. 'x' is not considered here

Depending on the actual delay and the serial flash clock frequency, the appropriate sampling point can be chosen. The following remarks should be considered when selecting the appropriate setting:

- Theoretically there should be two settings possible to capture the correct data, since the serial flash output is valid for 1 clock cycle, disregarding rise and fall times and timing uncertainties.

- Depending on the timing uncertainties, it may turn out in actual applications that only one possible sample position remains. This is subject to careful consideration depending on the actual implementation.
- The delay $t_{Del,total}$ is an absolute size to shift the point in time when the serial flash data get valid at the QuadSPI input.
- For decreasing frequency of the serial flash clock the distance between the edges increases. So for large differences in the frequency the required setting may change.
- For commands running at half of the regular serial flash clock (QSPI_SMPR[HSENA] bit set) the sampling point must be figured separately to allow for the compensation of the absolute shift in time with respect to the sample-relative setting in the QSPI_SPMPR register.

35.10.2.1.2 DQS sampling method

Data sampling in SDR mode can be supported using the DQS sampling method. Refer to Data Strobe (DQS) sampling method for more details.

35.10.2.2 DDR Mode

The increasing requirement of improved throughput has introduced the double data rate (DDR) mode. In DDR mode, the data is transferred on both the rising and falling edges of the serial flash clock. The DDR serial flashes sample as well as drive the data on both rising and falling edges of serial flash clock.

35.10.2.2.1 Internal sampling (4x sampling method)

When the serial flash memories function in DDR mode, the data is valid for only half a clock cycle. This, along with the fact that the time for which the data is actually valid is smaller than half a clock cycle, requires that we provide closely spaced sampling points. The QuadSPI module provides a mechanism to sample the incoming data at multiple sampling points provided by a 4x serial flash clock in DDR mode. The figure below shows the different sampling points as configured by QuadSPI_SMPR[DDRSMP]. The FSDLY/FSPHS and HSDLY/HSPHS bits are ignored for DDR instructions.

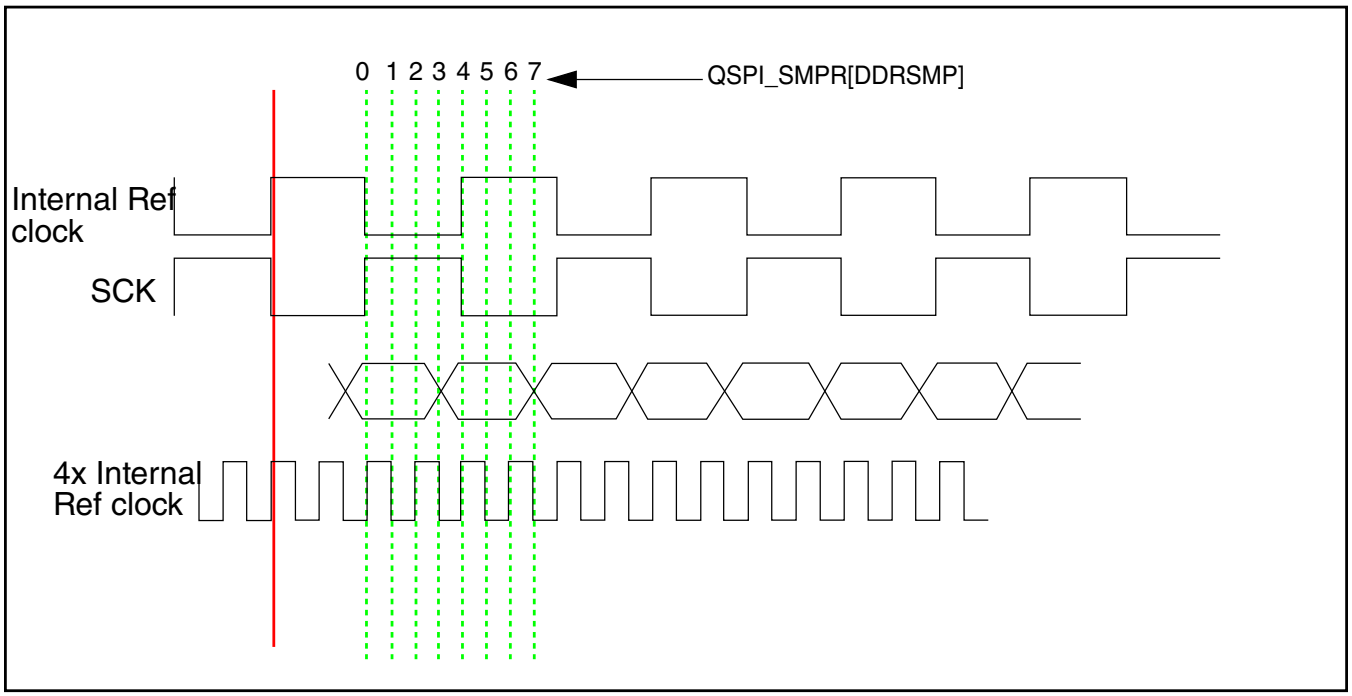


Figure 35-13. 4x sampling edges in DDR mode

Software should ensure that the correct sampling value is configured in the QuadSPI_SMPR[DDRSMP] register.

NOTE

Higher frequency can be achieved using data learning. For details, refer to Data Learning.

35.10.2.2.2 DQS sampling method

Data sampling in DDR mode can be supported using DQS method. Refer to Data Strobe (DQS) sampling method for more details.

A higher frequency can be achieved using Data learning. For details refer to Data Learning.

35.10.3 Data Strobe (DQS) sampling method

35.10.3.1 Basic Description

In DQS mode, the data strobe signal (DQS/RWDS) is used to sample the read data. Here, both DQS and the data sent by the flash move in the same direction, so it is relatively easier to achieve at higher frequencies.

When using DQS for SDR reads, QuadSPI internally samples the incoming data on only one of the edges (either rising edge or falling edge) of the strobe signal. The edge on which the data will be sampled depends on the setting of QuadSPI_SMPR[FSPHS].

QSPI_SMPR[FSPHS]	Sampling edge
0	rising
1	falling

The figure below shows sampling read data in SDR mode using DQS.

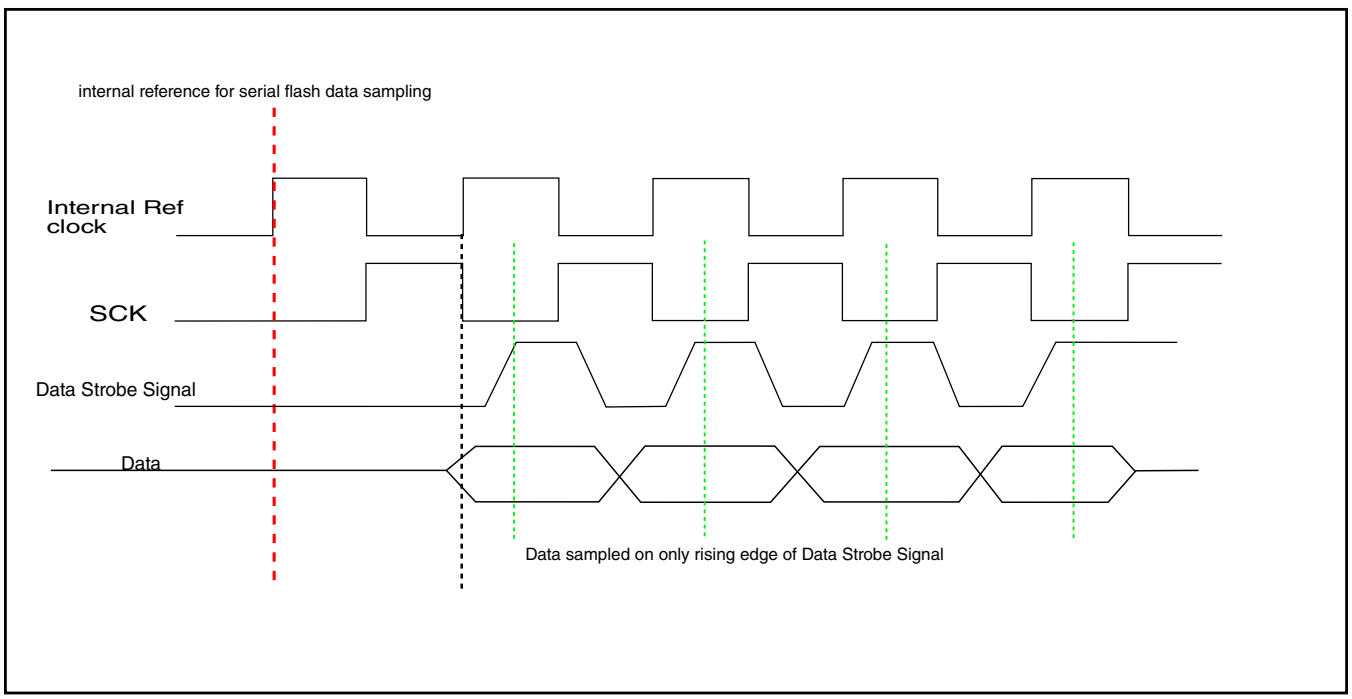


Figure 35-14. Data Strobe functionality in SDR mode

When using DQS for DDR reads, QuadSPI internally samples the incoming data on both the edges of the strobe signal. Refer to the figure below for more detail.

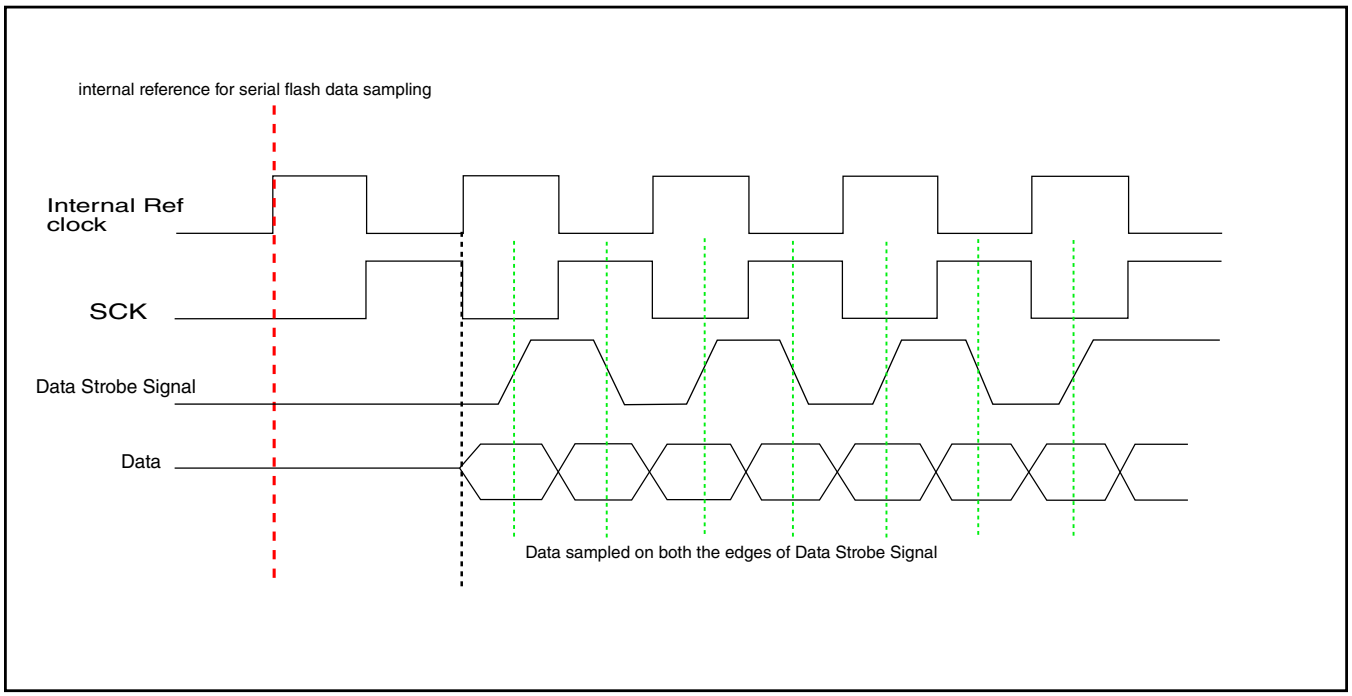


Figure 35-15. Data Strobe functionality in DDR mode

35.10.3.2 Internally generated DQS

In this mode, the internal reference clock is fed as a strobe to QuadSPI for read data sampling. The data strobe must be generated in a way such that the data is correctly sampled by the QuadSPI module. This internally generated data strobe signal can be used by QuadSPI to capture the data in:

- SDR mode
- DDR mode

Refer to QuadSPI chip-specific information for additional details about internally generated DQS.

35.10.3.3 External DQS

In serial flash memories supporting DQS, the data strobe signal is an output from the flash device that indicates when data is being transferred from the flash to the host controller. The data is then captured by the controller on:

- Only one edge (either rising or falling edge) of DQS signal in SDR mode
- Both rising/falling edge of the DQS signal in DDR mode

Some serial flash memories (for example, HyperFlash) provide the data strobe output (RWDS) with latency cycles included in between the ongoing read transaction. The data is sampled on both the edges of this signal taking into consideration that when no signal is provided by flash between ongoing transaction, data is not sampled then and at the end of transaction the required number of data to be fetched remains the same. Refer to the figure below for more detail.

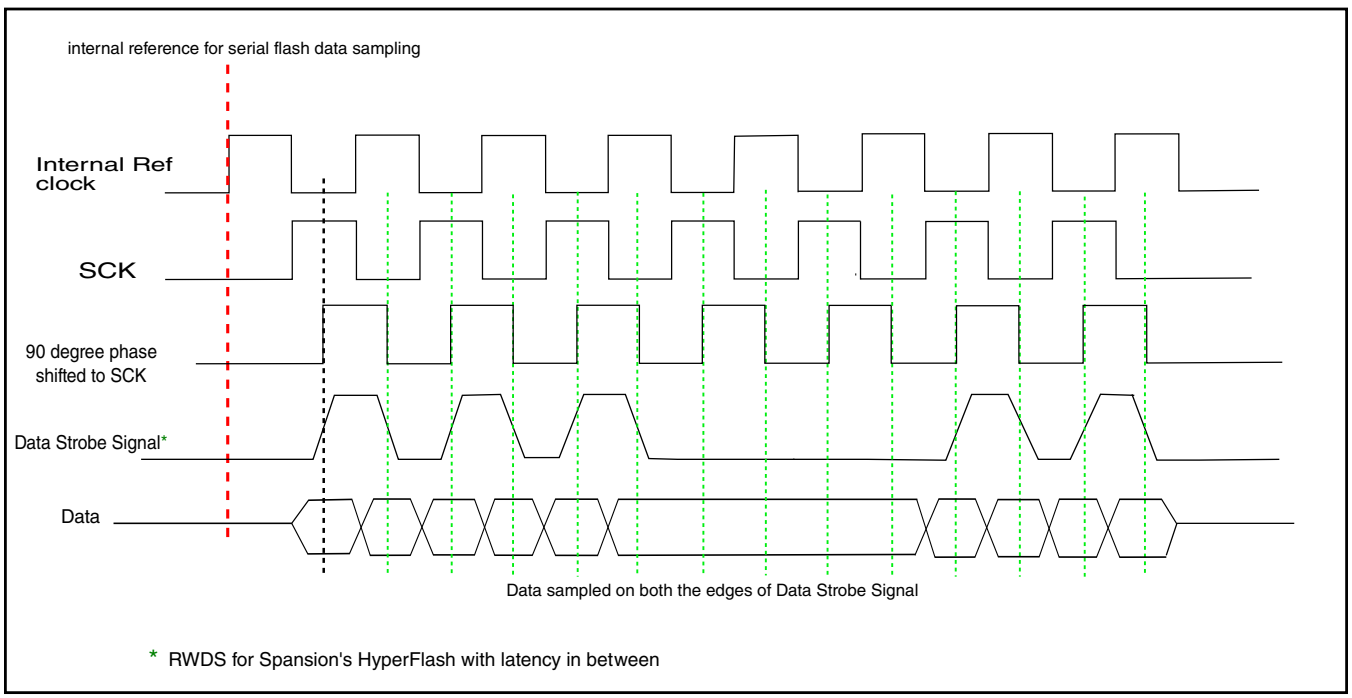


Figure 35-16. Data Strobe functionality with latency included

35.10.3.3.1 External Center Aligned Read Strobe

The QuadSPI supports serial flash memories using the Center Aligned Read Strobe (CRS) protocol. By virtue of this protocol, in addition to the SCK, the host controller provides a 90^{o} phase shifted SCK that allows the flash memory to generate valid data timed off SCK edges and a Data Read Strobe (DQS) signal output that is timed off the 90^{o} phase shifted SCK edges that is centered within the valid read data window. This allows the host controller to easily capture read data from the flash by using the data read strobe transitions and thereby allows for higher performance operation. Since the flash memory is the originator of the center aligned data read strobe as well as the read data, the temporal timing phase alignment between these signals is easier to guarantee across PVT (Process, Voltage and Temperature) corners.

The figure below shows data strobe functionality using CRS protocol.

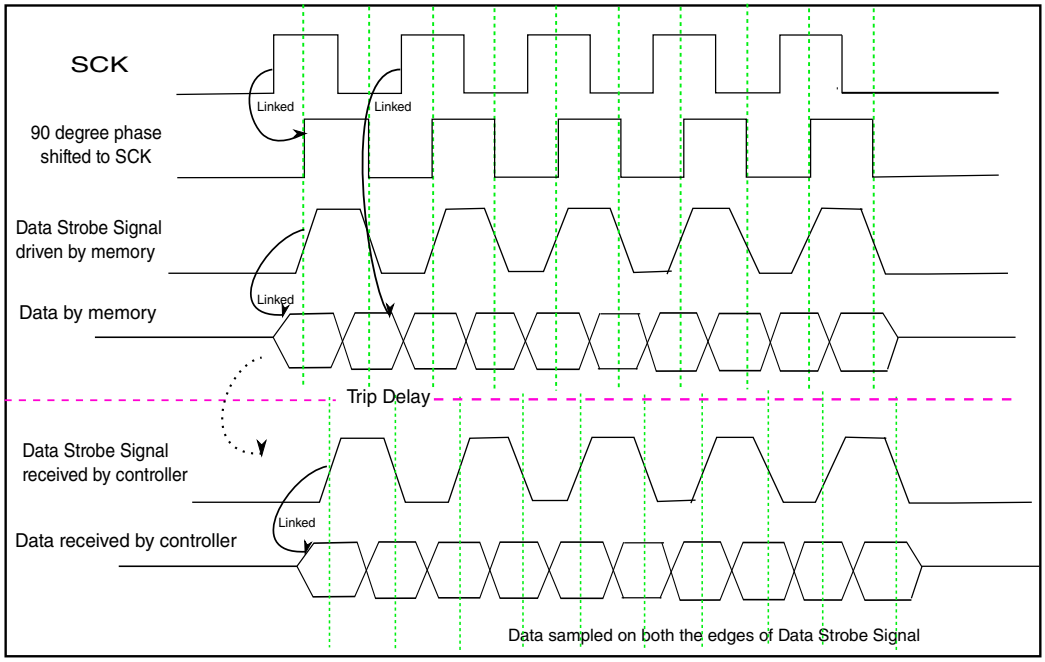


Figure 35-17. Data Strobe functionality using CRS protocol

35.10.4 Data Learning

35.10.4.1 Basic description

“Data learning” is used to manage varying data valid windows from flash memory as well as any variations in the chip based on PVT conditions in DDR mode. QuadSPI provides this feature via the DATA_LEARN instruction for all flash memories, irrespective of whether the flash supports it. The DATA_LEARN instruction accepts an operand that defines the number of bits of the known pattern for which the data learning has to be done. The known pattern is provided in the QSPI_DLPR register.

QuadSPI supports the data learning methods below:

- 4x sampling method
- Data Strobe (DQS) sampling method

The following sections explain data learning in greater detail.

35.10.4.2 4x sampling method

Automatic data learning is supported in 4x sampling mode. To start data learning:

1. Certain flash memories provide data learning as a feature for DDR data reads.

   For flash memories that support data learning, configure a known pattern in the Data Learn register inside flash. The pattern should be chosen to have multiple low and high transitions in the data bus. Set up a QuadSPI DDR data read sequence, inserting a DATA_LEARN instruction before the READ_DDR instruction. The flash will itself return the data learning pattern in between dummy cycles in every read command. Each I/O will output the same DLP value for every clock edge.

   For flash memories that *do not* support data learning, configure a known location in flash memory with a known pattern. The pattern should not have 0x00 or 0xFF in it. The pattern should be chosen to have multiple low and high transitions in the data bus. Set up a QuadSPI DDR data read sequence, inserting a DATA_LEARN instruction before the READ_DDR instruction. The address in the QuadSPI_SFAR register should point to the known location with the pattern. Configure the known pattern in the QuadSPI_DLPR [DLPV] field. Refer to Table 35-55 for details on how the data has to be ordered in memory for correct operation.

2. Select a sampling point using QuadSPI_SMPR[DDRSMP].
3. Initiate the read via a peripheral transaction.
4. QuadSPI reads the data from the flash. It then encounters the DATA_LEARN instruction and samples the incoming data on all 8 possible sampling points for all the data bus bits.
5. The sample point at which data matches the known pattern is reported in QuadSPI_SR[DLPSMP].
6. If data doesn't match at any sample point, QuadSPI_FR[DLPFF] is set.
7. If the correct sampling point is located, QuadSPI uses the new sampling point to sample the read data following the DATA_LEARN instruction.

This feature may be used to auto-calibrate the sampling point for DDR reads. This auto-calibration may be triggered at fixed intervals, or depending on a change in PVT conditions.

The following figure shows data learning with 4x DDR sampling:

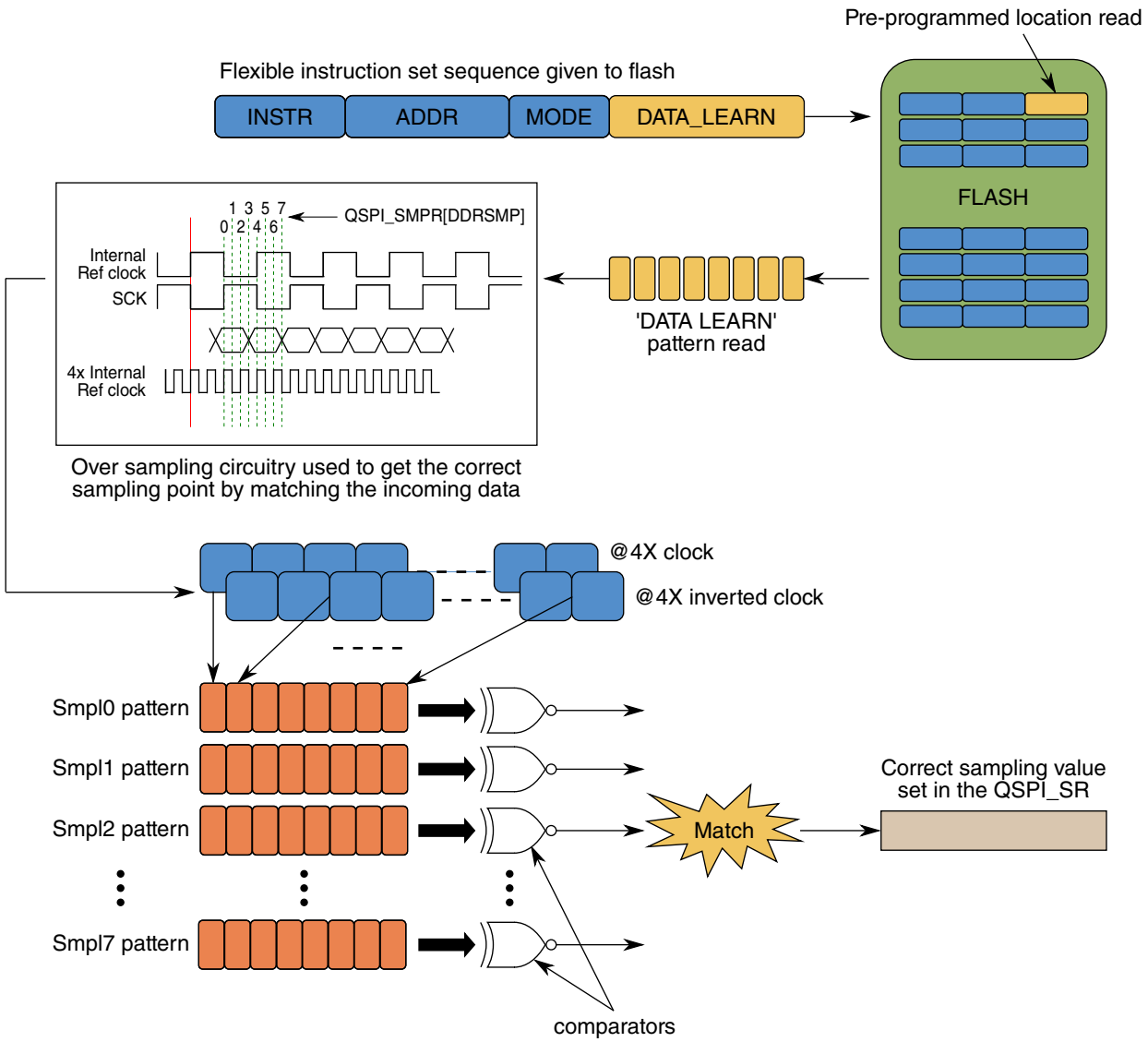


Figure 35-18. Data learning with 4x sampling method

35.10.4.3 Data Strobe (DQS) sampling method

Semi-automatic data learning is supported in DQS sampling method. To start data learning:

1. Certain flash memories provide data learning as a feature for DDR data reads.

   For flash memories that support data learning, configure a known pattern in the Data Learn register inside flash. The pattern should be chosen to have multiple low and high transitions in the data bus. Set up a QuadSPI DDR data read sequence, inserting a DATA_LEARN instruction before the READ_DDR instruction. The flash will itself return the data learning pattern in between dummy cycles in every read command. Each I/O will output the same DLP value for every clock edge.

For flash memories that *do not* support data learning, configure a known location in flash memory with a known pattern. The pattern should not have 0x00 or 0xFF in it. The pattern should be chosen to have multiple low and high transitions in the data bus. Set up a QuadSPI DDR data read sequence, inserting a DATA_LEARN instruction before the READ_DDR instruction. The address in the QSPI_SFAR register should point to the known location with the pattern. Configure the known pattern in the QSPI_DLPR[DLPV] field. Refer to Table 35-55 for details on how the data has to be ordered in memory for correct operation.

2. Select a sampling point. See chip-specific QuadSPI information for selection of the sampling point.
3. Initiate the read via a peripheral transaction.
4. QuadSPI reads the data from the flash. It then encounters the DATA_LEARN instruction and samples the incoming data on both edges (rising and falling) of the DQS.
5. If the data from the flash doesn't match the data learning pattern, the QuadSPI_FR[DLPFF] flag is set.
6. No sampling point is reported automatically by the QuadSPI module.
7. These steps (2-7) are repeated with varying VT conditions for a particular process until the QuadSPI_FR[DLPFF] is set. The sampling points where no QuadSPI_FR[DLPFF] is set signify valid setting.
8. In the case of multiple valid settings, software should choose the middle point.
9. The sampling point should be fixed with the above chosen point for the next READ transactions.

NOTE

It must be ensured that QuadSPI is not accessed during calibration.

NOTE

Ensure that there is no latency in between DQS edges while doing data learning using DQS sampling method.

This feature may be used to auto-calibrate the sampling point for DDR reads. This auto-calibration may be triggered at fixed intervals, or depending on a change in PVT conditions.

The following figure shows data learning in DQS mode:

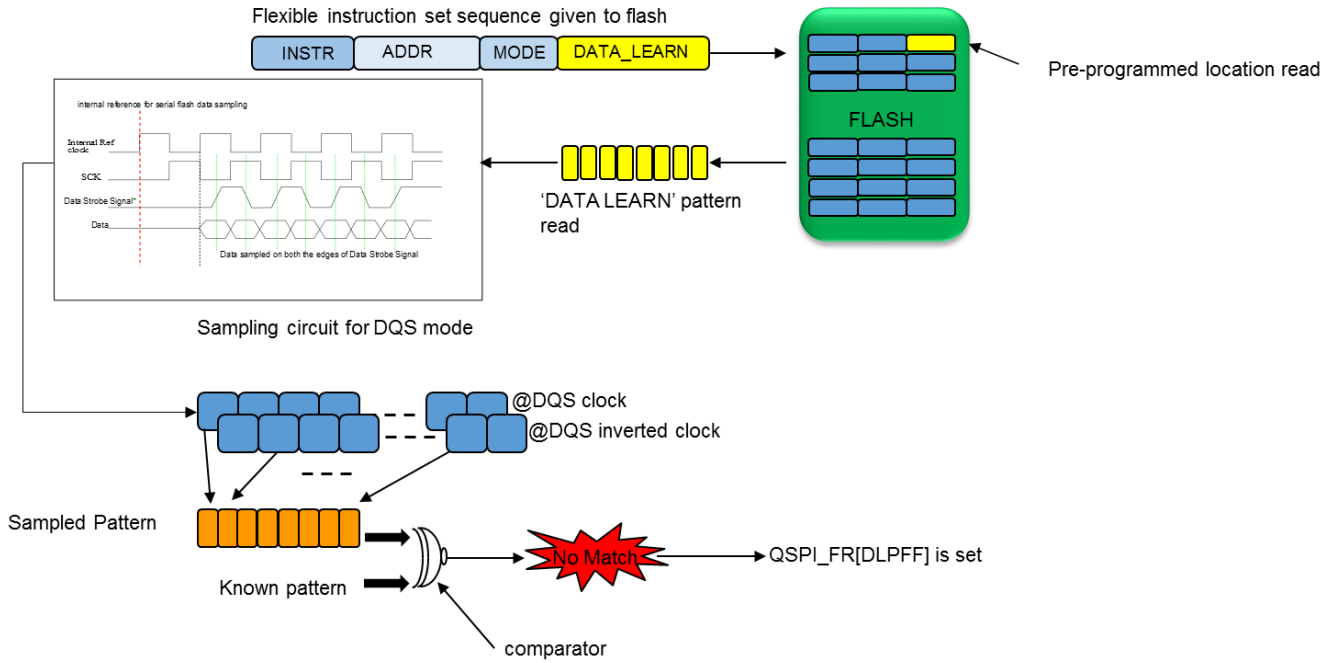


Figure 35-19. Data learning with DQS sampling method

35.10.4.4 Example: programming the data learning pattern in flash memory

If QSPI_DLPR[DLPV] is set to 0x43, QuadSPI tries to match this sample on every incoming data line for single, dual, quad, or octal configurations. The table below shows the data programmed in flash with endianness taken into consideration.

Table 35-55. Programming the Data Learning Pattern in flash memory

Endianness	Single IO	Dual IO	Quad IO	Octal IO
BE	0x43	0x300F	0x0F0000FF	0x00FF00000000FFFF
LE	0x43	0X0F30	0XFF00000F	0XFFFF0000000000FF

35.11 Data Input Hold Requirement of Flash

In DDR mode, the data is valid only for half clock cycle. It is difficult to meet the data input hold time requirement of flash. QuadSPI internally delays the data sent to flash so that it will be easy to meet the hold requirement. The QSPI_FLSHCR[TDH] is used for this purpose. If QSPI_FLSHCR[TDH] is configured to 0, the data sent to flash is aligned

with the internal reference clock of QuadSPI, if it is 1, then, the data is aligned to 2x internal reference half clock, if it is 2, then, the data is aligned to 4x internal reference half clock.

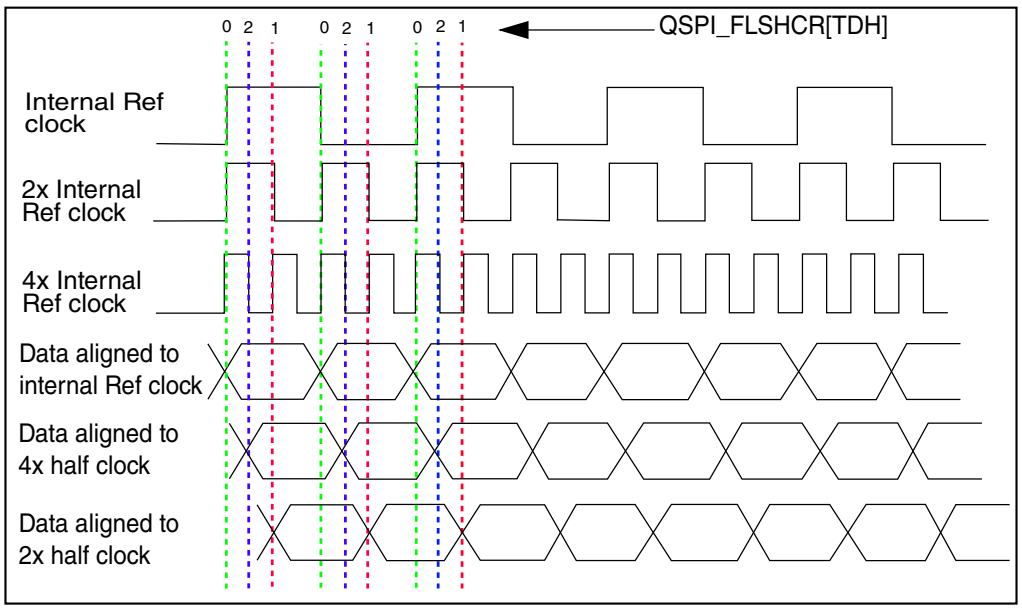


Figure 35-20. Data Hold

Chapter 36
Cyclic Redundancy Check (CRC)

36.1 Introduction

The cyclic redundancy check (CRC) module generates 16/32-bit CRC code for error detection.

The CRC module provides a programmable polynomial and other parameters required to implement a 16-bit or 32-bit CRC standard.

The 16/32-bit code is calculated for 32 bits of data at a time.

36.1.1 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

36.1.2 Block diagram

The following is a block diagram of the CRC.

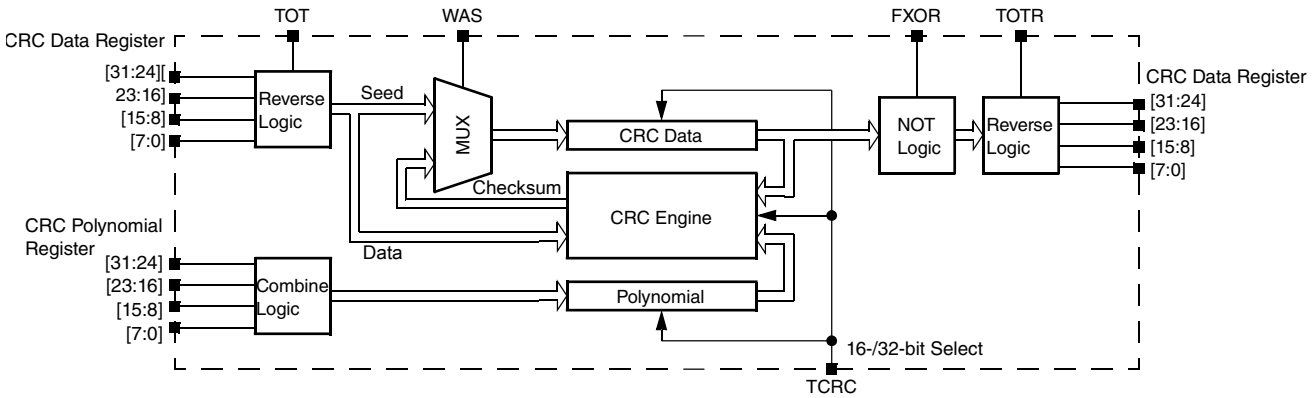


Figure 36-1. Programmable cyclic redundancy check (CRC) block diagram

36.1.3 Modes of operation

Various MCU modes affect the CRC module's functionality.

36.1.3.1 Run mode

This is the basic mode of operation.

36.1.3.2 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 dependent on the MCU.

36.2 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	36.2.1/895
4003_2004	CRC Polynomial register (CRC_GPOLY)	32	R/W	0000_1021h	36.2.2/896
4003_2008	CRC Control register (CRC_CTRL)	32	R/W	0000_0000h	36.2.3/896

36.2.1 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 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

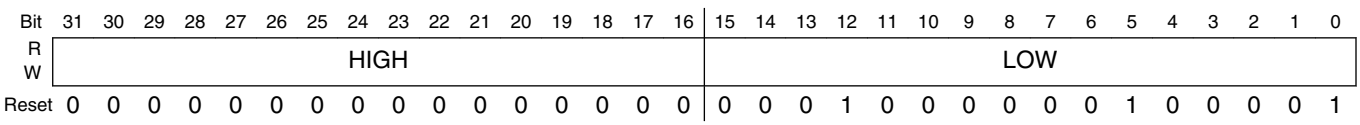
CRC_DATA field descriptions

Field	Description
31–24 HU	CRC High Upper Byte In 16-bit CRC mode (CTRL[TCRC] is 0), this field is not used for programming a seed value. In 32-bit CRC mode (CTRL[TCRC] is 1), values written to this field are part of the seed value when CTRL[WAS] is 1. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation in both 16-bit and 32-bit CRC modes.
23–16 HL	CRC High Lower Byte In 16-bit CRC mode (CTRL[TCRC] is 0), this field is not used for programming a seed value. In 32-bit CRC mode (CTRL[TCRC] is 1), values written to this field are part of the seed value when CTRL[WAS] is 1. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation in both 16-bit and 32-bit CRC modes.
15–8 LU	CRC Low Upper Byte When CTRL[WAS] is 1, values written to this field are part of the seed value. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation.
LL	CRC Low Lower Byte When CTRL[WAS] is 1, values written to this field are part of the seed value. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation.

36.2.2 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



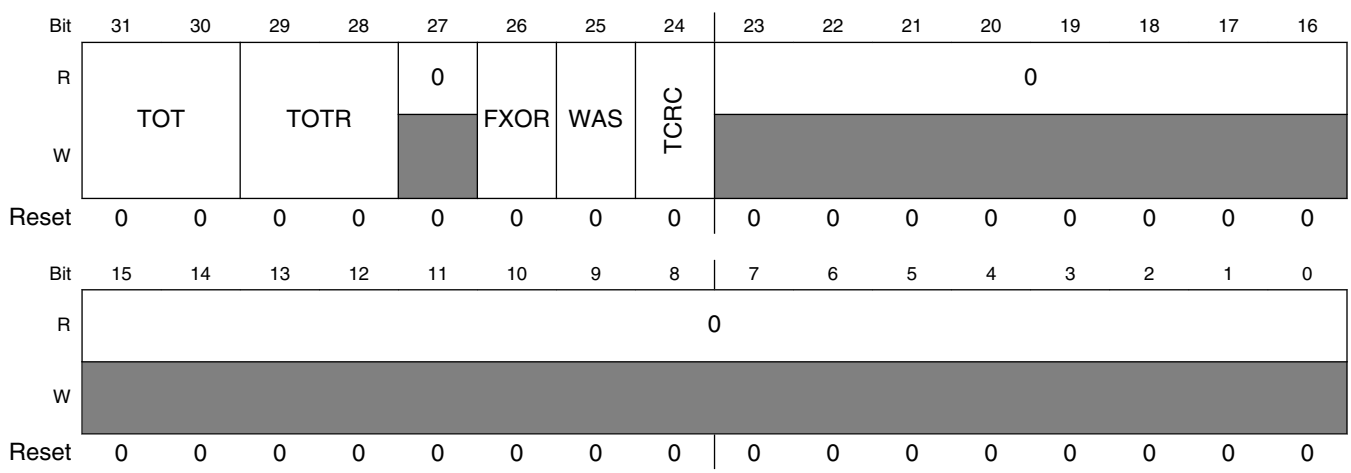
CRC_GPOLY field descriptions

Field	Description
31–16 HIGH	High Polynominal Half-word Writable and readable in 32-bit CRC mode (CTRL[TCRC] is 1). This field is not writable in 16-bit CRC mode (CTRL[TCRC] is 0).
LOW	Low Polynominal Half-word Writable and readable in both 32-bit and 16-bit CRC modes.

36.2.3 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 TOT	Type Of Transpose For Writes Defines the transpose configuration of the data written to the CRC data register. See the description of the transpose feature for the available transpose options. 00 No transposition. 01 Bits in bytes are transposed; bytes are not transposed. 10 Both bits in bytes and bytes are transposed. 11 Only bytes are transposed; no bits in a byte are transposed.
29–28 TOTR	Type Of Transpose For Read Identifies the transpose configuration of the value read from the CRC Data register. See the description of the transpose feature for the available transpose options. 00 No transposition. 01 Bits in bytes are transposed; bytes are not transposed. 10 Both bits in bytes and bytes are transposed. 11 Only bytes are transposed; no bits in a byte are transposed.
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 FXOR	Complement Read Of CRC Data Register Some CRC protocols require the final checksum to be XORed with 0xFFFFFFFF or 0xFFFF. Asserting this bit enables on the fly complementing of read data. 0 No XOR on reading. 1 Invert or complement the read value of the CRC Data register.
25 WAS	Write CRC Data Register As Seed When asserted, a value written to the CRC data register is considered a seed value. When deasserted, a value written to the CRC data register is taken as data for CRC computation. 0 Writes to the CRC data register are data values. 1 Writes to the CRC data register are seed values.
24 TCRC	Width of CRC protocol. 0 16-bit CRC protocol. 1 32-bit CRC protocol.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

36.3 Functional description

36.3.1 CRC initialization/reinitialization

To enable the CRC calculation, the user must program CRC_CTRL[WAS], CRC_GPOLY,necessary parameters for transposition and CRC result inversion in the applicable registers. Asserting CRC_CTRL[WAS] enables the programming of the seed value into the CRC_DATA register.

After a completed CRC calculation, the module can be reinitialized for a new CRC computation by reasserting CRC_CTRL[WAS] and programming a new, or previously used, seed value. All other parameters must be set before programming the seed value and subsequent data values.

36.3.2 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.

36.3.2.1 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.

36.3.2.2 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.

36.3.3 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.

36.3.3.1 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]}

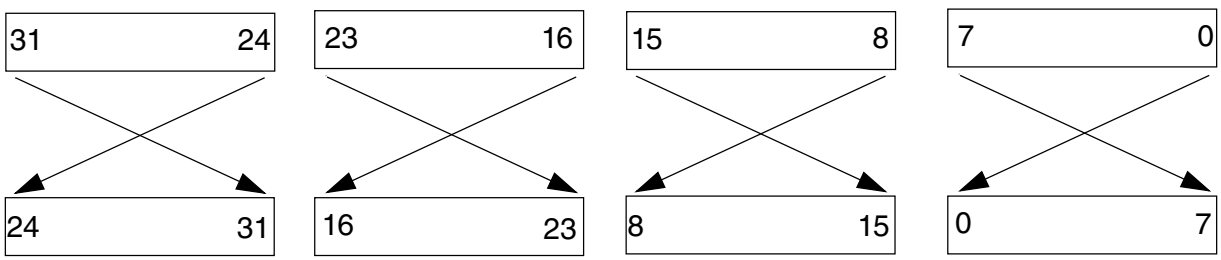


Figure 36-2. Transpose type 01

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]}

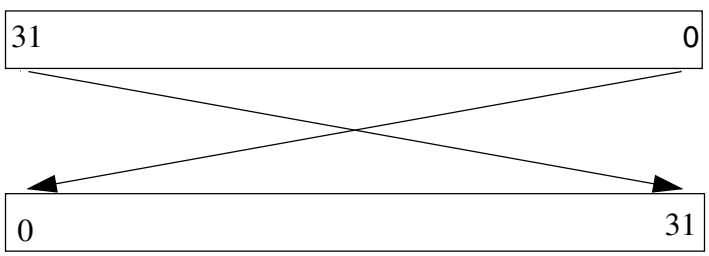


Figure 36-3. Transpose type 10

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]}

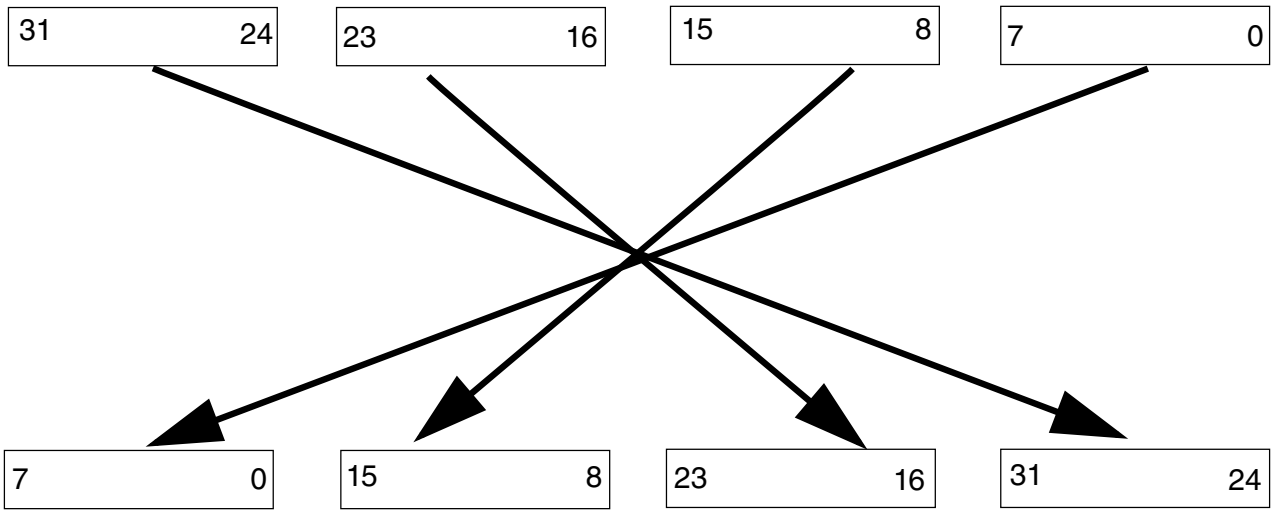


Figure 36-4. Transpose type 11

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.

36.3.4 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 37
True Random Number Generator

37.1 Standalone True Random Number Generator (SA-TRNG).

The Standalone True Random Number Generator (SA-TRNG) is hardware accelerator module that generates a 512-bit entropy as needed by an entropy consuming module or by other post processing functions. A typical entropy consumer is a pseudo random-number generator (PRNG) which can be implemented to achieve both true randomness and cryptographic strength random numbers using the TRNG output as its entropy seed. The PRNG is not part of this module.

The entropy generated by an TRNG is intended for direct use by functions that generate secret keys, per-message secrets, random challenges, and other similar quantities used in cryptographic algorithms. In each of these cases, it is important that a random number be difficult to guess or predict. It is important that a random number is at least as difficult to predict as it is difficult to break the cryptographic algorithm with which it is being used. This stringent requirement is particularly difficult to fulfill if the entropy source from a TRNG contains bias and/or correlation. To increase the trustworthiness/quality of the generated random data, PRNGs are often used to post process the output of a TRNG.

This document describes only the TRNG design functionality and usage.

Note that before entropy can be obtained from the TRNG, it must be initialized and instantiated in a particular mode by setting the appropriate TRNG registers.

The TRNG contains the following sub modules: IP Slave bus (SkyBlue bus) interface, the TRNG Core and the free running oscillator (OSC).

37.1.1 Standalone True Random Number Generator Block Diagram

The following figure is a top-level diagram of the True Random Number Generator.

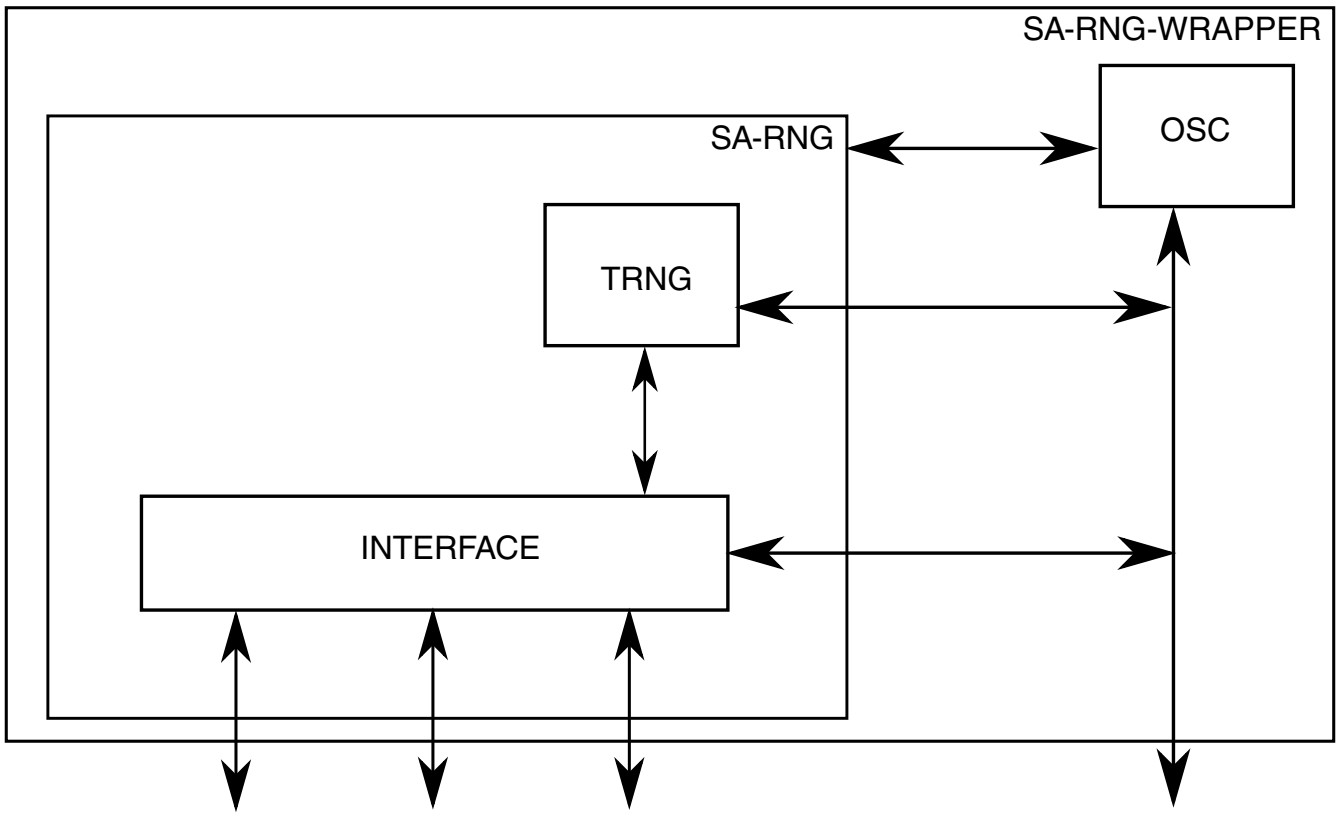


Figure 37-1. SA TRNG Block Diagram

37.1.2 TRNG Functional Description.

The TRNG consists of several functional sub-modules. Its overall functionality can be easily described from the top level in terms of generating entropy for seed generation. The functionality of each sub-module is briefly described in the following subsections.

TRNG is based on collecting bits from a random noise source. This random noise source is a ring oscillator that is sensitive to random noise (temperature variations, voltage variations, cross-talk and other random noise) within the device in which the TRNG is used. This noise causes various small changes in the period of the oscillator. Therefore, if the count of the ring oscillator clock cycles is sampled after a known period of time, this count will vary each time the sample is taken. By using the variance in this count over a large number of samples, random bits can be derived.

The TRNG comprises two entropy sources, each of which provides a single bit of output. Concatenated together, these 2 output bits are expected to provide 1 bit of entropy every 100 clock cycles. In addition to generating entropy, the TRNG also performs several statistical tests on its output.

37.1.3 SA-TRNG hardware functional description.

SA-TRNG functionality consists of several major subcomponents. This table describes these subcomponents.

Table 37-1. SA-TRNG subcomponents.

Description	Cross-reference(s)
Interfaces	
Register interface • Used for access to configuration, control, status and debugging registers	Register interface (IP Slave bus)
True Random Number Generator (TRNG)	Standalone True Random Number Generator (SA-TRNG).

37.1.3.1 Software Use Cases for the Stand Alone TRNG.

There are four things that a user (programmer/integrator) will want to do with a TRNG.

- Initialization.

  Set up the parameters to proper values, and start generation of the first block of entropy. This is done once.

- Read entropy from the TRNG, and start generation of the next block of entropy.

  This is done many times and is the normal flow of operation.

- Run a self-test on the TRNG, to assure proper continued operation.

  This involves taking TRNG off-line, setting some self-test parameters, running TRNG, and then reading the statistical test registers, to see that they are within proper operation values. This may not be needed, as TRNG has built-in self-test.

- Off-line determination and checking of TRNG parameter values.

  This is done in development in order to determine the proper initialization and self-test parameters. The TRNG is taken off-line. Test parameter values are written and entropy generation is started. If the statistical tests indicate poor operation (i.e., failing statistical tests), the entropy_delay value should be increased and entropy generation should be re-started. Every case is a variation of setting TRNG parameter values, starting or re-starting entropy generation and reading out the entropy. This process requires pausing or stopping and re-starting the TRNG.

The TRNG is designed to operate as a slave module on the standard IP Slave Bus. By understanding the TRNG register descriptions in "TRNG Register Descriptions" section below, the TRNG module can be controlled via the IP slave bus. In order to write to most TRNG registers, the MCTL register must be initialized in programming mode as described in the "TRNG Register Descriptions" section. At Power On Reset (POR), the TRNG resets to programming mode. And the it will not generate entropy until it is out of programming mode (in run mode) and access to Entropy Registers have been enabled.

Here is an example program flow of using the TRNG.

- After POR the TRNG will be reset into programming mode with the OK to stop bit set(MCTL[TSTOP_OK]=1). The TRNG must be put into Run Mode for Entropy Generation to begin (MCTL[PRGM]=0). Additionally, in order to have access to the Entropy registers and other critical TRNG registers, the TRNG access bit must be set (MCTL[TRNG_ACC]=1). Using the default self test limits that exist after bootup, the entropy valid bit can be polled until asserted (MCTL[ENT_VAL]=1). Alternatively, if using the interrupt, and the interrupts are enabled via the INT_MASK register and the ipi_rng_int_b is asserted when MCTL[ENT_VAL]=1.
- After the polling completes, the 512-bit entropy generated by the TRNG can be read. The values can be read in any order from entropy register 0 to register 15 (ENT0 to ENT 15). After reading ENT 15, the old entropy value is reset and a new entropy value is generated.

NOTE

Reading ENT 15always resets the entropy, so should always be read last.

- You can poll again for the new entropy value or you can use the Interrupt Status Register to handle reading the entropy values when the entropy valid interrupt is triggered.
- The interrupt can be masked or cleared as needed. See the Interrupt Status Register description.
- To change the self-test limits,the seed counters, how fast the entropy is generated, and how entropy is sampled, see the register description section. In particular, see the the TRNG Frequency Count Minimum Limit Register (FRQMIN), the seed control register(SCML), the statistical run length registers, and other parameter registers.
- Once in Run Mode, the entropy is re-generated automatically after ENT 15 is read. To stop the TRNG or access to TRNG registers at any point while in running mode, you can always set MCTL[TRNG_ACC]=0. Setting the TRNG back to programming mode (MCTL[PRGM]=1) also achieves the purpose of stopping entropy generation.

37.1.3.2 Register interface (IP Slave bus)

The TRNG's register interface (32-bit IP bus) is used to read and write registers within TRNG for the following purposes:

Table 37-2. Summary of register interface uses

Purpose	For more information, see
During chip initialization time	
To configure TRNG including initialization of the • Registers • TRNG Register Interface	
During hardware and software debugging	
Read status registers	• RNG TRNG Status Register • For all registers, see the TRNG Register Descriptions" appendix.

NOTE

Accesses to registers must use full-word (32-bit) reads or writes.

37.1.3.3 TRNG0 Register Descriptions

All accesses of undefined addresses always return zero and assert IPS transfer error. Writes to undefined and read-only addresses are ignored. Undefined addresses are those undocumented, protected or reserved addresses within and outside the range of the addresses defined in the memory map below. Although many of the TRNG0 registers hold more than 32 bits, the register addresses shown in the Memory Map below represent how these registers are accessed over the register bus as 32-bit words.

The format and fields in each TRNG0 register are defined below. Some of the register format figures apply to several different registers. In such cases a different register name will be associated with each of the register offset addresses that appear at the top of the register format figure. Although these registers share the same format, they are independent registers. In addition, many registers can be accessed at multiple addresses. In these cases there will be a single register name and the list of addresses at which that register is accessible will be indicated as aliases. Unless noted in the individual register descriptions, registers are reset only at Power-On Reset (POR).

37.1.3.3.1 TRNG0 Memory Map

Offset	Register	Width (In bits)	Access	Reset value
0h	TRNG0 Miscellaneous Control (TRNG0_MCTL)	32	RW	00012001h
4h	TRNG0 Statistical Check Miscellaneous (TRNG0_SCMISC)	32	RW	00010022h[1]
8h	TRNG0 Poker Range (TRNG0_PKRRNG)	32	RW	000009A3h
Ch	TRNG0 Poker Maximum Limit (TRNG0_PKRMAX)	32	RW	00006920h
Ch	TRNG0 Poker Square Calculation Result (TRNG0_PKRSQ)	32	RO	00000000h
10h	TRNG0 Seed Control (TRNG0_SDCTL)	32	RW	0C8009C4h
14h	TRNG0 Sparse Bit Limit (TRNG0_SBLIM)	32	RW	0000003Fh
14h	TRNG0 Total Samples (TRNG0_TOTSAM)	32	RO	00000000h
18h	TRNG0 Frequency Count Minimum Limit (TRNG0_FRQMIN)	32	RW	00000640h
1Ch	TRNG0 Frequency Count Maximum Limit (TRNG0_FRQMAX)	32	RW	00006400h
1Ch	TRNG0 Frequency Count (TRNG0_FRQCNT)	32	RO	00000000h
20h	TRNG0 Statistical Check Monobit Count (TRNG0_SCMC)	32	RO	00000000h
20h	TRNG0 Statistical Check Monobit Limit (TRNG0_SCML)	32	RW	010C0568h
24h	TRNG0 Statistical Check Run Length 1 Limit (TRNG0_SCR1L)	32	RW	00B20195h
24h	TRNG0 Statistical Check Run Length 1 Count (TRNG0_SCR1C)	32	RO	00000000h
28h	TRNG0 Statistical Check Run Length 2 Limit (TRNG0_SCR2L)	32	RW	007A00DCh
28h	TRNG0 Statistical Check Run Length 2 Count (TRNG0_SCR2C)	32	RO	00000000h
2Ch	TRNG0 Statistical Check Run Length 3 Count (TRNG0_SCR3C)	32	RO	00000000h
2Ch	TRNG0 Statistical Check Run Length 3 Limit (TRNG0_SCR3L)	32	RW	0058007Dh
30h	TRNG0 Statistical Check Run Length 4 Count (TRNG0_SCR4C)	32	RO	00000000h
30h	TRNG0 Statistical Check Run Length 4 Limit (TRNG0_SCR4L)	32	RW	0040004Bh
34h	TRNG0 Statistical Check Run Length 5 Limit (TRNG0_SCR5L)	32	RW	002E002Fh
34h	TRNG0 Statistical Check Run Length 5 Count (TRNG0_SCR5C)	32	RO	00000000h
38h	TRNG0 Statistical Check Run Length 6+ Limit (TRNG0_SCR6PL)	32	RW	002E002Fh
38h	TRNG0 Statistical Check Run Length 6+ Count (TRNG0_SCR6PC)	32	RO	00000000h
3Ch	TRNG0 Status (TRNG0_STATUS)	32	RO	00000000h
40h - 7Ch	TRNG0 Entropy Read (TRNG0_ENT0 - TRNG0_ENT15)	32	RO	00000000h
80h	TRNG0 Statistical Check Poker Count 1 and 0 (TRNG0_PKRCNT10)	32	RO	00000000h
84h	TRNG0 Statistical Check Poker Count 3 and 2 (TRNG0_PKRCNT32)	32	RO	00000000h
88h	TRNG0 Statistical Check Poker Count 5 and 4 (TRNG0_PKRCNT54)	32	RO	00000000h
8Ch	TRNG0 Statistical Check Poker Count 7 and 6 (TRNG0_PKRCNT76)	32	RO	00000000h
90h	TRNG0 Statistical Check Poker Count 9 and 8 (TRNG0_PKRCNT98)	32	RO	00000000h
94h	TRNG0 Statistical Check Poker Count B and A (TRNG0_PKRCNT BA)	32	RO	00000000h
98h	TRNG0 Statistical Check Poker Count D and C (TRNG0_PKRCNT DC)	32	RO	00000000h
9Ch	TRNG0 Statistical Check Poker Count F and E (TRNG0_PKRCNT FE)	32	RO	00000000h
A0h	TRNG0 Security Configuration (TRNG0_SEC_CFG)	32	RW	00000000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
A4h	TRNG0 Interrupt Control (TRNG0_INT_CTRL)	32	RW	FFFFFFFFh
A8h	TRNG0 Mask (TRNG0_INT_MASK)	32	RW	00000000h
ACh	TRNG0 Interrupt Status (TRNG0_INT_STATUS)	32	RW	00000000h
F0h	TRNG0 Version ID (MS) (TRNG0_VID1)	32	RO	00300100h
F4h	TRNG0 Version ID (LS) (TRNG0_VID2)	32	RO	00000000h

1. Reset occurs at POR, and when TRNG0_MCTL[RST_DEF] is written to 1.

37.1.3.3.2 TRNG0 Miscellaneous Control (TRNG0_MCTL)

37.1.3.3.2.1 Address

Register	Offset
TRNG0_MCTL	0h

37.1.3.3.2.2 Function

This register is intended to be used for programming, configuring and testing the RNG. It is the main register to read/write, in order to enable Entropy generation, to stop entropy generation and to block access to entropy registers. This is done via the special TRNG_ACC and PRGM bits below.

The TRNG0 Miscellaneous Control Register is a read/write register used to control the RNG's True Random Number Generator (TRNG) access, operation and test.

NOTE

Note that in many cases two RNG registers share the same address, and a particular register at the shared address is selected based upon the value in the PRGM field of the TRNG0_MCTL register.

37.1.3.3.2.3 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		TSTOP_OK	ERR	TST_OUT	ENT_VAL	FCT_VAL	FCT_FAIL	FOR_SCLK		TRNG_ACC	UNUSED	OSC_DIV		SAMP_MODE	
W				w1c						RST_DEF						
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1

37.1.3.3.2.4 Fields

Field	Function
31-17 —	Reserved.
16 PRGM	Programming Mode Select. When this bit is 1, the TRNG is in Program Mode, otherwise it is in Run Mode. No Entropy value will be generated while the TRNG is in Program Mode. Note that different RNG registers are accessible at the same address depending on whether PRGM is set to 1 or 0. This is noted in the RNG register descriptions.
15-14 —	Reserved.
13 TSTOP_OK	TRNG_OK_TO_STOP. Software should check that this bit is a 1 before transitioning TRNG0 to low power mode (TRNG0 clock stopped). TRNG0 turns on the TRNG free-running ring oscillator whenever new entropy is being generated and turns off the ring oscillator when entropy generation is complete. If the TRNG0 clock is stopped while the TRNG ring oscillator is running, the oscillator will continue running even though the TRNG0 clock is stopped. TSTOP_OK is asserted when the TRNG ring oscillator is not running. and therefore it is ok to stop the TRNG0 clock.
12 ERR	Read: Error status. 1 = error detected. 0 = no error. Write: Write 1 to clear errors. Writing 0 has no effect.
11 TST_OUT	Read only: Test point inside ring oscillator.
10 ENT_VAL	Read only: Entropy Valid. Will assert only if TRNG ACC bit is set, and then after an entropy value is generated. Will be cleared at most one (1) bus clock cycle after reading the TRNG0_ENT15 register. (TRNG0_ENT0 through TRNG0_ENT14 should be read before reading TRNG0_ENT15).
9 FCT_VAL	Read only: Frequency Count Valid. Indicates that a valid frequency count may be read from TRNG0_FRQCNT.
8 FCT_FAIL	Read only: Frequency Count Fail. The frequency counter has detected a failure. This may be due to improper programming of the TRNG0_FRQMAX and/or TRNG0_FRQMIN registers, or a hardware failure in the ring oscillator. This error may be cleared by writing a 1 to the ERR bit.

Table continues on the next page...

Field	Function
7 FOR_SCLK	Force System Clock. If set, the system clock is used to operate the TRNG, instead of the ring oscillator. This is for test use only, and indeterminate results may occur. This bit is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously to writing this bit. This bit is cleared by writing the RST_DEF bit to 1.
6 RST_DEF	Reset Defaults. Writing a 1 to this bit clears various TRNG registers, and bits within registers, to their default state. This bit is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously to writing this bit. Reading this bit always produces a 0.
5 TRNG_ACC	TRNG Access Mode. If this bit is set to 1, the TRNG will generate an Entropy value that can be read via the TRNG0_ENT0-TRNG0_ENT15 registers. The Entropy value may be read once the ENT VAL bit is asserted. Also see TRNG0_ENTa register descriptions (For a = 0 to 15).
4 UNUSED	This bit is unused but write-able. Must be left as zero.
3-2 OSC_DIV	Oscillator Divide. Determines the amount of dividing done to the ring oscillator before it is used by the TRNG. This field is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously to writing this field. This field is cleared to the default POR value by writing the RST_DEF bit to 1. 00 - use ring oscillator with no divide 01 - use ring oscillator divided-by-2 10 - use ring oscillator divided-by-4 11 - use ring oscillator divided-by-8
1-0 SAMP_MODE	Sample Mode. Determines the method of sampling the ring oscillator while generating the Entropy value: This field is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously with writing this field. This field is cleared to the POR default value by writing the RST_DEF bit to 1. 00 - use Von Neumann data into both Entropy shifter and Statistical Checker 01 - use raw data into both Entropy shifter and Statistical Checker 10 - use Von Neumann data into Entropy shifter. Use raw data into Statistical Checker 11 - undefined/reserved.

37.1.3.3.3 TRNG0 Statistical Check Miscellaneous (TRNG0_SCMISC)

37.1.3.3.3.1 Address

Register	Offset
TRNG0_SCMISC	4h

37.1.3.3.3.2 Function

The TRNG0 Statistical Check Miscellaneous Register contains the Long Run Maximum Limit value and the Retry Count value. This register is accessible only when the TRNG0_MCTL[PRGM] bit is 1, otherwise this register will read zeroes, and cannot be written.

37.1.3.3.3.3 Diagram

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

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

1. Reset occurs at POR, and when TRNG0_MCTL[RST_DEF] is written to 1.

37.1.3.3.3.4 Fields

Field	Function
31-20 —	Reserved.
19-16 RTY_CT	RETRY COUNT. If a statistical check fails during the TRNG Entropy Generation, the RTY_CT value indicates the number of times a retry should occur before generating an error. This field is writable only if TRNG0_MCTL[PRGM] bit is 1. This field will read zeroes if TRNG0_MCTL[PRGM] = 0. This field is cleared to the default POR value by writing the TRNG0_MCTL[RST_DEF] bit to 1.
15-8 —	Reserved.
7-0 LRUN_MAX	LONG RUN MAX LIMIT. This value is the largest allowable number of consecutive samples of all 1, or all 0, that is allowed during the Entropy generation. This field is writable only if TRNG0_MCTL[PRGM] bit is 1. This field will read zeroes if TRNG0_MCTL[PRGM] = 0. This field is cleared to the POR reset value by writing the TRNG0_MCTL[RST_DEF] bit to 1.

37.1.3.3.4 TRNG0 Poker Range (TRNG0_PKRRNG)

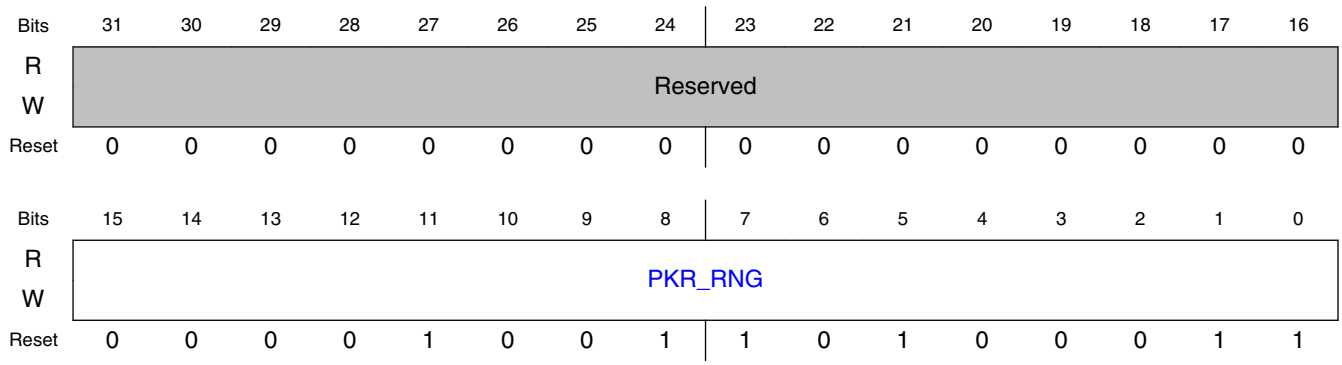
37.1.3.3.4.1 Address

Register	Offset
TRNG0_PKRRNG	8h

37.1.3.3.4.2 Function

The TRNG0 Poker Range Register defines the difference between the TRNG Poker Maximum Limit and the minimum limit. These limits are used during the TRNG Statistical Check Poker Test.

37.1.3.3.4.3 Diagram



37.1.3.3.4.4 Fields

Field	Function
31-16 —	Reserved. Always 0.
15-0 PKR_RNG	Poker Range. During the TRNG Statistical Checks, a "Poker Test" is run which requires a maximum and minimum limit. The maximum is programmed in the PKRMAX[PKR_MAX] register, and the minimum is derived by subtracting the PKR_RNG value from the programmed maximum value. This field is writable only if TRNG0_MCTL[PRGM] bit is 1. This field will read zeroes if TRNG0_MCTL[PRGM] = 0. This field is cleared to the default POR value by writing the TRNG0_MCTL[RST_DEF] bit to 1. Note that the minimum allowable Poker result is PKR_MAX - PKR_RNG + 1.

37.1.3.3.5 TRNG0 Poker Maximum Limit (TRNG0_PKRMAX)

37.1.3.3.5.1 Address

Register	Offset	Description
TRNG0_PKRMAX	Ch	Accessible at this address when TRNG0_MCTL[PRGM] = 1]

37.1.3.3.5.2 Function

The TRNG0 Poker Maximum Limit Register defines Maximum Limit allowable during the TRNG Statistical Check Poker Test. Note that this offset (0x0C) is used as TRNG0_PKRMAX only if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this offset is used as the TRNG0_PKRSQ readback register.

37.1.3.3.5.3 Diagram

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

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

37.1.3.3.5.4 Fields

Field	Function
31-24 —	
23-0 PKR_MAX	Poker Maximum Limit. During the TRNG Statistical Checks, a "Poker Test" is run which requires a maximum and minimum limit. The maximum allowable result is programmed in the TRNG0_PKRMAX[PKR_MAX] register. This field is writable only if TRNG0_MCTL[PRGM] bit is 1. This register is cleared to the default POR value by writing the TRNG0_MCTL[RST_DEF] bit to 1. Note that the TRNG0_PKRMAX and TRNG0_PKRRNG registers combined are used to define the minimum allowable Poker result, which is PKR_MAX - PKR_RNG + 1. Note that if TRNG0_MCTL[PRGM] bit is 0, this register address is used to read the Poker Test Square Calculation result in register TRNG0_PKRSQ, as defined in the following section.

37.1.3.3.6 TRNG0 Poker Square Calculation Result (TRNG0_PKRSQ)

37.1.3.3.6.1 Address

Register	Offset	Description
TRNG0_PKRSQ	Ch	Accessible at this address when TRNG0_MCTL[PRGM] = 0]

37.1.3.3.6.2 Function

The TRNG0 Poker Square Calculation Result Register is a read-only register used to read the result of the TRNG Statistical Check Poker Test's Square Calculation. This test starts with the TRNG0_PKRMAX value and decreases towards a final result, which is read here. For the Poker Test to pass, this final result must be less than the programmed TRNG0_PKRRNG value. Note that this offset (0x0C) is used as TRNG0_PKRMAX if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this offset is used as TRNG0_PKRSQ readback register, as described here.

37.1.3.3.6.3 Diagram

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

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

37.1.3.3.6.4 Fields

Field	Function
31-24 —	
23-0 PKR_SQ	Poker Square Calculation Result. During the TRNG Statistical Checks, a "Poker Test" is run which starts with the value TRNG0_PKRMAX[PKR_MAX]. This value decreases according to a "sum of squares" algorithm, and must remain greater than zero, but less than the TRNG0_PKRRNG[PKR_RNG] limit. The resulting value may be read through this register, if TRNG0_MCTL[PRGM] bit is 0. Note that if TRNG0_MCTL[PRGM] bit is 1, this register address is used to access the Poker Test Maximum Limit in register TRNG0_PKRMAX, as defined in the previous section.

37.1.3.3.7 TRNG0 Seed Control (TRNG0_SDCTL)

37.1.3.3.7.1 Address

Register	Offset
TRNG0_SDCTL	10h

37.1.3.3.7.2 Function

The TRNG0 Seed Control Register contains two fields. One field defines the length (in system clocks) of each Entropy sample (ENT_DLY), and the other field indicates the number of samples that will taken during each TRNG Entropy generation (SAMP_SIZE).

37.1.3.3.7.3 Diagram

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

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

37.1.3.3.7.4 Fields

Field	Function
31-16 ENT_DLY	Entropy Delay. Defines the length (in system clocks) of each Entropy sample taken. This field is writable only if TRNG0_MCTL[PRGM] bit is 1. This field will read zeroes if TRNG0_MCTL[PRGM] = 0. This field is cleared to its reset value at POR.
15-0 SAMP_SIZE	Sample Size. Defines the total number of Entropy samples that will be taken during Entropy generation. This field is writable only if TRNG0_MCTL[PRGM] bit is 1. This field will read zeroes if TRNG0_MCTL[PRGM] = 0. This field is cleared to the default POR value by writing the TRNG0_MCTL[RST_DEF] bit to 1.

37.1.3.3.8 TRNG0 Sparse Bit Limit (TRNG0_SBLIM)

37.1.3.3.8.1 Address

Register	Offset	Description
TRNG0_SBLIM	14h	Accessible at this address when TRNG0_MCTL[PRGM] = 1]

37.1.3.3.8.2 Function

The TRNG0 Sparse Bit Limit Register is used when Von Neumann sampling is selected during Entropy Generation. It defines the maximum number of consecutive Von Neumann samples which may be discarded before an error is generated. Note that this address (0x14) is used as TRNG0_SBLIM only if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this address is used as TRNG0_TOTSAM readback register.

37.1.3.3.8.3 Diagram

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

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

37.1.3.3.8.4 Fields

Field	Function
31-10 —	Reserved. Always 0.
9-0 SB_LIM	Sparse Bit Limit. During Von Neumann sampling (if enabled by TRNG0_MCTL[SAMP_MODE], samples are discarded if two consecutive raw samples are both 0 or both 1. If this discarding occurs for a long period of time, it indicates that there is insufficient Entropy. The Sparse Bit Limit defines the maximum number of consecutive samples that may be discarded before an error is generated. This field is writable only if TRNG0_MCTL[PRGM] bit is 1. This register is cleared to the default POR value by writing the TRNG0_MCTL[RST_DEF] bit to 1. Note that if TRNG0_MCTL[PRGM] bit is 0, this register address is used to read the Total Samples count in register TRNG0_TOTSAM, as defined in the following section.

37.1.3.3.9 TRNG0 Total Samples (TRNG0_TOTSAM)

37.1.3.3.9.1 Address

Register	Offset	Description
TRNG0_TOTSAM	14h	Accessible at this address when TRNG0_MCTL[PRGM] = 0]

37.1.3.3.9.2 Function

The TRNG0 Total Samples Register is a read-only register used to read the total number of samples taken during Entropy generation. It is used to give an indication of how often a sample is actually used during Von Neumann sampling. Note that this offset (0x14) is used as TRNG0_SBLIM if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this offset is used as TRNG0_TOTSAM readback register, as described here.

37.1.3.3.9.3 Diagram

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

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

37.1.3.3.9.4 Fields

Field	Function
31-20 —	Reserved. Always 0.
19-0 TOT_SAM	Total Samples. During Entropy generation, the total number of raw samples is counted. This count is useful in determining how often a sample is used during Von Neumann sampling. The count may be read through this register, if TRNG0_MCTL[PRGM] bit is 0. Note that if TRNG0_MCTL[PRGM] bit is 1, this register address is used to access the Sparse Bit Limit in register TRNG0_SBLIM, as defined in the previous section.

37.1.3.3.10 TRNG0 Frequency Count Minimum Limit (TRNG0_FRQMIN)

37.1.3.3.10.1 Address

Register	Offset
TRNG0_FRQMIN	18h

37.1.3.3.10.2 Function

The TRNG0 Frequency Count Minimum Limit Register defines the minimum allowable count taken by the Entropy sample counter during each Entropy sample. During any sample period, if the count is less than this programmed minimum, a Frequency Count Fail is flagged in TRNG0_MCTL[FCT_FAIL] and an error is generated.

37.1.3.3.10.3 Diagram

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

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

37.1.3.3.10.4 Fields

Field	Function
31-22 —	Reserved. Always 0.
21-0 FRQ_MIN	Frequency Count Minimum Limit. Defines the minimum allowable count taken during each entropy sample. This field is writable only if TRNG0_MCTL[PRGM] bit is 1. This field will read zeroes if TRNG0_MCTL[PRGM] = 0. This field is cleared to its reset value at POR.

37.1.3.3.11 TRNG0 Frequency Count (TRNG0_FRQCNT)

37.1.3.3.11.1 Address

Register	Offset	Description
TRNG0_FRQCNT	1Ch	Accessible at this address when TRNG0_MCTL[PRGM] = 0]

37.1.3.3.11.2 Function

The TRNG0 Frequency Count Register is a read-only register used to read the frequency counter within the TRNG entropy generator. It will read all zeroes unless TRNG0_MCTL[TRNG_ACC] = 1. Note that this offset (0x1C) is used as TRNG0_FRQMAX if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this offset is used as TRNG0_FRQCNT readback register, as described here.

37.1.3.3.11.3 Diagram

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

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

37.1.3.3.11.4 Fields

Field	Function
31-22 —	Reserved. Always 0.
21-0 FRQ_CT	Frequency Count. If TRNG0_MCTL[TRNG_ACC] = 1, reads a sample frequency count taken during entropy generation. Requires TRNG0_MCTL[PRGM] = 0. Note that if TRNG0_MCTL[PRGM] bit is 1, this register address is used to access the Poker Test Maximum Limit in register TRNG0_PKRMAX, as defined in the previous section.

37.1.3.3.12 TRNG0 Frequency Count Maximum Limit (TRNG0_FRQMAX)

37.1.3.3.12.1 Address

Register	Offset	Description
TRNG0_FRQMAX	1Ch	Accessible at this address when TRNG0_MCTL[PRGM] = 1]

37.1.3.3.12.2 Function

The TRNG0 Frequency Count Maximum Limit Register defines the maximum allowable count taken by the Entropy sample counter during each Entropy sample. During any sample period, if the count is greater than this programmed maximum, a Frequency Count Fail is flagged in TRNG0_MCTL[FCT_FAIL] and an error is generated. Note that this address (001C) is used as TRNG0_FRQMAX only if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this address is used as TRNG0_FRQCNT readback register.

37.1.3.3.12.3 Diagram

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

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

37.1.3.3.12.4 Fields

Field	Function
31-22 —	Reserved. Always 0.
21-0 FRQ_MAX	Frequency Counter Maximum Limit. Defines the maximum allowable count taken during each entropy sample. This field is writable only if TRNG0_MCTL[PRGM] bit is 1. This field is cleared to its reset value at POR. Note that if TRNG0_MCTL[PRGM] bit is 0, this register address is used to read the Frequency Count result in register TRNG0_FRQCNT, as defined in the following section.

37.1.3.3.13 TRNG0 Statistical Check Monobit Count (TRNG0_SCMC)

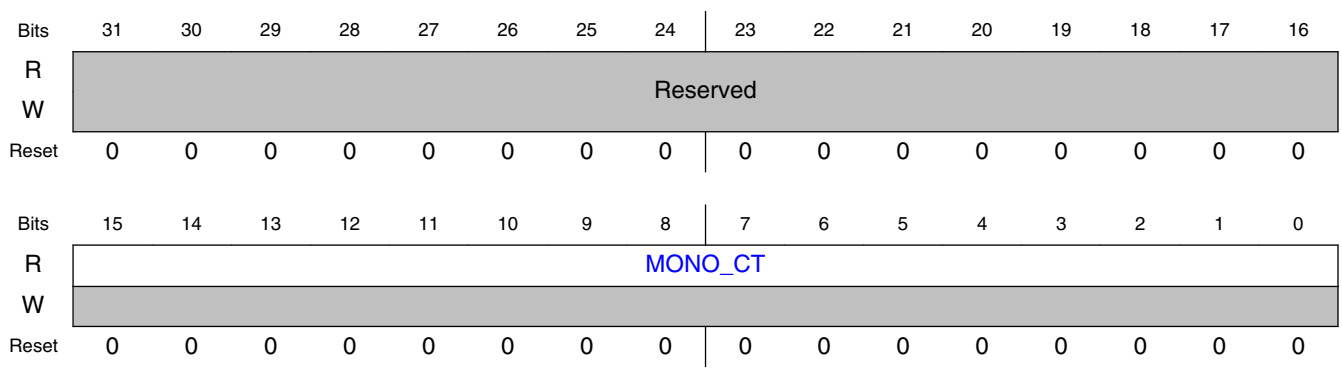
37.1.3.3.13.1 Address

Register	Offset	Description
TRNG0_SCMC	20h	Accessible at this address when TRNG0_MCTL[PRGM] = 0]

37.1.3.3.13.2 Function

The TRNG0 Statistical Check Monobit Count Register is a read-only register used to read the final monobit count after entropy generation. This counter starts with the value in TRNG0_SCML[MONO_MAX], and is decremented each time a one is sampled. Note that this offset (0x20) is used as TRNG0_SCML if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this offset is used as TRNG0_SCMC readback register, as described here.

37.1.3.3.13.3 Diagram



37.1.3.3.13.4 Fields

Field	Function
31-16 —	Reserved. Always 0.
15-0 MONO_CT	Monobit Count. Reads the final Monobit count after entropy generation. Requires TRNG0_MCTL[PRGM] = 0. Note that if TRNG0_MCTL[PRGM] bit is 1, this register address is used to access the Statistical Check Monobit Limit in register TRNG0_SCML, as defined in the previous section.

37.1.3.3.14 TRNG0 Statistical Check Monobit Limit (TRNG0_SCML)

37.1.3.3.14.1 Address

Register	Offset	Description
TRNG0_SCML	20h	Accessible at this address when TRNG0_MCTL[PRGM] = 1]

37.1.3.3.14.2 Function

The TRNG0 Statistical Check Monobit Limit Register defines the allowable maximum and minimum number of ones/zero detected during entropy generation. To pass the test, the number of ones/zeroes generated must be less than the programmed maximum value, and the number of ones/zeroes generated must be greater than (maximum - range). If this test fails, the Retry Counter in TRNG0_SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this offset (0x20) is used as TRNG0_SCML only if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this offset is used as TRNG0_SCMC readback register.

37.1.3.3.14.3 Diagram

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

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

37.1.3.3.14.4 Fields

Field	Function
31-16 MONO_RNG	Monobit Range. The number of ones/zeroes detected during entropy generation must be greater than MONO_MAX - MONO_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the TRNG0_MCTL[RST_DEF] bit to 1.

Table continues on the next page...

Field	Function
15-0 MONO_MAX	Monobit Maximum Limit. Defines the maximum allowable count taken during entropy generation. The number of ones/zeroes detected during entropy generation must be less than MONO_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the TRNG0_MCTL[RST_DEF] bit to 1.

37.1.3.3.15 TRNG0 Statistical Check Run Length 1 Count (TRNG0_SCR1C)

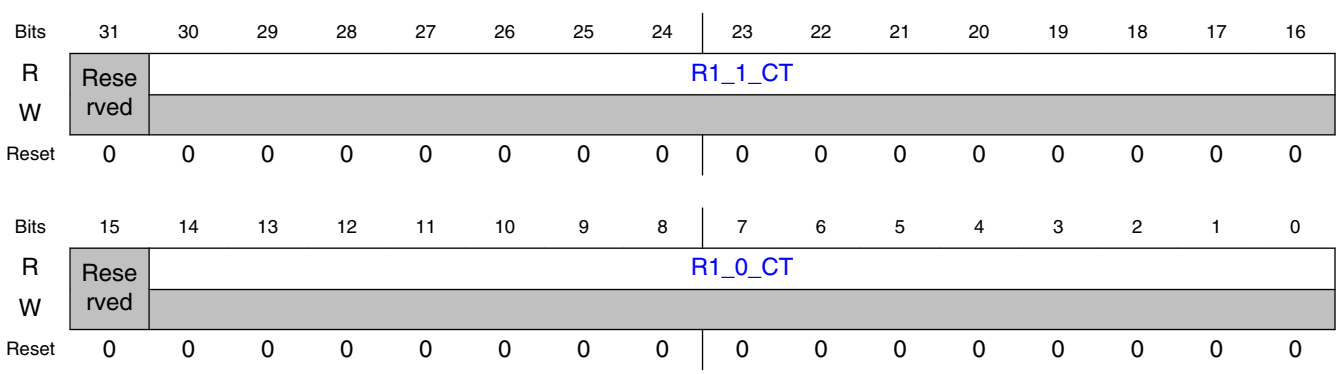
37.1.3.3.15.1 Address

Register	Offset	Description
TRNG0_SCR1C	24h	Accessible at this address when TRNG0_MCTL[PRGM] = 0]

37.1.3.3.15.2 Function

The TRNG0 Statistical Check Run Length 1 Counters Register is a read-only register used to read the final Run Length 1 counts after entropy generation. These counters start with the value in TRNG0_SCRxC1L[RUN1_MAX]. The R1_1_CT decrements each time a single one is sampled (preceded by a zero and followed by a zero). The R1_0_CT decrements each time a single zero is sampled (preceded by a one and followed by a one). Note that this offset (0x24) is used as TRNG0_SCRxC1L if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this offset is used as TRNG0_SCRxC1C readback register, as described here.

37.1.3.3.15.3 Diagram



37.1.3.3.15.4 Fields

Field	Function
31 —	Reserved. Always 0.
30-16 R1_1_CT	Runs of One, Length 1 Count. Reads the final Runs of Ones, length 1 count after entropy generation. Requires TRNG0_MCTL[PRGM] = 0.
15 —	Reserved. Always 0.
14-0 R1_0_CT	Runs of Zero, Length 1 Count. Reads the final Runs of Zeroes, length 1 count after entropy generation. Requires TRNG0_MCTL[PRGM] = 0.

37.1.3.3.16 TRNG0 Statistical Check Run Length 1 Limit (TRNG0_SCR1L)

37.1.3.3.16.1 Address

Register	Offset	Description
TRNG0_SCR1L	24h	Accessible at this address when TRNG0_MCTL[PRGM] = 1]

37.1.3.3.16.2 Function

The TRNG0 Statistical Check Run Length 1 Limit Register defines the allowable maximum and minimum number of runs of length 1 detected during entropy generation. To pass the test, the number of runs of length 1 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 1 must be greater than (maximum - range). If this test fails, the Retry Counter in TRNG0_SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x24) is used as TRNG0_SCRxC1L only if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this address is used as TRNG0_SCRxC1C readback register.

37.1.3.3.16.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								RUN1_RNG							
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								RUN1_MAX							
W																
Reset	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1

37.1.3.3.16.4 Fields

Field	Function
31 —	Reserved. Always 0.
30-16 RUN1_RNG	Run Length 1 Range. The number of runs of length 1 (for both 0 and 1) detected during entropy generation must be greater than RUN1_MAX - RUN1_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the TRNG0_MCTL[RST_DEF] bit to 1.
15 —	Reserved. Always 0.
14-0 RUN1_MAX	Run Length 1 Maximum Limit. Defines the maximum allowable runs of length 1 (for both 0 and 1) detected during entropy generation. The number of runs of length 1 detected during entropy generation must be less than RUN1_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the TRNG0_MCTL[RST_DEF] bit to 1.

37.1.3.3.17 TRNG0 Statistical Check Run Length 2 Count (TRNG0_SCR2C)

37.1.3.3.17.1 Address

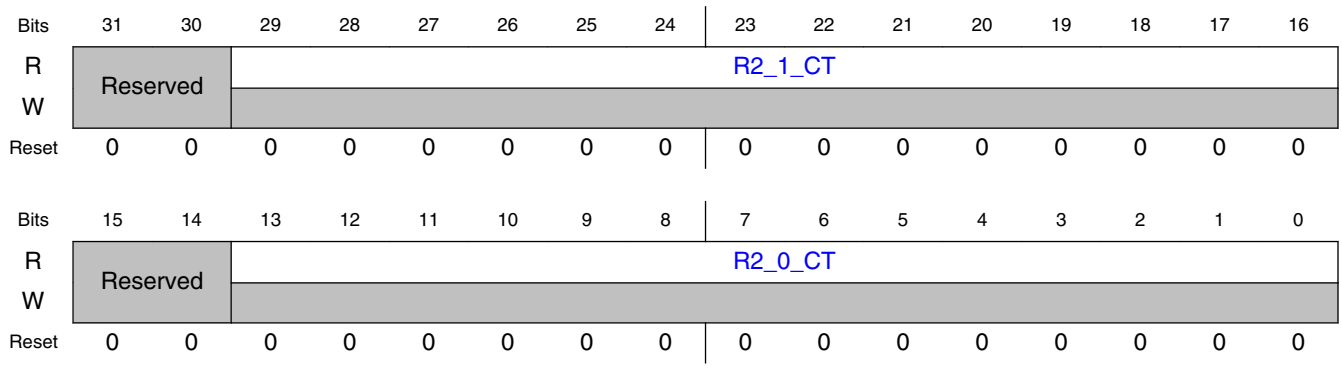
Register	Offset	Description
TRNG0_SCR2C	28h	Accessible at this address when TRNG0_MCTL[PRGM] = 0]

37.1.3.3.17.2 Function

The TRNG0 Statistical Check Run Length 2 Counters Register is a read-only register used to read the final Run Length 2 counts after entropy generation. These counters start with the value in TRNG0_SCRxC2L[RUN2_MAX]. The R2_1_CT decrements each time two consecutive ones are sampled (preceded by a zero and followed by a zero). The

R2_0_CT decrements each time two consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x28) is used as TRNG0_SCRxC2L if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this offset is used as TRNG0_SCRxC2C readback register, as described here.

37.1.3.3.17.3 Diagram



37.1.3.3.17.4 Fields

Field	Function
31-30 —	Reserved. Always 0.
29-16 R2_1_CT	Runs of One, Length 2 Count. Reads the final Runs of Ones, length 2 count after entropy generation. Requires TRNG0_MCTL[PRGM] = 0.
15-14 —	Reserved. Always 0.
13-0 R2_0_CT	Runs of Zero, Length 2 Count. Reads the final Runs of Zeroes, length 2 count after entropy generation. Requires TRNG0_MCTL[PRGM] = 0.

37.1.3.3.18 TRNG0 Statistical Check Run Length 2 Limit (TRNG0_SCR2L)

37.1.3.3.18.1 Address

Register	Offset	Description
TRNG0_SCR2L	28h	Accessible at this address when TRNG0_MCTL[PRGM] = 1]

37.1.3.3.18.2 Function

The TRNG0 Statistical Check Run Length 2 Limit Register defines the allowable maximum and minimum number of runs of length 2 detected during entropy generation. To pass the test, the number of runs of length 2 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 2 must be greater than (maximum - range). If this test fails, the Retry Counter in TRNG0_SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x28) is used as TRNG0_SCRxC2L only if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this address is used as TRNG0_SCRxC2C readback register.

37.1.3.3.18.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		RUN2_RNG													
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		RUN2_MAX													
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0

37.1.3.3.18.4 Fields

Field	Function
31-30 —	Reserved. Always 0.
29-16 RUN2_RNG	Run Length 2 Range. The number of runs of length 2 (for both 0 and 1) detected during entropy generation must be greater than RUN2_MAX - RUN2_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the TRNG0_MCTL[RST_DEF] bit to 1.
15-14 —	Reserved. Always 0.
13-0 RUN2_MAX	Run Length 2 Maximum Limit. Defines the maximum allowable runs of length 2 (for both 0 and 1) detected during entropy generation. The number of runs of length 2 detected during entropy generation must be less than RUN2_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the TRNG0_MCTL[RST_DEF] bit to 1.

37.1.3.3.19 TRNG0 Statistical Check Run Length 3 Count (TRNG0_SCR3C)

37.1.3.3.19.1 Address

Register	Offset	Description
TRNG0_SCR3C	2Ch	Accessible at this address when TRNG0_MCTL[PRGM] = 0]

37.1.3.3.19.2 Function

The TRNG0 Statistical Check Run Length 3 Counters Register is a read-only register used to read the final Run Length 3 counts after entropy generation. These counters start with the value in TRNG0_SCRxC3L[RUN3_MAX]. The R3_1_CT decrements each time three consecutive ones are sampled (preceded by a zero and followed by a zero). The R3_0_CT decrements each time three consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x2C) is used as TRNG0_SCRxC3L if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this offset is used as TRNG0_SCRxC3C readback register, as described here.

37.1.3.3.19.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved			R3_1_CT												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved			R3_0_CT												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

37.1.3.3.19.4 Fields

Field	Function
31-29 —	Reserved. Always 0.
28-16 R3_1_CT	Runs of Ones, Length 3 Count. Reads the final Runs of Ones, length 3 count after entropy generation. Requires TRNG0_MCTL[PRGM] = 0.
15-13	Reserved. Always 0.

Table continues on the next page...

Field	Function
—	
12-0 R3_0_CT	Runs of Zeroes, Length 3 Count. Reads the final Runs of Zeroes, length 3 count after entropy generation. Requires TRNG0_MCTL[PRGM] = 0.

37.1.3.3.20 TRNG0 Statistical Check Run Length 3 Limit (TRNG0_SCR3L)

37.1.3.3.20.1 Address

Register	Offset	Description
TRNG0_SCR3L	2Ch	Accessible at this address when TRNG0_MCTL[PRGM] = 1]

37.1.3.3.20.2 Function

The TRNG0 Statistical Check Run Length 3 Limit Register defines the allowable maximum and minimum number of runs of length 3 detected during entropy generation. To pass the test, the number of runs of length 3 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 3 must be greater than (maximum - range). If this test fails, the Retry Counter in TRNG0_SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x2C) is used as TRNG0_SCRxC3L only if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this address is used as TRNG0_SCRxC3C readback register.

37.1.3.3.20.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved			RUN3_RNG												
Reset	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved			RUN3_MAX												
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1

37.1.3.3.20.4 Fields

Field	Function
31-29 —	Reserved. Always 0.
28-16 RUN3_RNG	Run Length 3 Range. The number of runs of length 3 (for both 0 and 1) detected during entropy generation must be greater than RUN3_MAX - RUN3_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the TRNG0_MCTL[RST_DEF] bit to 1.
15-13 —	Reserved. Always 0.
12-0 RUN3_MAX	Run Length 3 Maximum Limit. Defines the maximum allowable runs of length 3 (for both 0 and 1) detected during entropy generation. The number of runs of length 3 detected during entropy generation must be less than RUN3_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the TRNG0_MCTL[RST_DEF] bit to 1.

37.1.3.3.21 TRNG0 Statistical Check Run Length 4 Count (TRNG0_SCR4C)

37.1.3.3.21.1 Address

Register	Offset	Description
TRNG0_SCR4C	30h	Accessible at this address when TRNG0_MCTL[PRGM] = 0]

37.1.3.3.21.2 Function

The TRNG0 Statistical Check Run Length 4 Counters Register is a read-only register used to read the final Run Length 4 counts after entropy generation. These counters start with the value in TRNG0_SCRxC4L[RUN4_MAX]. The R4_1_CT decrements each time four consecutive ones are sampled (preceded by a zero and followed by a zero). The R4_0_CT decrements each time four consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x30) is used as TRNG0_SCRxC4L if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this offset is used as TRNG0_SCRxC4C readback register, as described here.

37.1.3.3.21.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				R4_1_CT											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				R4_0_CT											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

37.1.3.3.21.4 Fields

Field	Function
31-28 —	Reserved. Always 0.
27-16 R4_1_CT	Runs of One, Length 4 Count. Reads the final Runs of Ones, length 4 count after entropy generation. Requires TRNG0_MCTL[PRGM] = 0.
15-12 —	Reserved. Always 0.
11-0 R4_0_CT	Runs of Zero, Length 4 Count. Reads the final Runs of Ones, length 4 count after entropy generation. Requires TRNG0_MCTL[PRGM] = 0.

37.1.3.3.22 TRNG0 Statistical Check Run Length 4 Limit (TRNG0_SCR4L)

37.1.3.3.22.1 Address

Register	Offset	Description
TRNG0_SCR4L	30h	Accessible at this address when TRNG0_MCTL[PRGM] = 1]

37.1.3.3.22.2 Function

The TRNG0 Statistical Check Run Length 4 Limit Register defines the allowable maximum and minimum number of runs of length 4 detected during entropy generation. To pass the test, the number of runs of length 4 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 4 must be greater than (maximum - range). If this test fails, the Retry Counter in TRNG0_SCMISC

will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x30) is used as TRNG0_SCRxC4L only if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this address is used as TRNG0_SCRxC4C readback register.

37.1.3.3.22.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved				RUN4_RNG											
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved				RUN4_MAX											
Reset	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	1

37.1.3.3.22.4 Fields

Field	Function
31-28 —	Reserved. Always 0.
27-16 RUN4_RNG	Run Length 4 Range. The number of runs of length 4 (for both 0 and 1) detected during entropy generation must be greater than RUN4_MAX - RUN4_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the TRNG0_MCTL[RST_DEF] bit to 1.
15-12 —	Reserved. Always 0.
11-0 RUN4_MAX	Run Length 4 Maximum Limit. Defines the maximum allowable runs of length 4 (for both 0 and 1) detected during entropy generation. The number of runs of length 4 detected during entropy generation must be less than RUN4_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the TRNG0_MCTL[RST_DEF] bit to 1.

37.1.3.3.23 TRNG0 Statistical Check Run Length 5 Count (TRNG0_SCR5C)

37.1.3.3.23.1 Address

Register	Offset	Description
TRNG0_SCR5C	34h	Accessible at this address when TRNG0_MCTL[PRGM] = 0]

37.1.3.3.23.2 Function

The TRNG0 Statistical Check Run Length 5 Counters Register is a read-only register used to read the final Run Length 5 counts after entropy generation. These counters start with the value in TRNG0_SCRxC5L[RUN5_MAX]. The R5_1_CT decrements each time five consecutive ones are sampled (preceded by a zero and followed by a zero). The R5_0_CT decrements each time five consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x34) is used as TRNG0_SCRxC5L if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this offset is used as TRNG0_SCRxC5C readback register, as described here.

37.1.3.3.23.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved					R5_1_CT										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved					R5_0_CT										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

37.1.3.3.23.4 Fields

Field	Function
31-27 —	Reserved. Always 0.
26-16 R5_1_CT	Runs of One, Length 5 Count. Reads the final Runs of Ones, length 5 count after entropy generation. Requires TRNG0_MCTL[PRGM] = 0.
15-11 —	Reserved. Always 0.
10-0 R5_0_CT	Runs of Zero, Length 5 Count. Reads the final Runs of Ones, length 5 count after entropy generation. Requires TRNG0_MCTL[PRGM] = 0.

37.1.3.3.24 TRNG0 Statistical Check Run Length 5 Limit (TRNG0_SCR5L)

37.1.3.3.24.1 Address

Register	Offset	Description
TRNG0_SCR5L	34h	Accessible at this address when TRNG0_MCTL[PRGM] = 1]

37.1.3.3.24.2 Function

The TRNG0 Statistical Check Run Length 5 Limit Register defines the allowable maximum and minimum number of runs of length 5 detected during entropy generation. To pass the test, the number of runs of length 5 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 5 must be greater than (maximum - range). If this test fails, the Retry Counter in TRNG0_SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x34) is used as TRNG0_SCRxC5L only if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this address is used as TRNG0_SCRxC5C readback register.

37.1.3.3.24.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved					RUN5_RNG										
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved					RUN5_MAX										
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1

37.1.3.3.24.4 Fields

Field	Function
31-27 —	Reserved. Always 0.
26-16 RUN5_RNG	Run Length 5 Range. The number of runs of length 5 (for both 0 and 1) detected during entropy generation must be greater than RUN5_MAX - RUN5_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the TRNG0_MCTL[RST_DEF] bit to 1.
15-11	Reserved. Always 0.

Table continues on the next page...

Field	Function
—	
10-0 RUN5_MAX	Run Length 5 Maximum Limit. Defines the maximum allowable runs of length 5 (for both 0 and 1) detected during entropy generation. The number of runs of length 5 detected during entropy generation must be less than RUN5_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the TRNG0_MCTL[RST_DEF] bit to 1.

37.1.3.3.25 TRNG0 Statistical Check Run Length 6+ Count (TRNG0_SCR6PC)

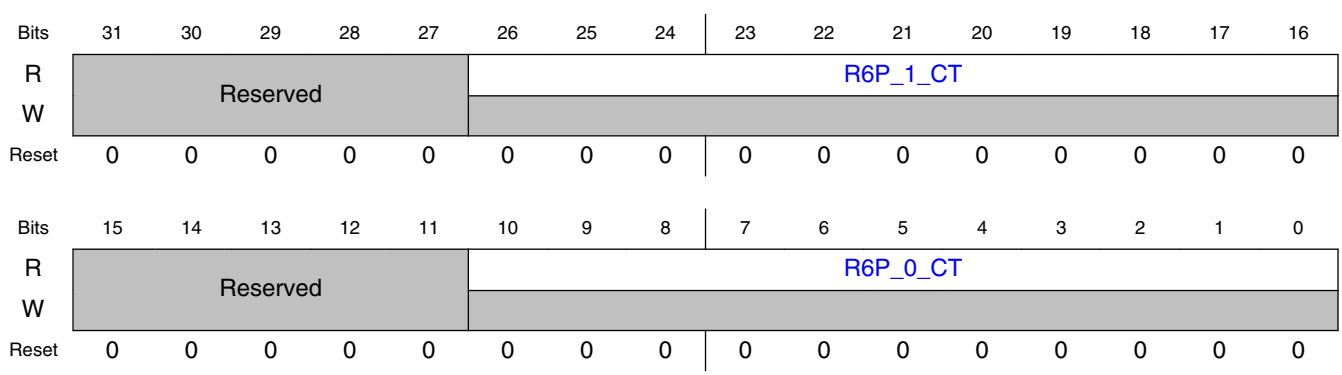
37.1.3.3.25.1 Address

Register	Offset	Description
TRNG0_SCR6PC	38h	Accessible at this address when TRNG0_MCTL[PRGM] = 0]

37.1.3.3.25.2 Function

The TRNG0 Statistical Check Run Length 6+ Counters Register is a read-only register used to read the final Run Length 6+ counts after entropy generation. These counters start with the value in TRNG0_SCRxC6PL[RUN6P_MAX]. The R6P_1_CT decrements each time six or more consecutive ones are sampled (preceded by a zero and followed by a zero). The R6P_0_CT decrements each time six or more consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x38) is used as TRNG0_SCRxC6PL if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this offset is used as TRNG0_SCRxC6PC readback register, as described here.

37.1.3.3.25.3 Diagram



37.1.3.3.25.4 Fields

Field	Function
31-27 —	Reserved. Always 0.
26-16 R6P_1_CT	Runs of One, Length 6+ Count. Reads the final Runs of Ones, length 6+ count after entropy generation. Requires TRNG0_MCTL[PRGM] = 0.
15-11 —	Reserved. Always 0.
10-0 R6P_0_CT	Runs of Zero, Length 6+ Count. Reads the final Runs of Ones, length 6+ count after entropy generation. Requires TRNG0_MCTL[PRGM] = 0.

37.1.3.3.26 TRNG0 Statistical Check Run Length 6+ Limit (TRNG0_SCR6PL)

37.1.3.3.26.1 Address

Register	Offset	Description
TRNG0_SCR6PL	38h	Accessible at this address when TRNG0_MCTL[PRGM] = 1]

37.1.3.3.26.2 Function

The TRNG0 Statistical Check Run Length 6+ Limit Register defines the allowable maximum and minimum number of runs of length 6 or more detected during entropy generation. To pass the test, the number of runs of length 6 or more (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 6 or more must be greater than (maximum - range). If this test fails, the Retry Counter in TRNG0_SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this offset (0x38) is used as TRNG0_SCRxC6PL only if TRNG0_MCTL[PRGM] is 1. If TRNG0_MCTL[PRGM] is 0, this offset is used as TRNG0_SCRxC6PC readback register.

37.1.3.3.26.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved					RUN6P_RNG										
Reset	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved					RUN6P_MAX										
Reset	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1

37.1.3.3.26.4 Fields

Field	Function
31-27 —	Reserved. Always 0.
26-16 RUN6P_RNG	Run Length 6+ Range. The number of runs of length 6 or more (for both 0 and 1) detected during entropy generation must be greater than RUN6P_MAX - RUN6P_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the TRNG0_MCTL[RST_DEF] bit to 1.
15-11 —	Reserved. Always 0.
10-0 RUN6P_MAX	Run Length 6+ Maximum Limit. Defines the maximum allowable runs of length 6 or more (for both 0 and 1) detected during entropy generation. The number of runs of length 6 or more detected during entropy generation must be less than RUN6P_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the TRNG0_MCTL[RST_DEF] bit to 1.

37.1.3.3.27 TRNG0 Status (TRNG0_STATUS)

37.1.3.3.27.1 Address

Register	Offset
TRNG0_STATUS	3Ch

37.1.3.3.27.2 Function

Various statistical tests are run as a normal part of the TRNG's entropy generation process. The least-significant 16 bits of the TRNG0_STATUS register reflect the result of each of these tests. The status of these bits will be valid when the TRNG has finished its entropy generation process. Software can determine when this occurs by polling the ENT_VAL bit in the TRNG0 Miscellaneous Control Register.

Note that there is a very small probability that a statistical test will fail even though the TRNG is operating properly. If this happens the TRNG will automatically retry the entire entropy generation process, including running all the statistical tests. The value in RETRY_CT is decremented each time an entropy generation retry occurs. If a statistical check fails when the retry count is nonzero, a retry is initiated. But if a statistical check fails when the retry count is zero, an error is generated by the RNG. By default RETRY_CT is initialized to 1, but software can increase the retry count by writing to the RTY_CT field in the TRNG0_SCMISC register.

All 0s will be returned if this register address is read while the RNG is in Program Mode (see PRGM field in TRNG0_MCTL register. If this register is read while the RNG is in Run Mode the value returned will be formatted as follows.

37.1.3.3.27.3 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TFMB	TFP	TFLR	TFSB	TF6PBR1	TF6PBR0	TF5BR1	TF5BR0	TF4BR1	TF4BR0	TF3BR1	TF3BR0	TF2BR1	TF2BR0	TF1BR1	TF1BR0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

37.1.3.3.27.4 Fields

Field	Function
31-20 —	Reserved. Always 0.
19-16 RETRY_CT	RETRY COUNT. This represents the current number of entropy generation retries left before a statistical text failure will cause the RNG to generate an error condition.
15	Test Fail, Mono Bit. If TFMB=1, the Mono Bit Test has failed.

Table continues on the next page...

Field	Function
TFMB	
14 TFP	Test Fail, Poker. If TFP=1, the Poker Test has failed.
13 TFLR	Test Fail, Long Run. If TFLR=1, the Long Run Test has failed.
12 TFSB	Test Fail, Sparse Bit. If TFSB=1, the Sparse Bit Test has failed.
11 TF6PBR1	Test Fail, 6 Plus Bit Run, Sampling 1s. If TF6PBR1=1, the 6 Plus Bit Run, Sampling 1s Test has failed.
10 TF6PBR0	Test Fail, 6 Plus Bit Run, Sampling 0s. If TF6PBR0=1, the 6 Plus Bit Run, Sampling 0s Test has failed.
9 TF5BR1	Test Fail, 5-Bit Run, Sampling 1s. If TF5BR1=1, the 5-Bit Run, Sampling 1s Test has failed.
8 TF5BR0	Test Fail, 5-Bit Run, Sampling 0s. If TF5BR0=1, the 5-Bit Run, Sampling 0s Test has failed.
7 TF4BR1	Test Fail, 4-Bit Run, Sampling 1s. If TF4BR1=1, the 4-Bit Run, Sampling 1s Test has failed.
6 TF4BR0	Test Fail, 4-Bit Run, Sampling 0s. If TF4BR0=1, the 4-Bit Run, Sampling 0s Test has failed.
5 TF3BR1	Test Fail, 3-Bit Run, Sampling 1s. If TF3BR1=1, the 3-Bit Run, Sampling 1s Test has failed.
4 TF3BR0	Test Fail, 3-Bit Run, Sampling 0s. If TF3BR0=1, the 3-Bit Run, Sampling 0s Test has failed.
3 TF2BR1	Test Fail, 2-Bit Run, Sampling 1s. If TF2BR1=1, the 2-Bit Run, Sampling 1s Test has failed.
2 TF2BR0	Test Fail, 2-Bit Run, Sampling 0s. If TF2BR0=1, the 2-Bit Run, Sampling 0s Test has failed.
1 TF1BR1	Test Fail, 1-Bit Run, Sampling 1s. If TF1BR1=1, the 1-Bit Run, Sampling 1s Test has failed.
0 TF1BR0	Test Fail, 1-Bit Run, Sampling 0s. If TF1BR0=1, the 1-Bit Run, Sampling 0s Test has failed.

37.1.3.3.28 TRNG0 Entropy Read (TRNG0_ENTa)

37.1.3.3.28.1 Address

For a = 0 to 15:

Register	Offset	Description
TRNG0_ENTa	40h + (a × 4h)	Word a

37.1.3.3.28.2 Function

The RNG TRNG can be programmed to generate an entropy value that is readable via the SkyBlue bus. To do this, set the TRNG0_MCTL[TRNG_ACC] bit to 1. Once the entropy value has been generated, the TRNG0_MCTL[ENT_VAL] bit will be set to 1. At this point, TRNG0_ENT0 through TRNG0_ENT15 may be read to retrieve the 512-bit entropy value. Note that once TRNG0_ENT15 is read, the entropy value will be cleared and a new value will begin generation, so it is important that TRNG0_ENT15 be read last. These registers are readable only when TRNG0_MCTL[PRGM] = 0 (Run Mode), TRNG0_MCTL[TRNG_ACC] = 1 (TRNG access mode) and TRNG0_MCTL[ENT_VAL] = 1. After at most one (1) bus clock cycle of reading a valid TRNG0_ENT15 register value, reading any TRNG0_ENT0 through TRNG0_ENT15 register would return zeroes.

37.1.3.3.28.3 Diagram

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

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

37.1.3.3.28.4 Fields

Field	Function
31-0 ENT	Entropy Value. Will be non-zero only if TRNG0_MCTL[PRGM] = 0 Run Mode) and TRNG0_MCTL[ENT_VAL] = 1 (Entropy Valid). The most significant bits of the entropy are read from the lowest offset, and the least significant bits are read from the highest offset. Note that reading the highest offset also clears the entire entropy value, and starts a new entropy generation.

37.1.3.3.29 TRNG0 Statistical Check Poker Count 1 and 0 (TRNG0_PKRCNT10)

37.1.3.3.29.1 Address

Register	Offset
TRNG0_PKRCNT10	80h

37.1.3.3.29.2 Function

The TRNG0 Statistical Check Poker Count 1 and 0 Register is a read-only register used to read the final Poker test counts of 1h and 0h patterns. The Poker 0h Count increments each time a nibble of sample data is found to be 0h. The Poker 1h Count increments each time a nibble of sample data is found to be 1h. Note that this register is readable only if TRNG0_MCTL[PRGM] is 0, otherwise zeroes will be read.

37.1.3.3.29.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PKR_1_CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PKR_0_CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

37.1.3.3.29.4 Fields

Field	Function
31-16 PKR_1_CT	Poker 1h Count. Total number of nibbles of sample data which were found to be 1h. Requires TRNG0_MCTL[PRGM] = 0.
15-0 PKR_0_CT	Poker 0h Count. Total number of nibbles of sample data which were found to be 0h. Requires TRNG0_MCTL[PRGM] = 0.

37.1.3.3.30 TRNG0 Statistical Check Poker Count 3 and 2 (TRNG0_PKRCNT32)

37.1.3.3.30.1 Address

Register	Offset
TRNG0_PKRCNT32	84h

37.1.3.3.30.2 Function

The TRNG0 Statistical Check Poker Count 3 and 2 Register is a read-only register used to read the final Poker test counts of 3h and 2h patterns. The Poker 2h Count increments each time a nibble of sample data is found to be 2h. The Poker 3h Count increments each time a nibble of sample data is found to be 3h. Note that this register is readable only if TRNG0_MCTL[PRGM] is 0, otherwise zeroes will be read.

37.1.3.3.30.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PKR_3_CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PKR_2_CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

37.1.3.3.30.4 Fields

Field	Function
31-16 PKR_3_CT	Poker 3h Count. Total number of nibbles of sample data which were found to be 3h. Requires TRNG0_MCTL[PRGM] = 0.
15-0 PKR_2_CT	Poker 2h Count. Total number of nibbles of sample data which were found to be 2h. Requires TRNG0_MCTL[PRGM] = 0.

37.1.3.3.31 TRNG0 Statistical Check Poker Count 5 and 4 (TRNG0_PKRCNT54)

37.1.3.3.31.1 Address

Register	Offset
TRNG0_PKRCNT54	88h

37.1.3.3.31.2 Function

The TRNG0 Statistical Check Poker Count 5 and 4 Register is a read-only register used to read the final Poker test counts of 5h and 4h patterns. The Poker 4h Count increments each time a nibble of sample data is found to be 4h. The Poker 5h Count increments each time a nibble of sample data is found to be 5h. Note that this register is readable only if TRNG0_MCTL[PRGM] is 0, otherwise zeroes will be read.

37.1.3.3.31.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PKR_5_CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PKR_4_CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

37.1.3.3.31.4 Fields

Field	Function
31-16 PKR_5_CT	Poker 5h Count. Total number of nibbles of sample data which were found to be 5h. Requires TRNG0_MCTL[PRGM] = 0.
15-0 PKR_4_CT	Poker 4h Count. Total number of nibbles of sample data which were found to be 4h. Requires TRNG0_MCTL[PRGM] = 0.

37.1.3.3.32 TRNG0 Statistical Check Poker Count 7 and 6 (TRNG0_PKRCNT76)

37.1.3.3.32.1 Address

Register	Offset
TRNG0_PKRCNT76	8Ch

37.1.3.3.32.2 Function

The TRNG0 Statistical Check Poker Count 7 and 6 Register is a read-only register used to read the final Poker test counts of 7h and 6h patterns. The Poker 6h Count increments each time a nibble of sample data is found to be 6h. The Poker 7h Count increments each time a nibble of sample data is found to be 7h. Note that this register is readable only if TRNG0_MCTL[PRGM] is 0, otherwise zeroes will be read.

37.1.3.3.32.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PKR_7_CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PKR_6_CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

37.1.3.3.32.4 Fields

Field	Function
31-16 PKR_7_CT	Poker 7h Count. Total number of nibbles of sample data which were found to be 7h. Requires TRNG0_MCTL[PRGM] = 0.
15-0 PKR_6_CT	Poker 6h Count. Total number of nibbles of sample data which were found to be 6h. Requires TRNG0_MCTL[PRGM] = 0.

37.1.3.3.33 TRNG0 Statistical Check Poker Count 9 and 8 (TRNG0_PKRCNT98)

37.1.3.3.33.1 Address

Register	Offset
TRNG0_PKRCNT98	90h

37.1.3.3.33.2 Function

The TRNG0 Statistical Check Poker Count 9 and 8 Register is a read-only register used to read the final Poker test counts of 9h and 8h patterns. The Poker 8h Count increments each time a nibble of sample data is found to be 8h. The Poker 9h Count increments each time a nibble of sample data is found to be 9h. Note that this register is readable only if TRNG0_MCTL[PRGM] is 0, otherwise zeroes will be read.

37.1.3.3.33.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PKR_9_CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PKR_8_CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

37.1.3.3.33.4 Fields

Field	Function
31-16 PKR_9_CT	Poker 9h Count. Total number of nibbles of sample data which were found to be 9h. Requires TRNG0_MCTL[PRGM] = 0.
15-0 PKR_8_CT	Poker 8h Count. Total number of nibbles of sample data which were found to be 8h. Requires TRNG0_MCTL[PRGM] = 0.

37.1.3.3.34 TRNG0 Statistical Check Poker Count B and A (TRNG0_PKRCNTBA)

37.1.3.3.34.1 Address

Register	Offset
TRNG0_PKRCNTBA	94h

37.1.3.3.34.2 Function

The TRNG0 Statistical Check Poker Count B and A Register is a read-only register used to read the final Poker test counts of Bh and Ah patterns. The Poker Ah Count increments each time a nibble of sample data is found to be Ah. The Poker Bh Count increments each time a nibble of sample data is found to be Bh. Note that this register is readable only if TRNG0_MCTL[PRGM] is 0, otherwise zeroes will be read.

37.1.3.3.34.3 Diagram

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

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

37.1.3.3.34.4 Fields

Field	Function
31-16 PKR_B_CT	Poker Bh Count. Total number of nibbles of sample data which were found to be Bh. Requires TRNG0_MCTL[PRGM] = 0.
15-0 PKR_A_CT	Poker Ah Count. Total number of nibbles of sample data which were found to be Ah. Requires TRNG0_MCTL[PRGM] = 0.

37.1.3.3.35 TRNG0 Statistical Check Poker Count D and C (TRNG0_PKRCNTDC)

37.1.3.3.35.1 Address

Register	Offset
TRNG0_PKRCNTDC	98h

37.1.3.3.35.2 Function

The TRNG0 Statistical Check Poker Count D and C Register is a read-only register used to read the final Poker test counts of Dh and Ch patterns. The Poker Ch Count increments each time a nibble of sample data is found to be Ch. The Poker Dh Count increments each time a nibble of sample data is found to be Dh. Note that this register is readable only if TRNG0_MCTL[PRGM] is 0, otherwise zeroes will be read.

37.1.3.3.35.3 Diagram

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

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

37.1.3.3.35.4 Fields

Field	Function
31-16 PKR_D_CT	Poker Dh Count. Total number of nibbles of sample data which were found to be Dh. Requires TRNG0_MCTL[PRGM] = 0.
15-0 PKR_C_CT	Poker Ch Count. Total number of nibbles of sample data which were found to be Ch. Requires TRNG0_MCTL[PRGM] = 0.

37.1.3.3.36 TRNG0 Statistical Check Poker Count F and E (TRNG0_PKRCNTFE)

37.1.3.3.36.1 Address

Register	Offset
TRNG0_PKRCNTFE	9Ch

37.1.3.3.36.2 Function

The TRNG0 Statistical Check Poker Count F and E Register is a read-only register used to read the final Poker test counts of Fh and Eh patterns. The Poker Eh Count increments each time a nibble of sample data is found to be Eh. The Poker Fh Count increments each time a nibble of sample data is found to be Fh. Note that this register is readable only if TRNG0_MCTL[PRGM] is 0, otherwise zeroes will be read.

37.1.3.3.36.3 Diagram

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

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

37.1.3.3.36.4 Fields

Field	Function
31-16 PKR_F_CT	Poker Fh Count. Total number of nibbles of sample data which were found to be Fh. Requires TRNG0_MCTL[PRGM] = 0.
15-0 PKR_E_CT	Poker Eh Count. Total number of nibbles of sample data which were found to be Eh. Requires TRNG0_MCTL[PRGM] = 0.

37.1.3.3.37 TRNG0 Security Configuration (TRNG0_SEC_CFG)

37.1.3.3.37.1 Address

Register	Offset
TRNG0_SEC_CFG	A0h

37.1.3.3.37.2 Function

The TRNG0 Security Configuration Register is a read/write register used to control the test mode, programmability and state modes of the TRNG0. Many bits are place holders for this version. More configurability will be added here. Clears on asynchronous reset. For TRNG0 releases before 2014/July/01, offsets 0xA0 to 0xAC used to be 0xB0 to 0xBC respectively. So, update newer tests that use these registers, if hard coded.

37.1.3.3.37.3 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved													SK_VAL	NO_PRGM	SH0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

37.1.3.3.37.4 Fields

Field	Function
31-3 —	Reserved.
2 SK_VAL	Reserved. DRNG-specific, not applicable to this version. 0 - See DRNG version. 1 - See DRNG version.
1 NO_PRGM	If set, the TRNG registers cannot be programmed. That is, regardless of the TRNG access mode in the TRNG0 Miscellaneous Control Register. 0 - Programability of registers controlled only by the TRNG0 Miscellaneous Control Register's access mode bit. 1 - Overides TRNG0 Miscellaneous Control Register access mode and prevents TRNG register programming.
0 SH0	Reserved. DRNG specific, not applicable to this version. 0 - See DRNG version. 1 - See DRNG version.

37.1.3.3.38 TRNG0 Interrupt Control (TRNG0_INT_CTRL)

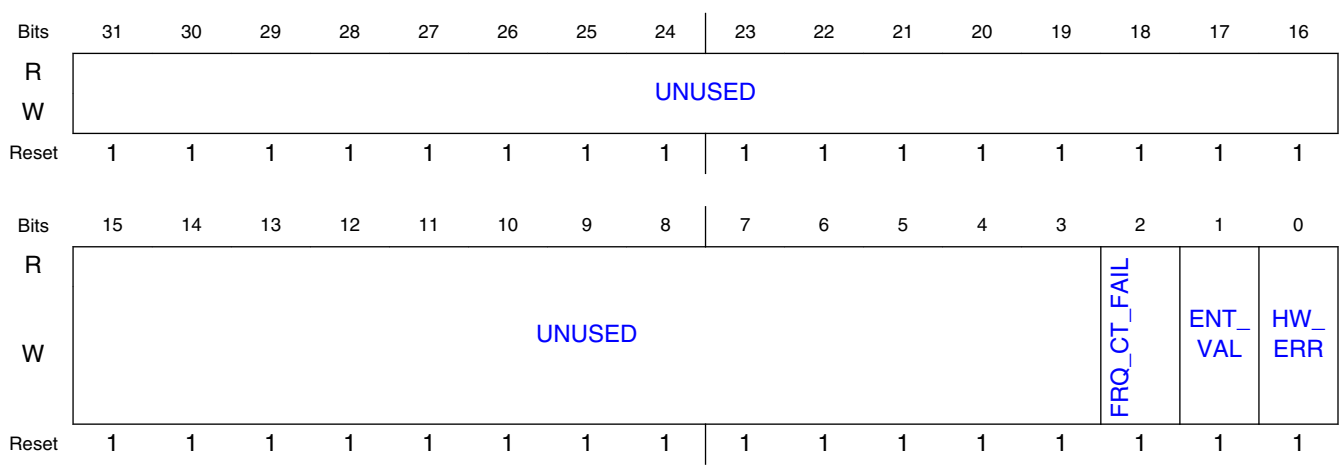
37.1.3.3.38.1 Address

Register	Offset
TRNG0_INT_CTRL	A4h

37.1.3.3.38.2 Function

The TRNG0 Interrupt Control Register is a read/write register used to control the status for the (currently) three important interrupts that are generated by the TRNG. See TRNG0_INT_STATUS register description above. Each interrupt can be cleared by de-asserting the corresponding bit in the TRNG0_INT_CTRL register. Only a new interrupt will reassert the corresponding bit in the status register. Even if the interrupt is cleared or masked, interrupt status information can be read from the TRNG0_MCTL register.

37.1.3.3.38.3 Diagram



37.1.3.3.38.4 Fields

Field	Function
31-3 UNUSED	Reserved but writeable.
2 FRQ_CT_FAIL	Same behavior as bit 0 above. 0 - Same behavior as bit 0 above.

Table continues on the next page...

Field	Function
	1 - Same behavior as bit 0 above.
1 ENT_VAL	Same behavior as bit 0 above. 0 - Same behavior as bit 0 above. 1 - Same behavior as bit 0 above.
0 HW_ERR	Bit position that can be cleared if corresponding bit of TRNG0_INT_STATUS has been asserted. 0 - Corresponding bit of TRNG0_INT_STATUS cleared. 1 - Corresponding bit of TRNG0_INT_STATUS active.

37.1.3.3.39 TRNG0 Mask (TRNG0_INT_MASK)

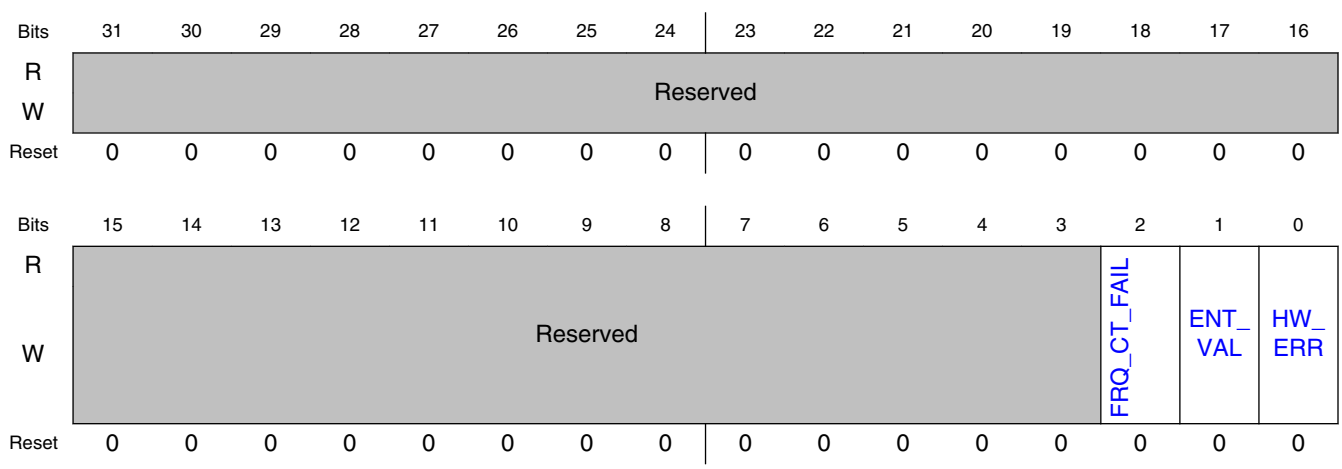
37.1.3.3.39.1 Address

Register	Offset
TRNG0_INT_MASK	A8h

37.1.3.3.39.2 Function

The TRNG0 Interrupt Mask Register is a read/write register used to disable/mask the status reporting of the (currently) three important interrupts that are generated by the TRNG. See TRNG0_INT_STATUS register description above. Each interrupt can be masked/disabled by de-asserting the corresponding bit in the TRNG0_INT_MASK register. Only setting this bit high will re-enable the interrupt in the status register. Even if the interrupt is cleared or masked, interrupt status information can be read from the TRNG0_MCTL register.

37.1.3.3.39.3 Diagram



37.1.3.3.39.4 Fields

Field	Function
31-3 —	Reserved.
2 FRQ_CT_FAIL	Same behavior as bit 0 above. 0 - Same behavior as bit 0 above. 1 - Same behavior as bit 0 above.
1 ENT_VAL	Same behavior as bit 0 above. 0 - Same behavior as bit 0 above. 1 - Same behavior as bit 0 above.
0 HW_ERR	Bit position that can be cleared if corresponding bit of TRNG0_INT_STATUS has been asserted. 0 - Corresponding interrupt of TRNG0_INT_STATUS is masked. 1 - Corresponding bit of TRNG0_INT_STATUS is active.

37.1.3.3.40 TRNG0 Interrupt Status (TRNG0_INT_STATUS)

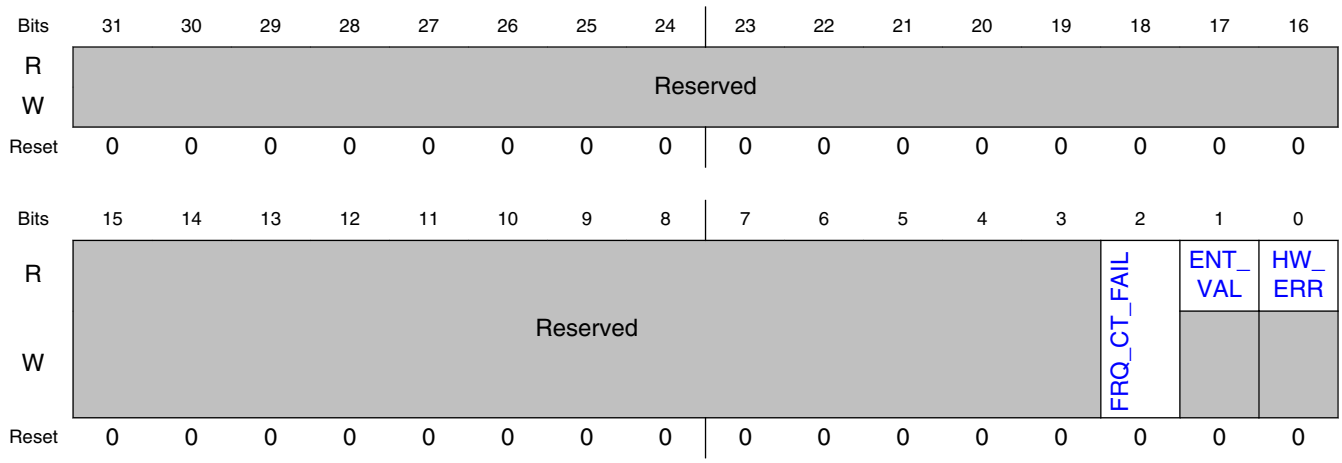
37.1.3.3.40.1 Address

Register	Offset
TRNG0_INT_STATUS	ACh

37.1.3.3.40.2 Function

The TRNG0 Interrupt Status Register is a read register used to control and provide status for the (currently) three important interrupts that are generated by the TRNG. The ipi_rng_int_b interrupt signals that TRNG0 has either generated a Frequency Count Fail, Entropy Valid or Error Interrupt. The cause of the interrupt can be decoded by checking the least significant bits of the TRNG0_INT_STATUS register. Each interrupt can be temporarily cleared by de-asserting the corresponding bit in the TRNG0_INT_CTRL register. To mask the interrupts, clear the corresponding bits in the TRNG0_INT_MASK register. The description of each of the 3 interrupts is defined in the Block Guide under the TRNG0_MCTL register description. Even if the interrupt is cleared or masked, interrupt status information can be read from the TRNG0_MCTL register.

37.1.3.3.40.3 Diagram



37.1.3.3.40.4 Fields

Field	Function
31-3 —	Reserved.
2 FRQ_CT_FAIL	Read only: Frequency Count Fail. The frequency counter has detected a failure. This may be due to improper programming of the TRNG0_FRQMAX and/or TRNG0_FRQMIN registers, or a hardware failure in the ring oscillator. 0 - No hardware nor self test frequency errors. 1 - The frequency counter has detected a failure.
1 ENT_VAL	Read only: Entropy Valid. Will assert only if TRNG ACC bit is set, and then after an entropy value is generated. Will be cleared when TRNG0_ENT15 is read. (TRNG0_ENT0 through TRNG0_ENT14 should be read before reading TRNG0_ENT15). 0 - Busy generation entropy. Any value read is invalid. 1 - TRNG can be stopped and entropy is valid if read.
0 HW_ERR	Read: Error status. 1 = error detected. 0 = no error. Any HW error in the TRNG will trigger this interrupt. 0 - no error 1 - error detected.

37.1.3.3.41 TRNG0 Version ID (MS) (TRNG0_VID1)

37.1.3.3.41.1 Address

Register	Offset
TRNG0_VID1	F0h

37.1.3.3.41.2 Function

The TRNG0 Version ID Register is a read only register used to identify the version of the TRNG in use. This register as well as TRNG0_VID2 should both be read to verify the expected version.

37.1.3.3.41.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TRNG0_IP_ID															
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TRNG0_MAJ_REV								TRNG0_MIN_REV							
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

37.1.3.3.41.4 Fields

Field	Function
31-16 TRNG0_IP_ID	Shows the Freescale IP ID. 0000000000110000 - ID for TRNG.
15-8 TRNG0_MAJ_REV	Shows the Freescale IP's Major revision of the TRNG. 00000001 - Major revision number for TRNG.
7-0 TRNG0_MIN_REV	Shows the Freescale IP's Minor revision of the TRNG. 00000000 - Minor revision number for TRNG.

37.1.3.3.42 TRNG0 Version ID (LS) (TRNG0_VID2)

37.1.3.3.42.1 Address

Register	Offset
TRNG0_VID2	F4h

37.1.3.3.42.2 Function

The TRNG0 Version ID Register LSB is a read only register used to identify the architecture of the TRNG in use. This register as well as TRNG0_VID1 should both be read to verify the expected version.

37.1.3.3.42.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TRNG0_ERA								TRNG0_INTG_OPT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TRNG0_ECO_REV								TRNG0_CONFIG_OPT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

37.1.3.3.42.4 Fields

Field	Function
31-24 TRNG0_ERA	Shows the Freescale compile options for the TRNG. 00000000 - COMPILE_OPT for TRNG.
23-16 TRNG0_INTG_OPT	Shows the Freescale integration options for the TRNG. 00000000 - INTG_OPT for TRNG.
15-8 TRNG0_ECO_REV	Shows the Freescale IP's ECO revision of the TRNG. 00000000 - TRNG_ECO_REV for TRNG.
7-0 TRNG0_CONFIG_OPT	Shows the Freescale IP's Configuaration options for the TRNG. 00000000 - TRNG_CONFIG_OPT for TRNG.

37.1.4 Another TRNG usage example.

The TRNG can be used by a post processing pseudo-random number generator function. For example, TRNG can be used to seed a hardware or software based implementation of a DRBG defined by SP800-90.

Chapter 38
LP Trusted Cryptography (LTC)

38.1 LP Trusted Cryptography Block Diagram

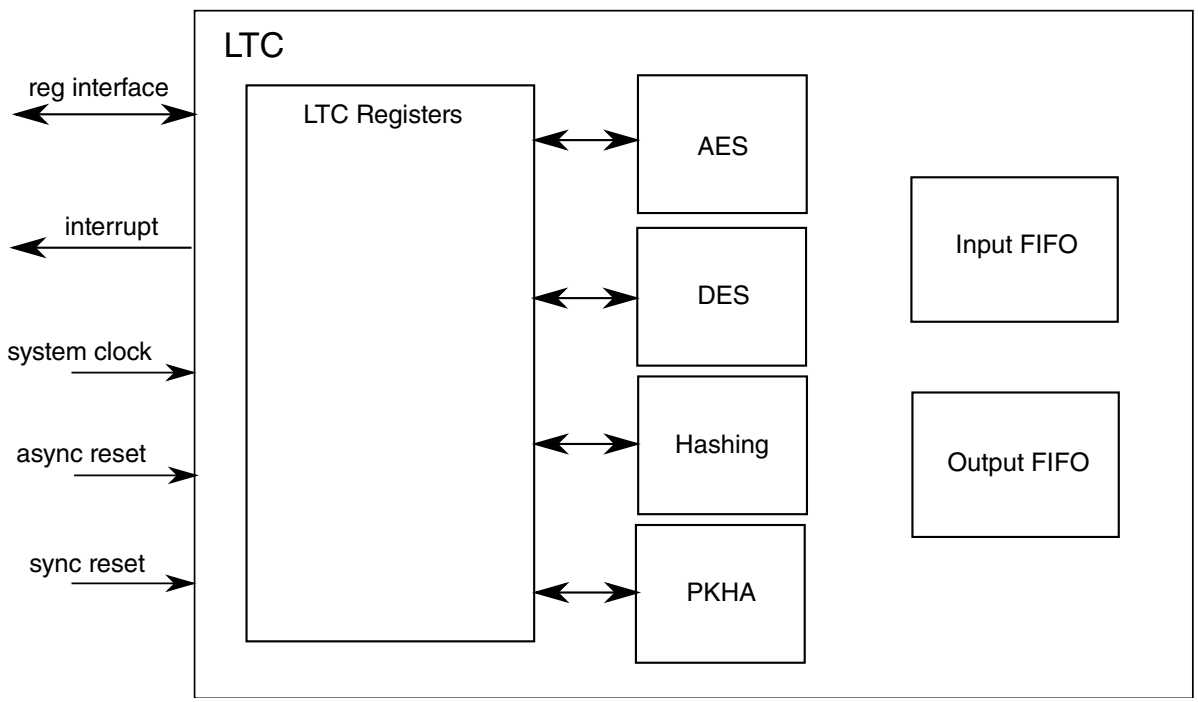


Figure 38-1. LTC Block Diagram

38.2 Feature summary

LTC includes the following features:

- Cryptographic authentication
  - Hashing algorithms
    - SHA-1

        - SHA-224
        - SHA-256
    - Message authentication codes (MAC)
        - AES-CMAC
        - AES-XCBC-MAC
    - Auto padding
    - ICV checking
- Authenticated encryption algorithms
    - AES-CCM (counter with CBC-MAC)
    - AES-GCM (Galois counter mode)
- Symmetric key block ciphers
    - AES (128-bit, 192-bit or 256-bit keys)
    - DES (64-bit keys, including key parity)
    - 3DES (128-bit or 192-bit keys, including key parity)
    - Cipher modes
        - ECB, CBC, CTR for AES
        - ECB, CBC, CFB, OFB for DES
- Public key cryptography
    - Modular Arithmetic
        - Addition, subtraction, multiplication, exponentiation, reduction, inversion, greatest common denominator
        - Both integer and binary polynomial functions
        - Modulus size up to 2048 bits
        - Arithmetic operations performed with 16-bit-digit arithmetic unit
        - Timing-equalized and normal versions of modular exponentiation
    - DSA up to 2048 bits
    - Diffie-Hellman (DH) key agreement up to 2048 bits
    - RSA up to 2048 bits
    - Primality testing
        - Maximum size 2048 bits
    - Elliptic curve cryptography
        - Point add, point double, point multiply
        - Point validation (is point on curve)
        - Timing-equalized and normal versions of point multiplication
        - Both prime field and binary polynomial field functions
        - Elliptic curve Diffie-Hellman key agreement
        - Modulus size up to 512 bits
- Secure Scan

38.3 AES accelerator (AESA) functionality

The advanced encryption standard accelerator (AESA) module is a hardware co-processor capable of accelerating the advanced encryption standard (AES) cryptographic algorithm.

38.3.1 Differences between the AES encrypt and decrypt keys

The decrypt form of the key is different from the encrypt form of the key, because AES successively modifies the cryptographic key during the steps of the cryptographic operation. The decryption operation yields the correct result only if the modified form of the key (the decrypt key) is used at the beginning of the decryption operation. Unless told otherwise (via the DK bit in the Mode Register), AES assumes that a key loaded from memory is the encrypt key, that is, the form appropriate for encryption. If a decryption operation is specified and DK = 0, AES first goes through the steps required to derive the decrypt key from the encrypt key, and then performs the decryption operation. If a decryption operation is specified and DK = 1, the steps required to derive the decrypt key are skipped and the decryption operation is performed immediately, significantly improving performance for small data blocks.

Note that the difference between the encrypt key and the decrypt key must be taken into account when sharing keys between jobs. When an AES decryption job loads a key from memory, it is probably an encrypt key, so the DK bit in the Mode Register should be set to 0 so that AES derives the decrypt key from the encrypt key before beginning the decryption operation. But when a subsequent AES decryption job shares the key from a previous decryption job, the key that is shared is a decrypt key. In that case, the DK bit should be set to 1, which tells AES to skip the key derivation steps. If DK were set to 0 in this case, the decrypt key would be modified as if it were an encrypt key, and consequently, the wrong key value would be used in the decryption operation.

38.3.2 AESA modes of operation

The following modes are supported by AESA:

- Electronic codebook (ECB)
- Cipher block chaining (CBC)
- Counter (CTR)
- Extended cipher block chaining message authentication code (XCBC-MAC)
- Cipher-based MAC (CMAC)

- CTR and CBC-MAC (CCM)
- Galois/Counter mode (GCM)

AES modes can be classified into these categories:

- Confidentiality (ECB, CBC, CTR)
- Authenticated Confidentiality (CCM, GCM)
- Authentication (XCBC-MAC, CMAC)

CBC Mode can also be viewed as an authentication mode when used to encrypt data, because it provides CBC-MAC in the context registers.

38.3.3 AESA use of registers

Note the following regarding the AESA's use of registers:

- For all modes, if AES is selected and the mode code written to the Mode Register does not correspond to any of the implemented AES modes, the illegal-mode error is generated.
- KEY SIZE, MODE and DATA SIZE can be written in any order. The operation will begin after all of these have been written.
- When sharing context between consecutive AES jobs, software reset is not issued. To prepare AES for the next job, the Data Size Register and Mode Register must be cleared, as well as the Done Interrupt. The order of these should be such that the Done Interrupt is not cleared first.
- If ICV-only jobs are created (no data to be processed, only ICV to be checked) in modes that support ICV check, the AS mode field should be reset.

38.3.4 AES ECB mode

The electronic codebook (ECB) mode is a confidentiality mode that features, for a given key, the assignment of a fixed, ciphertext block to each plaintext block, analogous to the assignment of code words in a codebook. In ECB encryption, the forward cipher function is applied directly and independently to each block of the plaintext. The resulting sequence of output blocks is the ciphertext. In ECB decryption, the inverse cipher function is applied directly and independently to each block of the ciphertext. The resulting sequence of output blocks is the plaintext.

38.3.4.1 AES ECB mode use of the Mode Register

AES ECB mode uses the Mode Register as follows:

- The Encrypt (ENC) field should be 1 for ECB encryption and 0 for ECB decryption.
- The ICV/TEST bit is used in ECB mode to activate the fault detection test logic. This logic verifies that the fault detection logic is operational by injecting bit-level errors into input data and key bytes. Because ECB mode does not normally use the Context Registers, the first 128 bits of the context are used in the ECB TEST mode to define which byte of the input data and the key has a bit error injected.
- The Algorithm State (AS) field is not used in ECB mode.
- The Additional Algorithm Information (AAI) field must be set with value 20h that activates ECB mode. Setting the MSB in the AAI field (interpreted as the Decrypt Key or DK bit for AES operations) specifies that the key loaded to the Key Register is the decryption form of the key, rather than the encryption form of the key. If DK = 0, when a decryption operation is requested LTC processes the content of the Key Register to yield the decryption form of the key. If DK = 1, skips this processing. The illegal-mode error is generated if DK = 1 and ENC=1.
- The Algorithm (ALG) field is used to activate AESA by setting it to 10h.

38.3.4.2 AES ECB mode use of the Context Register

ECB does not use Context Registers.

38.3.4.3 AES ECB Mode use of the Data Size Register

The length of the message to be processed in bytes must be written to the Data Size register. If this value is not divisible by 16, the Data Size error is generated.

38.3.4.4 AES ECB Mode use of the Key Register

ECB keys must be written to the Key Register and can have 16, 24, or 32 bytes.

38.3.4.5 AES ECB Mode use of the Key Size Register

The number of bytes in the ECB key must be written to the Key Size register. The KEY SIZE, MODE and DATA SIZE can be written in any order. Processing starts after all of them have been written. Any value other than 16, 24, or 32 causes the key-size error to be generated.

38.3.5 AES CBC mode

The CBC mode is described in this table.

Table 38-1. AES CBC Mode

Name	Abbreviation	Function
Cipher-block chaining mode	CBC	Confidentiality mode whose encryption process features the combining ("chaining") of the plaintext blocks with the previous ciphertext blocks. The CBC mode requires an IV (Initialization Vector) to combine with the first plaintext block **NOTE:** CBC mode uses both forward and inverse AES cipher.

38.3.5.1 AES CBC mode use of the Mode Register

The AES CBC mode uses the Mode Register as follows:

- The Encrypt (ENC) field should be 1 for encryption and 0 for decryption.
- The ICV/TEST bit is not used in these modes.
- The Algorithm State (AS) field is used only in CBC mode to prevent IV update in the context for the last data block when set to "Finalize" (2h).
- The Additional Algorithm Information (AAI) field defines which mode is used for processing. For CBC this value is 10h. The Decrypt Key [DK] (AAI field MSB) bit affects CBC mode and specifies that the key loaded to the Key Register is the decrypt key. The illegal mode error is generated if DK=1 and ENC=1.
- The Algorithm (ALG) field is used to activate AESA by setting it to 10h.

38.3.5.2 AES CBC mode use of the Context Register

The AES CBC mode uses the Context Register as follows:

- AES CBC uses the Context Registers to provide IV, which is updated with every processed block of a message. When a message is split into chunks and processed in multiple sessions, the IV must be saved and later restored for the next chunk to be processed correctly. At the end of CBC processing, IV is also the MAC of the message.
- If the AS field of the Mode Register is set to "Finalize" (2h) in the CBC mode, the last IV update is not written to the context. This enables CBC encryption to effectively perform ECB encryption transformation of a single-block message

located in the context in place of IV, and with an all-zero block provided as input data through the FIFO without overwriting the context.

Table 38-2. Context usage in CBC mode

Context Word	Definition
0	IV [127:96]
1	IV [95:64]
2	IV [63:32]
3	IV [31:0]

38.3.5.3 AES CBC mode use of the Data Size Register

The AES CBC mode uses the Data Size Register as follows:

- The byte length of the message to be processed must be written to the Data Size Register.
- After the Data Size Register is written, its value must be divisible by 16 in CBC mode, otherwise the data-size error is generated.

38.3.5.4 AES CBC mode use of the Key Register

The AES CBC mode uses the Key Register as follows:

- A CBC key must be written to the Key Register.
- Keys can have 16, 24, or 32 bytes.

38.3.5.5 AES CBC mode use of the Key Size Register

The AES CBC mode uses the Key Size Register as follows:

- The number of bytes in a key must be written to the Key Size register.
- Any value other than 16, 24, or 32 causes a key-size error to be generated.

38.3.6 AES CTR mode

The counter (CTR) mode is a confidentiality mode that features the application of the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice

versa. Note that the counter value must be unique for each data block that is encrypted with the same key. uses a 128-bit counter to ensure that the counter value will not overflow and wrap around.

NOTE

It is the user's responsibility to ensure that the same key value is not used again following a reset.

38.3.6.1 AES CTR mode use of the Mode Register

The AES CTR mode uses the Mode Register as follows:

- The Additional Algorithm Information (AAI) field should be set to 00h to activate CTR mode. If the Decrypt Key [DK] (AAI field MSB) bit is set, the illegal-mode error is generated, because CTR uses only forward AES cipher requiring encryption rather than decryption keys.
- The Algorithm State (AS) field when set to "Finalize" (2h) prevents counter update in the context for the last data block.
- The Algorithm (ALG) field is used to activate AESA by setting it to 10h.

38.3.6.2 AES CTR mode use of the Context Register

The AES CTR mode uses the Context Register as follows:

- CTR uses context words 4,5,6 and 7 to provide initial counter value (CTR0). This value is incremented with every processed block of a message. When a message is split into chunks and processed in multiple sessions, the CTRi field of context has to be saved and later restored for the next chunk to be processed correctly.
- If the AS field of the Mode Register is set to Finalize (2h) in the CTR mode, the last counter update is not written to the context. This enables CTR encryption to effectively perform ECB encryption transformation of a single-block message located in the context words 4,5,6 and 7 in place of CTR0 and with all-zero block provided as input data through the FIFO without overwriting the context.

Table 38-3. Context usage in CTR mode

Context Word	Initial-input definition	Context-switching definition
0	-	-
1	-	-
2	-	-
3	-	-

Table continues on the next page...

Table 38-3. Context usage in CTR mode (continued)

Context Word	Initial-input definition	Context-switching definition
4	CTR0 [127:96]	CTRi [127:96]
5	CTR0 [95:64]	CTRi [95:64]
6	CTR0 [63:32]	CTRi [63:32]
7	CTR0 [31:0]	CTRi [31:0]

38.3.6.3 AES CTR mode use of the Data Size Register

The byte-length of the message to be processed must be written to the Data Size register. After the Data Size register is written, the value of this register may not be divisible by 16. CTR decrements the value in this register with every processed block.

38.3.6.4 AES CTR mode use of the Key Register

- CTR key must be written to the Key Register.
- The Key Register can have 16, 24 or 32 bytes.

38.3.6.5 AES CTR mode use of the Key Size Register

The number of bytes in a key must be written to the Key Size register. Any value other than 16, 24, or 32 will cause Key Size error to be generated.

38.3.7 AES XCBC-MAC and CMAC modes

The AES XCBC-MAC and CMAC modes are described together because of their similarities. They are extensions of the AES CBC mode that produces a key-dependent, one--way hash (or message authentication code (MAC)) in a secure fashion across messages of varying lengths. They also provide data-integrity and data-origin authentication regarding the original message source.

38.3.7.1 AES XCBC-MAC and CMAC modes use of the Mode Register

The AES XCBC-MAC and CMAC modes use the Mode Register as follows:

- The Encrypt (ENC) bit is ignored.

- The ICV_TEST bit must be set for computed MAC to be compared with the received MAC. The received MAC must be written to the Input Data FIFO after message data. If this bit is not set, XCBC-MAC and CMAC do not expect received ICV to be supplied after message data.
- The Algorithm State (AS) field is defined for XCBC-MAC as shown in this table.

Table 38-4. Mode Register[AS] operation selections in AES XCBC-MAC

Operation	Description
INITIALIZE	Message is processed in multiple sessions and the current session is the first one. During initialization, derived keys K3 and K2 that are XOR-ed with the last message block are computed and stored in the context to be used in the last processing session. The derived key K1 used as an AES key is computed and written back to the Key Register over the original key
INITIALIZE/FINALIZE	Message is processed in a single XCBC session and the final MAC is computed
UPDATE	Message is processed in multiple sessions and the current session is neither the first nor the last. Derived keys K2 and K3 are provided in the context and the derived key K1 is provided in the Key Register. If decryption is requested, and data size is not written or is set to 0, and ICV_TEST bit is 1 - AS = UPDATE means that Check ICV (CICV) job is requested. The CICV-only job does not process any data, it just pops received ICV/MAC from the Input Data FIFO, and compares it to the computed MAC that is restored with the rest of the context from the previous session.
FINALIZE	Message is processed in multiple sessions and the current session is the last one. Derived keys K2 and K3 are provided in the context and the derived key K1 is provided in the Key Register. The final MAC is computed

- The Algorithm State (AS) field is defined for CMAC as shown in this table.

Table 38-5. Mode Register[AS] operation selections in CMAC

Operation	Function
INITIALIZE	Message is processed in multiple sessions and the current session is the first one. During initialization, the constant L = E(K, 0) is computed as encrypted block of zeros using key K and stored in the context to be used in the last processing session for derivation of keys K1 and K2. One of these keys will be XOR-ed with the last message block.
INITIALIZE/FINALIZE	Message is processed in a single session and the final MAC is computed
UPDATE	Message is processed in multiple sessions and the current session is neither the first nor the last. The constant L used for key derivation is provided in the context. If decryption is requested, and data size is not written or is set to 0, and ICV_TEST:w bit is 1 - AS = UPDATE means that Check ICV (CICV) job is requested. The CICV-only job does not process any data, it just pops received ICV/MAC from the Input Data FIFO, and compares it to the computed MAC that is restored with the rest of the context from the previous session
FINALIZE	Message is processed in multiple sessions and the current session is the last one. The constant L used for key derivation is provided in the context. The final MAC is computed

- If the AS field is not set to either "Initialize/Finalize" or "Finalize" and the ICV_TEST bit is set to 1, the illegal-mode error is generated, except for CICV-only jobs.

- The Additional Algorithm Information (AAI) field must be set to 70h for XCBC and 60h for CMAC to be activated. Setting the DK bit (AAI field MSB) will cause the Illegal Mode error.
- The Algorithm (ALG) field is used to activate AESA by setting it to 10h.

38.3.7.2 AES XCBC-MAC and CMAC Modes use of the Context Register

The AES XCBC-MAC and CMAC modes use the Context Register as follows:

- No data needs to be provided in the context when starting a new XCBC or CMAC session.
- The computed MAC and the derived keys K2 and K3 are written back to the context by XCBC.
- The computed MAC and the constant L = E(K,0), computed as encrypted block of zeros using key K, are written back to the context by CMAC.
- When a message is split into chunks and processed in multiple sessions, these values need to be saved before context switch and restored before the next chunk of a message is to be processed. At the end of message processing the first 2 dwords of the context contain the MAC value.

Table 38-6. Context usage in XCBC-MAC and CMAC modes

Mode	Context dword	Context-switching definition	Final-result definition
XCBC-MAC	0	MAC[127:64]	MAC[127:64]
	1	MAC[63:0]	MAC[63:0]
	2	K3[127:64]	-
	3	K3[63:0]	-
	4	K2[127:64]	-
	5	K2[63:0]	-
CMAC	0	MAC[127:64]	MAC[127:64]
	1	MAC[63:0]	MAC[63:0]
	2	L[127:64]	-
	3	L[63:0]	-

38.3.7.3 AES XCBC-MAC and CMAC modes use of the ICV Size Register

The AES XCBC-MAC and CMAC modes use the ICV Size Register as follows:

- This register is used to provide received ICV/MAC byte-size.

- The computed ICV/MAC written to the context in the XCBC mode is always 16 bytes.
- In CMAC mode, this register determines also the computed MAC size-the remaining bytes are cleared.
- Supported values for ICV size are 4 to 16 bytes. If this register is 0, the size of ICV is 16 bytes.

38.3.7.4 AES XCBC-MAC and CMAC modes use of the Data Size Register

The AES XCBC-MAC and CMAC modes use the Data Size Register as follows:

- The byte-length of the message to be processed must be written to the Data Size register.
- XCBC-MAC and CMAC decrement the value in this register with every processed block.

38.3.7.5 AES XCBC-MAC and CMAC modes use of the Key Register

The AES XCBC-MAC and CMAC modes use the Key Register as follows:

- The key must be written to this register.
- For XCBC-MAC, if the AS mode field is set to either "Initialize" or "Initialize/ Finalize", it is the original XCBC key (K) that must be written here. Otherwise, the derived key (K1) must be restored to this register. CMAC only uses original key K as an AES key.

38.3.7.6 AES XCBC-MAC and CMAC modes use of the Key Size Register

The AES XCBC-MAC and CMAC modes use the Key Size Register as follows:

- The total number of key bytes must be written to the Key Size register.
- For XCBC-MAC, any value other than 16 causes a key-size error to be generated. For CMAC, this error is generated only if any value other than 16, 24, or 32 is written.

38.3.7.7 ICV checking in AES XCBC-MAC and CMAC modes

Automatic ICV checking is enabled by setting the ICV_TEST bit of the Mode Register to 1. When ICV is set to 1, the AS mode field must be set to either "Finalize" or "Initialize/ Finalize"; otherwise the illegal-mode error is generated, except for CICV-only (Check-ICV-only) jobs.

The received ICV must be provided on the FIFO after the message data. The size of the received and computed ICV is provided in the ICV Size register.

If the ICV check detects a mismatch between the decrypted received ICV and the computed ICV, the ICV error is generated.

38.3.8 AESA CCM and CCM* mode

CCM and CCM* consists of two related processes: generation encryption and decryption verification, which combine two cryptographic primitives: counter mode encryption (CTR) and cipher-block chaining based authentication (CBC-MAC). Only the forward cipher function of the block cipher algorithm is used within these primitives. Note that the counter value must be unique for each data block that is encrypted with the same key. uses a 128-bit counter to ensure that the counter value does not overflow and wrap around.

NOTE

It is the user's responsibility to ensure that the same key value is not used again following a reset.

38.3.8.1 Generation encryption

A cipher-block chaining is applied to the payload, the associated data (AAD), and the nonce to generate a message authentication code (MAC); then counter mode encryption is applied to the MAC and the payload to transform them into an unreadable form, called the ciphertext. Thus, CCM generation encryption expands the size of the payload by the size of the MAC.

38.3.8.2 Decryption verification

Counter-mode decryption is applied to the purported ciphertext to recover the MAC and the corresponding payload; then cipher block chaining is applied to the payload, the received associated data, and the received nonce to verify the correctness of the MAC.

38.3.8.3 AES CCM and CCM* mode use of the Mode Register

The AES CCM and CCM* mode uses the Mode Register as follows:

- The Encrypt (ENC) bit must be set to 1 for encryption and 0 for decryption.
- The ICV_TEST bit must be set for CCM and CCM* to compare computed MAC with the received MAC when decryption is requested.
- The received MAC must be written to the input-data FIFO after message data.
- Setting the ICV_TEST bit causes the received MAC to be decrypted and compared with the computed MAC.
- The number of MSBs to be compared is defined by the MAC size in the CCM IV (B_0) as described in the CCM and CCM* specification.
- If the AS field is set to FINALIZE, but ICV = 0, AESA does not expect received ICV to be put on the input-data FIFO. In that case, MAC is computed and truncated to the specified size for decryption.
- For encryption, the computed MAC is encrypted and truncated to size. The illegal-mode error is generated if ICV = 1 and ENC = 1.
- If ICV = 1 and the decrypted received MAC do not match computed MAC, the ICV error is generated.
- The Algorithm State (AS) field is defined for CCM as follows:

Table 38-7. Mode Register[AS] operation selections in AES CCM

Operation	Description
INITIALIZE	Message is processed in multiple sessions and the current session is the first one. During initialization, the initial counter CTR0 is encrypted in the CTR mode and the B0 is processed with the CBC-MAC mode. The resulting values are stored in the context. Also, the size of MAC is decoded from B0 and written to the context. This AS setting must be used whenever the first part (or whole) AAD is being processed
INITIALIZE/FINALIZE	Message is processed in a single CCM session and the final MAC is computed and encrypted. The initial counter CTR0 and B0 must be provided in the context
UPDATE	Message is processed in multiple sessions and the current session is neither the first nor the last. All context data is restored from the previous session and the key is written to the Key Register. If decryption is requested, and data size is not written or is set to 0, and ICV_TEST bit is 1 - AS=UPDATE means that a CICV-only job is requested. The CICV-only job does not process any data, it just pops received ICV/MAC from the Input Data FIFO, decrypts it and compares it to the computed MAC that is restored with the rest of the context from the previous session
FINALIZE	Message is processed in multiple sessions and the current session is the last one. All context data is restored from the previous session and the key is written to the Key Register. The final MAC is computed and encrypted

- Whenever AS is set to Initialize or Initialize/Finalize, context registers must be zero.
- If the AS field is not set to either Initialize/Finalize or Finalize and the ICV_TEST bit is set to 1, the illegal-mode error is generated. This does not apply in case when only ICV check is requested as described for AS = UPDATE.

- The Additional Algorithm Information (AAI) field must be set to 80h for CCM or CCM* to be activated. Setting the DK bit causes the illegal-mode error.
- The Algorithm (ALG) field is used to activate AESA by setting it to 10h.

38.3.8.4 AES CCM and CCM* modes use of the Context Register

The AES CCM and CCM* mode uses the Context Register as follows:

- B0 and the initial counter CTR0 must be provided in the context before the first chunk of the message is to be processed. During initialization, the initial counter CTR0 is encrypted in the CTR mode and B0 (which functions like a CBC-MAC IV in CCM and CCM*) is processed with the CBC-MAC mode. The resulting values are stored in the context. Also, the size of MAC is decoded from B0 and written to context word 13.
- If there is AAD, the first block of it defines its size, and that value is decoded and written to context word 12. All of the context data must be restored before the next chunk of the message is to be processed in multi-session processing.
- For CCM and CCM* encryption, the ICV (encrypted final MAC) is written to context words 8-11. For CCM and CCM* decryption, the ICV (received MAC), which is always encrypted, is decrypted to words 8-11. The final computed MAC is written (in clear) to context words 0-3.

Table 38-8. Context usage in CCM and CCM* mode encryption

Context Word	Initial-input definition	Intermediate definition	Final-output definition
0	B0[127:96]	-	MAC[127:96]
1	B0[95:64]	-	MAC[95:64]
2	B0[63:32]	-	MAC[63:32]
3	B0[31:0]	-	MAC[31:0]
4	CTR0[127:96]	CTR[127:96]	-
5	CTR0[95:64]	CTR[95:64]	-
6	CTR0[63:32]	CTR[63:32]	-
7	CTR0[31:0]	CTR[31:0]	-
8	-	E(CTR0)[127:96][1]	E(MAC)[127:96]
9	-	E(CTR0)[95:64] [1]	E(MAC)[95:64]
10	-	E(CTR0)[63:32] [1]	E(MAC)[63:32]
11	-	E(CTR0)[31:0] [1]	E(MAC)[31:0]
12	-	AAD size; see Table 38-10	-
13	-	MAC size; see Table 38-11	-

1. E(x) means encrypted x

Table 38-9. Context usage in CCM and CCM* modes decryption

Context Word	Initial-input definition	Context-switching Definition	Final-result definition
0	B0[127:96]	-	MAC[127:96]
1	B0[95:64]	-	MAC[95:64]
2	B0[63:32]	-	MAC[63:32]
3	B0[31:0]	-	MAC[31:0]
4	CTR0[127:96]	CTR[127:96]	-
5	CTR0[95:64]	CTR[95:64]	-
6	CTR0[63:32]	CTR[63:32]	-
7	CTR0[31:0]	CTR[31:0]	-
8	-	E(CTR0)[127:96]	Decrypted Received MAC[127:96]
9	-	E(CTR0)[95:64]	Decrypted Received MAC[95:64]
10	-	E(CTR0)[63:32]	Decrypted Received MAC[63:32]
11	-	E(CTR0)[31:0] [1]	Decrypted Received MAC[31:0]
12	-	AAD size; see Table 38-10	-
13	-	MAC size; see Table 38-11	-

Table 38-10. Format of Context Word 12 for AES-CCM and AES-CCM* mode

Bit 31	Bits 30-16	Bits 15-0
AAD Presence Flag	0	AAD Size

Table 38-11. Format of Context Word 13 for AES-CCM and AES-CCM* mode

Bits 31-3	Bits 2-0
0	Encoded MAC Size

38.3.8.5 AES CCM and CCM* mode use of the Data Size Register

The AES CCM and CCM* mode uses the Data Size Register as follows:

- The byte-length of the message to be processed must be written to the Data Size register.
- CCM decrements the value in this register with every processed block.
- The content of the Data Size register must be divisible by 16 after the last write to it if the AS mode field is set to either "Update" or "Initialize". Otherwise, the data-size

error is generated. In other words, message splitting can be done only on a 16-byte boundary.

38.3.8.6 AES CCM and CCM* mode use of the Key Register

CCM and CCM* key must be written to this register; it is always an encryption key.

38.3.8.7 AES CCM and CCM* mode use of the Key Size Register

The AES CCM and CCM* mode uses the Key Size Register as follows:

- The total number of key bytes must be written to the Key Size register.
- Any value other than 16, 24, or 32 causes a key-size error to be generated.

38.3.8.8 AES CCM and CCM* mode use of the ICV check

The AES CCM and CCM* mode uses ICV checking as follows:

- Automatic ICV checking is enabled by setting the ICV_TEST bit of the Mode Register to 1. When ICV is set to 1, the AS mode field must be set to either "Finalize" or "Initialize/Finalize"-otherwise the illegal-mode error is generated, unless data size is 0 indicating ICV check is only requested. Also, if ICV = 1, the ENC bit must be 0.
- The received ICV must be provided on the input data FIFO after the message data. In CCM, received ICV is always encrypted. The size of the received and computed ICV is for CCM encoded in the B0.
- If the ICV check detects mismatch between the decrypted received ICV and the computed ICV, the ICV error is generated.

38.3.9 AES GCM mode

The AES GCM provides the following:

- Data confidentiality using counter mode (CTR). Note that the counter value must be unique for each data block that is encrypted with the same key. uses a 128-bit counter to ensure that the counter value does not overflow and "wrap around", but it is the user's responsibility to ensure that the same key value is not used again following a reset.

- Authentication (assurance of integrity) of the confidential data using a universal hash function (GHASH) that is defined over a binary Galois (that is, finite) field. GCM can also provide authentication assurance for additional data (AAD) that is not encrypted.
- Stronger authentication assurance than a (non-cryptographic) checksum or error detecting code; in particular, GCM can detect both of the following:
  - Accidental modifications of the data
  - Intentional, unauthorized modifications

38.3.9.1 GMAC

If the GCM input is restricted to data that is not encrypted, the resulting specialization of GCM, called GMAC, is simply an authentication mode on the input data.

38.3.9.2 GCM data types

These data types must always be provided in the following order:

1. IV
2. AAD
3. Message data

Any of these may be missing.

38.3.9.3 IV processing

IV is processed using GHASH function if the size of IV is not 12 bytes. The result of IV processing is the initial counter (Y0) value used for encryption/decryption. GHASH function is also performed on AAD and textdata before the MAC can be computed.

38.3.9.4 GCM initialization

GCM initialization is completed when all of the IV data is processed and the initial counter value (Y0) is computed as a result. For that to happen, IV data needs to be supplied through the Input Data FIFO and the FIFO data type must be set to IV.

38.3.9.5 AES GCM mode use of the Mode Register

The AES GCM mode uses the Mode Register as follows:

- The Encrypt (ENC) bit must be set to 1 for encryption and 0 for decryption. Even though operations performed in either case are identical, the authentication is done of the cipher text in parallel with decryption when ENC = 0, and after encryption of each block when ENC = 1.
- The ICV_TEST bit must be set for GCM to compare computed MAC with the received MAC. The received MAC must be written to the input-data FIFO after message data and the FIFO data type must be set to ICV. If this bit is not set, GCM does not expect received ICV to be supplied after textdata. The illegal-mode error is generated if ICV = 1 and ENC = 1.
- The Algorithm State (AS) field is defined for GCM as shown in this table:

Table 38-12. Mode Register[AS] operation selections in AES GCM

Operation	Value	Description
INITIALIZE	1h	Message is processed in multiple sessions and the current session processes final part of IV or textdata; do the final GHASH step, but do not compute MAC. **NOTE:** This AS state does not indicate initialization in GCM; instead, it means that the final step of the GHASH function is to be performed. In general, whenever the final GHASH iteration needs to be computed (either for GHASH(IV) or GHASH(AAD, ciphertext)), and the current message size provided in the Data Size Register is not equal to the total size for either IV, AAD, or textdata, AS should be set to INITIALIZE (1h). Consequently, an AS = 1h also indicates that the Context Registers 6-7 need to provide the total length of IV, AAD, or textdata for this to be accomplished.
INITIALIZE/ FINALIZE	3h	Message is processed in multiple sessions and the current session is the last. The final MAC is computed.
UPDATE	0h	Message is processed in multiple sessions (descriptors) and the current session is not the last. The descriptor contains a non-final part of IV, AAD, textdata (IV, AAD or textdata split between descriptors). If decryption is requested, and data size is not written or is set to 0, and ICV_TEST bit is 1 - AS = UPDATE means that Check ICV (CICV) job is requested. The CICV-only job does not process any data, it just pops received ICV/MAC from the Input Data FIFO, and compares it to the computed MAC that is restored with the rest of the context from the previous session
FINALIZE	2h	Message is processed in a single session. MAC is computed.

- If the AS field is not set to either "Initialize/Finalize" or "Finalize" and the ICV_TEST bit is set to 1, the Illegal Mode error will be generated except for CICV-only jobs.

Proper AS field settings

Assume that a message has IV, AAD, and textdata and each of these types is split into two sessions (descriptors). The first IV descriptor should have AS set to "Update", the second IV Descriptor should have AS set to "Initialize", both AAD Descriptors and the first textdata descriptor should have AS field set to "Update", and the final Descriptor sets AS to "Initialize/Finalize".

- The Additional Algorithm Information (AAI) field must be set to 90h for GCM to be activated. Setting the DK bit causes an illegal-mode error.
- The Algorithm (ALG) field is used to activate AESA by setting it to 10h.

38.3.9.6 AES GCM mode use of the Context Register

The AES GCM mode uses the Context Register as follows:

- New message processing does not need any data provided in the context. All of the context data is written back by the GCM mode and needs to be restored before the next data chunk is to be processed in the multi-session processing. The final MAC is written in the context dwords 0-1.
- The initial counter value required for encryption/decryption is derived from IV and written to dwords 4-5. It is also required for the MAC computation.
- The incremented counter is placed in dwords 2-3 and is updated with every encrypted/decrypted block.
- Bit sizes of IV, AAD and textdata are required for GHASH computation and are accumulated in dwords 6-7 when multi-session processing is used.

Table 38-13. Context usage in GCM mode

Context DWord	Context-switching definition	Final-result definition
0	MAC[0:63]	MAC[0:63]
1	MAC[64:127]	MAC[64:127]
2	Yi[0:63]	-
3	Yi[64:127]	-
4	Y0[0:63]	-
5	Y0[64:127]	-
6	IV bit size (during GHASH of IV), AAD bit size (during message processing)	-
7	textdata bit size	-

38.3.9.7 AES GCM Mode use of the Data Size Register

The AES GCM mode uses the Data Size Register as follows:

- The byte-length of the message to be processed (including IV, AAD and textdata) must be written to the Data Size register (IV and AAD sizes must include padding to the 16 byte boundary).
- he first write to this register initiates processing. It can also be written during processing in which case the value written will be accumulated to the current state of the register.
- GCM decrements the value in this register with every processed block.
- Message splitting must be done only on a 16-byte boundary.

38.3.9.8 AES GCM mode use of the IV Size Register

The IV Size register is written with the number of bytes in the last IV block. If the total IV size is written, only the low 4 bits are registered. GCM needs this information to determine correct byte size of the IV used in the GHASH computation.

38.3.9.9 AES GCM mode use of the AAD Size Register

The AAD Size register is written with the number of bytes in the last AAD block. If the total AAD size is written, only the low 4 bits are registered. GCM needs this information to determine correct byte size of the AAD used in the GHASH computation.

38.3.9.10 AES GCM mode use of the ICV Size Register

The AES GCM mode uses the ICV Size Register as follows:

- This register is used to provide ICV/MAC byte-size.
- If the ICV mode bit is set, the ICV Size register also determines the number of bytes in the received ICV. Supported values for ICV size are 4 to 16 bytes. If this register is 0, ICV size will be 16 bytes.

38.3.9.11 AES GCM mode use of the Key Register

GCM key must be written to this register; it is always an encryption key.

38.3.9.12 AES GCM mode use of the Key Size Register

The AES GCM mode uses the Key Size Register as follows:

- The total number of key bytes must be written to the Key Size register.
- Any value other than 16, 24, or 32 causes key-size error to be generated.

38.3.9.13 AES GCM mode use of the ICV check

The AES GCM mode uses ICV checking as follows:

- Automatic ICV checking is enabled by setting ICV_TEST bit of the Mode Register to 1. When ICV is set to 1, the AS mode field must be set to either "Finalize" or "Initialize/Finalize"; otherwise the Illegal Mode error is generated except for CICV-only jobs. Also, if ICV = 1, the ENC bit must be 0.
- The received ICV must be provided on the input-data FIFO after the message data. The size of the received and computed ICV is for GCM written to the ICV Size register.
- If the ICV check detects mismatch between the decrypted received ICV and the computed ICV, the ICV error is generated.

38.4 Data encryption standard accelerator (DES) functionality

DES performs encryption and decryption on 64-bit values using the algorithm found in FIPS46-3. The DES module in LTC supports both single- and triple-DES functionality and ECB, CBC, CFB, and OFB modes as well as key parity checking in compliance with the DES specification.

38.4.1 DESA use of the Mode Register

The DESA uses the Mode Register as follows:

- The encryption field (ENC) controls whether DESA is encrypting or decrypting data.
- The Algorithm State (AS) field is not used to affect DESA functionality and should be set to zero at all times.
- The Additional Algorithm Information field (AAI) specifies the mode DESA runs. The supported modes are electronic code book (ECB), cipher block chaining (CBC), cipher feedback (CFB-8), and output feedback (OFB), described as follows:
  - ECB (0x20h) mode is a confidentiality mode that features, for a given key, the assignment of a fixed ciphertext block to each plaintext block (analogous to the assignment of code words in a codebook).

  - CBC (0x10h) mode is a confidentiality mode whose encryption process features the combining ("chaining") of the plaintext blocks with the previous ciphertext blocks. CBC mode requires an IV to combine with the first plaintext block. The IV does not need to be secret, but it must be unpredictable.
  - CFB (0x30h) mode is a confidentiality mode that features the feedback of successive ciphertext segments into the input blocks of the forward cipher to generate output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. The CFB mode requires an IV as the initial input-block.
  - OFB (0x40h) mode is a confidentiality mode that features the iteration of the forward cipher on an IV to generate a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. The OFB mode requires that the IV be unique for each execution of the mode under the given key.
- Key parity checking for DESA that checks for odd parity within each byte of the key is enabled with a value of (0x80h) in the AAI field.
- The algorithm field (ALG) must be programmed to DES (0x20h) or 3DES (0x21h).

38.4.2 DESA use of the Key Register

The DESA uses the Key Register as follows:

- The Key Register contains the 8-, 16-, or 24-byte key that is used during permutation in all DES modes.
- The DES specification defines the key as having odd parity in each byte.
- Key parity can be verified using the correct mode setting.

38.4.3 DESA use of the Key Size Register

DESA uses the Key Size Register as follows:

- Key size can be either 8, 16, or 24 bytes.
- A key size of 8 is valid only in single-DES mode.
- Values of 16 and 24 bytes can be used only in triple-DES mode.
- An illegal key size error is generated when in single-DES mode with a key size other than 8 or when in triple-DES mode with a key size other than 16 or 24.

38.4.4 DESA use of the Data Size Register

The DESA uses the Data Size Register as follows:

- The Data Size Register is written with the number of bytes of data to be processed.
- All DES modes except OFB expect to process data that is a multiple of 8 bytes and generates an error if the data size written is not an 8-byte multiple.
- This register must be written to start data processing.
- Because writing to the Data Size Register causes the written value to be added to the previous value in the register, the register may be written multiple times while data is being processed in order to increase the amount of input data that will be processed.

38.4.5 DESA Context Register

The DESA uses the Context Register as follows:

- For CBC, OFB, and CFB modes, the initialization vector is written to and read from the DESA Context Register.
- The value of this register changes as a result of the encryption process and reflects the context of DESA.
- DESA uses the first eight bytes of the Context Register to hold the beginning and final IV value for the CBC, OFB, and CFB modes.

38.4.6 Save and store operations in DESA context data

DESA is able to process data in chunks by saving the intermediate IV from the Context Register after each chunk of data and restoring the IV and key to the correct registers before processing any subsequent chunks of data.

38.5 Message digest hardware accelerator (MDHA) functionality

The MDHA performs hashing and authentication operations using the hashing algorithms defined in FIPS 180-3 (SHA-1, SHA-224, SHA-256) .

38.5.1 MDHA use of the Mode Register

The MDHA uses the Mode Register as follows:

- The Encryption field (ENC) and the Authenticate/Protect (AP) field are not used by the MDHA.
- The Algorithm field (ALG) must be programmed to SHA-1, SHA-224, or SHA-256.
- The Algorithm State (AS) field is defined as follows:

Table 38-14. Mode Register[AS] operation selections in MDHA

Operation	Description
INIT	The hashing algorithm is initialized with the chaining variables and then hashing begins. Input data must be a non-zero multiple of 64-byte blocks for SHA-1, SHA-224, SHA-256.
INIT/FINALIZE	The hashing algorithm is initialized with the chaining variables, and padding is automatically put on the final block of data. Any size of data is supported.
UPDATE	The hashing algorithm begins hashing with an intermediate context and running message length. Input data must be a multiple of 64-byte blocks for SHA-1, SHA-224, SHA-256.
FINALIZE	The hashing algorithms begin hashing with an intermediate context and running message length. Padding is performed on the final block of data. Any size of data is supported.

38.5.2 MDHA use of the Data Size Register

The MDHA uses the Data size Register as follows:

- The Data Size Register is written with the number of bytes of data to be processed.
- This register must be written to start data processing.
- This register may be written multiple times while data processing is in progress in order to add the amount written to the register to the previous value in the register.

38.5.3 MDHA use of the Context Register

The Context Register stores the current digest and running message length. The running message length will be 8 bytes immediately following the active digest. The digest size is defined as follows:

- SHA-1: 20 bytes
- SHA-224: 28 bytes final digest; 32 bytes running digest
- SHA-256: 32 bytes

38.5.4 Save and restore operations in MDHA context data

MDHA is able to process data in chunks by saving the intermediate context and running message length from the Context Register after each chunk of data and restoring the context and running message length to the Context Registers before processing any subsequent chunks of data.

38.6 Public-key hardware accelerator (PKHA) functionality

The PKHA module is capable of performing a number of different operations used in public-key cryptography, including modular arithmetic functions such as addition, subtraction, multiplication, exponentiation, reduction, and inversion, as well as elliptic-curve functions for point addition, point doubling, and point multiplication. All of these functions are provided in both integer and polynomial-binary field versions, except modular subtraction, which is the same as addition for binary polynomials. Most of these functions can be performed timing-equalized to thwart timing-related side-channel attacks. PKHA also includes a Miller-Rabin primality test function for detecting prime numbers.

The PKHA internally performs modular multiply operations using "Montgomery multiplication". For efficiency, many of these functions have a variant which allows either inputs or outputs in Montgomery form. Some have variants to supply the Montgomery conversion factor. These save time over the variations without. Internally, the PKHA operates on digits of these values. Different versions of the PKHA may have a different digit size. This PKHA has a digit size of 16 bits. This has implications for the inputs and outputs of certain functions. See the discussion on Montgomery arithmetic.

Because the numbers used in public-key cryptography are typically quite large and often referenced many times during a funciton, the inputs to PKHA are loaded into registers. PKHA has four of these labeled A, B, E, and N. A and B are for operands and results. E is for "keys", and N holds the modulus. For ECC functions, A and B are divided up into equal-size quadrants to accommodate the greater number of inputs required.

PKHA also has two other types of functions for manipulating the data in the registers. These are the Clear Memory and Copy Memory functions. The Clear Memory function allows all or any combination of the registers to be overwritten with zeros. The Copy Memory functions can be used to copy data from any of the A, B or N registers or register quadrant to any register A, B, E or N.

PKHA requires that all data for a given function be loaded before the Mode Register is written to invoke a function. This convention indicates to PKHA that all needed data has been loaded, and the function can now be launched. The required data includes:

- PKHA A, B, E and N registers, as required per the function to be executed.
- PKHA A Size, PKHA B Size, PKHA E Size and PKHA N Size, as required.

Note that the PKHA A, B, E and N registers can be accessed only when the Mode Register is not programmed to any PKHA operation. Once the Mode Register contains any PKHA operation value, writes to the PKHA registers will be ignored, and reads will return zero data. After the PKHA operation is complete, the results can be read from the PKHA registers only after the Mode Register is cleared.

38.6.1 PKHA MODE: clear memory function

Table 38-15. PKHA Mode register format for clear memory function

19	18	17	16	11-10	9	8	7	6	5-0
Aram	Bram	Eram	Nram	Reserved	Q3	Q2	Q1	Q0	Function
PKHA_MODE_MS				*PKHA_MODE_LS*					

If the Function field in PKHA MODE specifies the clear memory function, PKHA expects to be in the format shown in Table 38-15. The PKHA RAMs to be cleared may be selected in any combination. Selecting one or more Quadrants for clearing will cause only the specified quadrants (of the specified RAMs) to be cleared. If no Quadrants are selected, then the whole RAM will be cleared.

Table 38-16. PKHA mode register field descriptions for clear memory function

Bits	Description
19	Aram This bit selects the A RAM for zeroization. 0 : A not selected 1 : A selected.
18	Bram This bit selects the B RAM for zeroization. 0 : B not selected 1 : B selected
17	Eram This bit selects the E RAM for zeroization. 0 : E not selected 1 : E selected

Table continues on the next page...

Table 38-16. PKHA mode register field descriptions for clear memory function (continued)

Bits	Description
16	Nram This bit selects the N RAM for zeroization. 0 : N not selected 1 : N selected
11-10	Reserved
9	Quadrant 3 This bit selects the Quadrant 3 RAM for zeroization. 0 : not selected 1 : selected. Clearing will be only specified quadrant(s). Not valid if E RAM is selected.
8	Quadrant 2 This bit selects the Quadrant 2 RAM for zeroization. 0 : not selected 1 : selected. Clearing will be only specified quadrant(s). Not valid if E RAM is selected.
7	Quadrant 1 This bit selects the Quadrant 1 RAM for zeroization. 0 : not selected 1 : selected. Clearing will be only specified quadrant(s). Not valid is E RAM is selected.
6	Quadrant 0 This bit selects the Quadrant 0 RAM for zeroization. 0 : not selected 1 : selected. Clearing will be only specified quadrant(s). Not valid if E RAM is selected.
5-0	Function The Function value for clearmemory is 000001.

38.6.2 PKHA MODE: Arithmetic Functions

Table 38-17. PKHA Mode Register Format for Arithmetic Functions

19	18	17	16	11	10	9-8	7-6	5-0
inM	outM	F2m	R2	Reserved	Teq	OutSel	Reserved	Function
PKHA_MODE_MS				*PKHA_MODE_LS*				

Table 38-18. PKHA Mode register, format for arithmetic operation

Bits	Description
19 inM	Inputs in Montgomery Format. Indicates whether the inputs are in Montgomery format. If inM=0 : Normal value representation

Table continues on the next page...

Table 38-18. PKHA Mode register, format for arithmetic operation (continued)

Bits	Description
	If inM=1 : Montgomery format. (Not valid for all functions.)
18 outM	Outputs in Montgomery format. Indicates whether the outputs are to be left in Montgomery format or converted to normal values. If outM=0 : Normal value representation If outM=1 : Montgomery format. (Not valid for all functions.)
17 F2m	F2m. Indicates whether to use integer or binary polynomial arithmetic in executing the function. If F2m=0 : Integer If F2m=1 : Binary polynomial. (Not valid for all functions.)
16 R2	(R2 mod N). Indicates whether the term (R2 mod N) must be supplied as an input or will be calculated by the routine. If R2=0 : (R^2 mod N) is calculated and applied, if needed If R2=1 : (R^2 mod N) is an input. (Not valid for all functions.)
11 Reserved	Reserved
10 Teq	Timing Equalized. Indicates that a timing equalized version of the function should be executed. If Teq=0 : No timing equalization If Teq=1 : Timing equalization. (Not valid for all functions.)
9-8 OutSel	Output destination select. Indicates which memory should contain the output of the selected function. If OutSel=00b : B If OutSel=01b : A If OutSel=10b : Reserved If OutSel=11b : Reserved
7-6	Reserved
5-0 Function	Function. Indicates which arithmetic function to execute. If Function=000010b : Modular Addition (A + B) mod N If Function=000011b : Modular Subtraction 1 (A - B) mod N If Function=000100b : Modular Subtraction 2 (B - A) mod N If Function=000101b : Modular Multiplication (A x B) mod N If Function=000110b : Modular Exponentiation A^E mod N If Function=000111b : Modular Reduction A mod N If Function=001000b : Modular Inversion A^{-1} mod N If Function=001001b : ECC Point Add (P1 + P2) If Function=001010b : ECC Point Double (P2 + P2) If Function=001011b : ECC Point Multiply (E x P1) If Function=001110 : Greatest Common Divisor GCD(A,N)-see note below If Function=001111 : Miller-Rabin Primality Test -see note below

Table 38-18. PKHA Mode register, format for arithmetic operation

Bits	Description
	All other values for this field are currently reserved. **NOTE:** When using the GCD function or any ECC function, a divide-by-zero error occurs if the value of the most significant digit of N is all zeros. **NOTE:** When using the Miller-Rabin primality test function, if the most-significant digit of N is all zeros, the result is composite regardless of the value of N.

NOTE

Note that the arithmetic functions with outputs going to the A RAM are identical to those with outputs going to the B RAM. The only difference is the output destination.

Table 38-19. List of mode values for PKHA Integer Arithmetic Functions

Function name	Brief description	Output reg	Teq	Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex)	Detailed description
MOD_ADD	Integer Modular Addition	B	0	00002	Integer modular addition (MOD_ADD) function
		A	0	00102	
MOD_SUB_1	Integer modular subtraction (A - B)	B	0	00003	Integer Modular Subtraction (MOD_SUB_1) function
		A	0	00103	
MOD_SUB_2	Integer modular subtraction (B - A)	B	0	00004	Integer Modular Subtraction (MOD_SUB_2) function
		A	0	00104	
MOD_MUL MOD_MUL_TEQ	Integer modular multiplication	B	0	00005	Integer Modular Multiplication (MOD_MUL)
		A	0	00105	
	Timing equalized version	B	1	00405	
		A	1	00505	
MOD_MUL_IM MOD_MUL_IM_TEQ	Integer Modular Multiplication with Montgomery Inputs	B	0	80005	Integer Modular Multiplication with Montgomery Inputs (MOD_MUL_IM)
		A	0	80105	
	Timing equalized version	B	1	80405	
		A	1	80505	
MOD_MUL_IM_OM MOD_MUL_IM_OM_TEQ	Integer Modular Multiplication with Montgomery Inputs and Outputs	B	0	C0005	Integer Modular Multiplication with Montgomery Inputs and Outputs (MOD_MUL_IM_OM) Function
		A	0	C0105	
	Timinq equalized version	B	1	C0405	
		A	1	C0505	
MOD_EXP MOD_EXP_TEQ	Integer Modular Exponentiation	B	0	00006	Integer Modular Exponentiation (MOD_EXP)
		A	0	00106	
	Timing equalized version	B	1	00406	

Table continues on the next page...

Table 38-19. List of mode values for PKHA Integer Arithmetic Functions (continued)

Function name	Brief description	Output reg	Teq	Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex)	Detailed description
		A	1	00506	
MOD_EXP_IM MOD_EXP_IM_TEQ	Integer Modular Exponentiation with Montgomery Inputs	B	0	80006	Integer Modular Exponentiation with Montgomery Input (MOD_EXP_IM) Function
		A	0	80106	
	Timing equalized version	B	1	80406	
		A	1	80506	
MOD_AMODN	Integer Modular Reduction	B	0	00007	Integer Modulo Reduction (MOD_AMODN)
		A	0	00107	
MOD_INV	Integer Modular Inversion	B	0	00008	Integer Modular Inversion (MOD_INV)
		A	0	00108	
MOD_R2	Integer R^2 mod N	B	0	0000C	Integer Montgomery factor computation (MOD_R2)
		A	0	0010C	
MOD_GCD	Integer Greatest Common Divisor	B	0	0000E	Integer Greatest Common Divisor (MOD_GCD)
		A	0	0010E	
PRIME_TEST	Miller_Rabin primality test	B	0	0000F	Miller_Rabin primality test (PRIME_TEST)
		A	0	0010F	

1. PKHA_MODE_MS concatenated with 0h concatenated with PKHA_MODE_LS

Arithmetic functions on a binary polynomials (characterestic two) (F2M). All operate in polynomial basis.

Table 38-20. List of mode values for PKHA Binary Polynomial Arithmetic Functions

Function name	Brief description	Output reg	Teq	Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex)	Detailed description
F2M_ADD	Binary Polynomial Modular Addition	B	0	20002	Binary Polynomial (F_{2m}) addition (F2M_ADD) function
		A	0	20102	
F2M_MUL F2M_MUL_TEQ	Binary Polynomial Modular Multiplication	B	0	20005	Binary Polynomial (F2m) Modular Multiplication (F2M_MUL)
		A	0	20105	
	Timing equalized version	B	1	20405	
		A	1	20505	
F2M_MUL_IM F2M_MUL_IM_TEQ	Binary Polynomial Modular Multiplication with Montgomery Inputs	B	0	A0005	Binary Polynomial (F_{2m}) modular multiplication with Montgomery inputs (F2M_MUL_IM) Function
		A	0	A0105	
	Timing equalized version"	B	1	A0405	
		A	1	A0505	

Table continues on the next page...

Table 38-20. List of mode values for PKHA Binary Polynomial Arithmetic Functions (continued)

Function name	Brief description	Output reg	Teq	Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex)	Detailed description
F2M_MUL_IM_OM F2M_MUL_IM_OM_TEQ	Binary Polynomial Modular Multiplication with Montgomery Inputs and Output	B	0	E0005	Binary Polynomial (F_{2^m}) Modular Multiplication with Montgomery Inputs And outputs (F2M_MUL_IM_OM) Function
		A	0	E0105	
	Timing equalized version	B	1	E0405	
		A	1	E0505	
F2M_EXP F2M_EXP_TEQ	Binary Polynomial Modular Exponentiation	B	0	20006	Binary Polynomial (F_{2m}) Modular Exponentiation (F2M_EXP)
		A	0	20106	
	Timing equalized version	B	1	20406	
		A	1	20506	
F2M_AMODN	Binary Polynomial Modular Reduction	B	0	20007	Binary Polynomial (F_{2m}) modulo reduction (F2M_AMODN)
		A	0	20107	
F2M_INV	Binary Polynomial Modular Inversion	B	0	20008	Binary Polynomial (F_{2m}) modular inversion (F2M_INV)
		A	0	20108	
F2M_R2	Binary Polynomial R^2 mod n	B	0	2000C	Binary Polynomial (F_2m) R^2 mod N (F2M_R2) Function
		A	0	2010C	
F2M_GCD	Binary Polynomial Greatest Common Divisor	B	0	2000E	Binary Polynomial (F_{2m}) Greatest Common Divisor (F2M_GCD) Function
		A	0	2010E	

1. PKHA_MODE_MS concatenated with 0h concatenated with PKHA_MODE_LS

Elliptic Curve Functions over a prime field (ECC_MOD), where prime p > 3.

Table 38-21. List of mode values for Prime Field (F_p) Elliptic Curve Arithmetic Functions

Function name	Brief description	Output reg	Teq	Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex)	Detailed description
ECC_MOD_ADD	ECC prime field point add - affine coordinates	B	0	00009	ECC F_p Point Add - Affine Coordinates (ECC_MOD_ADD) Function
		A	0	00109	

Table continues on the next page...

Table 38-21. List of mode values for Prime Field (F_p) Elliptic Curve Arithmetic Functions (continued)

Function name	Brief description	Output reg	Teq	Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex)	Detailed description
ECC_MOD_ADD_R2	ECC prime field point add - affine coordinates, R2 input	B	0	10009	ECC F_p Point Add - Affine Coordinates, R2modN Input, (ECC_MOD_ADD_R2) Function
		A	0	10109	
ECC_MOD_DBL	ECC prime field point double - affine coordinates	B	0	0000A	ECC F_p Point Double - Affine Coordinates (ECC_MOD_DBL) Function
		A	0	0010A	
ECC_MOD_MUL ECC_MOD_MUL_TEQ	ECC prime field point multiply - affine coordinates	B	0	0000B	ECC F_p Point Multiply - Affine Coordinates (ECC_MOD_MUL) Function
		A	0	0010B	
	Timing equalized version	B	1	0040B	
		A	1	0050B	
ECC_MOD_MUL_R2 ECC_MOD_MUL_R2_TEQ	ECC prime field point multiply - affine coordinates, r2 mod n input	B	0	1000B	ECC F_p Point Multiply with R2modN Input - Affine Coordinates (ECC_MOD_MUL_R2) Function
		A	0	1010B	
	Timing equalized version	B	1	1040B	
		A	1	1050B	

1. PKHA_MODE_MS concatenated with 0h concatenated with PKHA_MODE_LS

Elliptic Curve Functions over a binary field (ECC_F2M). All operate in polynomial basis.

Table 38-22. List of mode values for Binary Field (F_{2^m}) Elliptic Curve Arithmetic Functions

Function name	Brief description	Output reg	Teq	Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex)	Detailed description
ECC_F2M_ADD	ECC binary field point add - affine coordinates	B	0	20009	ECC F_{2^m} Point Add - Affine Coordinates (ECC_F2M_ADD) Function
		A	0	20109	

Table continues on the next page...

Table 38-22. List of mode values for Binary Field (F_{2^m}) Elliptic Curve Arithmetic Functions (continued)

Function name	Brief description	Output reg	Teq	Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex)	Detailed description
ECC_F2M_ADD_R2	ECC binary field point add - affine coordinates, R2 input	B	0	30009	ECC F_{2^m} Point Add - Affine Coordinates, R2modN Input (ECC_F2M_ADD_R2) Function
		A	0	30109	
ECC_F2M_DBL	ECC binary field point double - affine coordinates	B	0	2000A	ECC F_{2^m} Point Double - Affine Coordinates (ECC_F2M_DBL) Function
		A	0	2010A	
ECC_F2M_MUL ECC_F2M_MUL_TEQ	ECC binary field point multiply - affine coordinates	B	0	2000B	ECC F_{2^m} Point Multiply - Affine Coordinates (ECC_F2M_MUL) Function
		A	0	2010B	
	Timing equalized version	B	1	2040B	
		A	1	2050B	
ECC_F2M_MUL_R2 ECC_F2M_MUL_R2_TEQ	ECC binary field point multiply - affine coordinates, r2 mod n input	B	0	3000B	ECC F_{2^m} Point Multiply with R2modN Input- Affine Coordinates (ECC_F2M_MUL_R2) Function
		A	0	3010B	
	Timing equalized version	B	1	3040B	
		A	1	3050B	

1. PKHA_MODE_MS concatenated with 0h concatenated with PKHA_MODE_LS

These functions are grouped here because they do not fall into one of the previous categories of PKHA functions.

38.6.3 PKHA MODE: copy memory functions

Table 38-23. PKHA Mode register, format for copy memory functions

19-17	16	11-10	9-8	7-6	5-0
Source Register	Destination Register		Source Segment	Destination Segment	Function
PKHA_MODE_MS	*PKHA_MODE_LS*				

Table 38-24. PKHA Mode register, field descriptions for copy memory functions

Bits	Description
19-17 Source Register	Source Register. Specifies the register to be copied from. If Source Register=000 : A If Source Register=001 : B If Source Register=011 : N All other values are currently reserved.
16 Destination Register 11-10	Destination Register. Specifies the register to be copied to. If Destination Register=000 : A If Destination Register=001 : B If Destination Register=010 : E If Destination Register=011 : N All other values are currently reserved. **NOTE:** The source register and destination register fields must not be the same.
9-8 Source Segment	Source Segment. Used when copying a register segment to specify which segment in the source register to copy from. If Source Segment=00 : Segment 0 If Source Segment=01 : Segment 1 If Source Segment=10 : Segment 2 If Source Segment=11 : Segment 3 **NOTE:**
7-6 Destination Segment	Destination Segment. Used when copying a register segment to specify which segment in the Destination Register to copy to. If Destination Segment=00 : Segment 0 If Destination Segment=01 : Segment 1 If Destination Segment=10 : Segment 2 If Destination Segment=11 : Segment 3 **NOTE:** These bits must be zero when E is the destination register.
5-0 Function	Function. Indicates which copy function to execute. If Function=010000 : Copy Memory N-Size (copies the same number of words as are in the modulus.) If Function=010001 : Copy Memory SRC-Size (copies the number of words specified in the source's size register)

This table gives the encodings from memory-to-memory copy. The top encoding in each cell is for Copy memory, N-Size (COPY_NSZ), the second is for Copy memory, source-size (COPY_SSZ)

The encoding is bits 19-0, including PKHA_MODE (i.e. PKHA_MODE_MS concatenated with 0h concatenated with PKHA_MODE_LS) and reserved bits. (Hex)

Table 38-25. Mode values for PKHA copy memory functions

	Destination Memory			
Source Memory	**A**	**B**	**N**	**E**
A		00410 00411	00C10 00C11	00810 00811
B	20010 20011		20C10 20C11	20810 20811
N	60010 60011	60410 60411		60810 60811

This table gives the encodings from memory-to-memory copy when segments are involved. The top encoding in each cell is for Copy memory, N-Size (COPY_NSZ), the second is for Copy memory, source-size (COPY_SSZ)

The encoding is bits 19-0, including PKHA_MODE and reserved bits [1] (Hex)

Table 38-26. Mode values for PKHA copy memory by segment functions

	Destination Quadrant											
Source Quadrant	**A0**	**A1**	**A2**	**A3**	**B0**	**B1**	**B2**	**B3**	**N0**	**N1**	**N2**	**N3**
A0					00410 00411	00450 00451	00490 00491	004D0 004D1	00C10 00C11	00C50 00C51	00C90 00C91	00CD0 00CD1
A1					00510 00511	00550 00551	00590 00591	005D0 005D1	00D10 00D11	00D50 00D51	00D90 00D91	00DD0 00DD1
A2					00610 00611	00650 00651	00690 00691	006D0 006D1	00E10 00E11	00E50 00E51	00E90 00E91	00ED0 00ED1
A3					00710 00711	00750 00751	00790 00791	007D0 007D1	00F10 00F11	00F50 00F51	00F90 00F91	00FD0 00FD1
B0	20010 20011	20050 20051	20090 20091	200D0 200D1					20C10 20C11	20C50 20C51	20C90 20C91	20CD0 20CD1
B1	20110 20111	20150 20151	20190 20191	201D0 201D1					20D10 20D11	20D50 20D51	20D90 20D91	20DD0 20DD1
B2	20210 20211	20250 20251	20290 20291	202D0 202D1					20E10 20E11	20E50 20E51	20E90 20E91	20ED0 20ED1
B3	20310 20311	20350 20351	20390 20391	203D0 203D1					20F10 20F11	20F50 20F51	20F90 20F91	20FD0 20FD1
N0	60010 60011	60050 60051	60090 60091	600D0 600D1	60410 60411	60450 60451	60490 60491	604D0 604D1				
N1	60110 60111	60150 60151	60190 60191	601D0 601D1	60510 60511	60550 60551	60590 60591	605D0 605D1				
N2	60210 60211	60250 60251	60290 60291	602D0 602D1	60610 60611	60650 60651	60690 60691	606D0 606D1				

Table continues on the next page...

Table 38-26. Mode values for PKHA copy memory by segment functions (continued)

	Destination Quadrant											
Source Quadrant	A0	A1	A2	A3	B0	B1	B2	B3	N0	N1	N2	N3
N3	60310 60311	60350 60351	60390 60391	603D0 603D1	60710 60711	60750 60751	60790 60791	607D0 607D1				

1. PKHA_MODE_MS concatenated with 0h concatenated with PKHA_MODE_LS

38.6.4 Modular math

Almost all math operations require with a modulus value in the N Memory. Math operations involving multiplication (multiplication, exponentiation, prime test, and ECC functions) are performed internally using Montgomery values.

38.6.5 About Montgomery values

The PKHA contains a Modular Arithmetic Unit. Multiplication is always modular multiplication:

$A * B \bmod N$.

The PKHA performs this computation with a Montgomery multiplier. A Montgomery multiplier can be more efficient than a multiply-then-reduce calculation because the modular reduction is done as part of the multiplication and the working product never gets larger than the modulus. In a normal multiplication, the product, before reduction, would be the size of the sum of the factors, so usually twice the size of the modulus. The factors in a Montgomery multiplication each have an R factor, and, as part of the multiplication and modular reduction, one R is removed. Thus, the computation performed is:

$(AR * BR) / R \bmod N$.

The equivalent of "division by R" occurs even if one of the inputs does not have an R factor.

A number of PKHA functions accept inputs in Montgomery form instead of normal values. Some instead take $R^2 \bmod N$ as an input. These functions can be faster than their normal-value alternatives if several operations are performed in a row or if these values are known in advance. This is because, before being used, ($R^2 \bmod N$ needs to be computed and) normal values need to be converted internally to Montgomery form.

The Montgomery form of a value is *value*R* mod *N*, referred to here as *value*. The term $R = 2^{SD}$ is the Montgomery factor, where D is the digit size (of a digit in the PKHA arithmetic unit), in bits, and S is the minimum number of digits needed to hold the value in N. R is therefore dependent on N and D.

To use the PKHA to convert a normal value to a Montgomery value, one must first compute (or know) R^2 mod *N*, the Montgomery Conversion Factor. The following steps can be used to convert a value from a normal value into its Montgomery form (A and B inputs may be reversed):

R^2 = MOD_R2(*N*)

A = MOD_MUL_IM_OM(*A*, B=R^2, *N*)

The equivalent F2M function can be used for binary polynomial values.

Eventually, the value needs to be converted out of Montgomery form. This can be done by performing another multiply (R^2 is not needed for this).

```
A = MOD_MUL_IM_OM(A=A, B=1, N)
```

Another method is to cause the PKHA to perform a multiplication and conversion to normal form. Internally, there are two multiplications: first the two inputs, then the product by one.

```
AB mod N = MOD_MUL_IM(A=A, B=B, N)
```

A third method is to have just one factor (either one) in Montgomery form:

```
AB mod N = MOD_MUL_IM_OM(A=A, B=B, N)
```

The following operations can be used to convert a value from a normal value into its Montgomery form (A and B inputs may be reversed):

The equivalent F2M functions can be used for binary polynomial values.

It is possible to add and subtract Montgomery values, if *R* mod *N* is the same. Do not mix and match Montgomery and normal values for addition or subtraction. 5 + 3R => *5/R + 3R* or *5/R + 3*; neither is likely the desired result.

38.6.6 Non-modular Math

Although addition, subtraction, and multiplication functions require a modulus, it is possible to perform these calculations without any reduction: the modulus must be larger than the expected result.

For addition and subtraction, this is easily done. For multiplication, the MOD_MUL function may be used, but it is not the most efficient, as internally first R^2 mod N will be computed, then two multiplications will be performed (first to convert one factor into Montgomery, then to compute the product, not in Montgomery).

For non-modular multiplication, MUL_IM_OM is much more efficient, as only one multiplication will be performed. This can be used if the factors are not in Montgomery form, i.e., if the product to be calculated is *A*B* instead of *A*B mod N*. Since the multiplier always "divides by *R*', a special modulus value in Nram is required which will make *R* have the value 1. This is done by creating a modulus *N = R-1* so that *R mod N* will have the value one. This way, normal values are the same as Montgomery values; no conversion is necessary and the multiplier will quietly "divide by one" to no effect.

As an example, on a PKHA with a digit size of 32 bits and a product which will be no more than six bytes long, $R = 2^{SD} = 2^{2*32} = 2^{64}$. Therefore the modulus must be 0xFFFFFFFFFFFFFFFF.

For computation with binary polynomials, the equivalent F2M functions may be used.

38.6.7 Elliptic-Curve Math

The PKHA provides point math operations on elliptic curves. These include the ability to add two points (+ operator), double a point, and multiply a point by an integer (scalar) value (x operator).

The input points are assumed to be valid points on the curve. If non-point coordinates or invalid curve parameters are used an input, then a non-point set of coordinates are likely to be returned as output.

The same minimum-value restriction for the modulus exists for Elliptic Curve math as exist for Modular operations. The maximum modulus is 512 bits (64 bytes) in length, or one quadrant.

If xx is the Point at Infinity, *P*and *Q* are points on the curve, and *j* and *k* are integers, then the following identities, as well as others easily derived by taking advantage of associative and commutive properites, hold:

- $P + Q = Q + P$
- $P = xx + P$
- $xx = 0 \times P$
- $(j + k)P = (j \times P) + (k \times P)$

There may be times when the negative of a point is necessary:

- When subtracting points $P_A - P_B$
- When multiplying by a negative integer: -abs(k) x P_A

To subtract, one can negate the second term and perform an addition, i.e.

$P_C = P_A - P_B = P_A + (-P_{B)}$

When multiplying by a negative value, one can either negate the starting point or the ending point. The multiplication value is the absolute value of the scalar, i.e., when *k* is negative

$P_C = k \times P_A = -abs(k) \times P_A = abs(k) \times (-1P_{A)} = -(abs(k) \times P_A)$

38.6.7.1 Point math over a prime field (F_p)

The ECC_MOD family of functions perform Add, Double, and scalar Multiply operations on points on a curve defined by the equation:

$E: y^2 = x^3 + ax + b \bmod p$

where p is the a prime integer > 3. These operations are available in Affine Coordinates (*x,y*).

The modulus (value in N memory) for these operations is *p*, also referred to as *q*.

The equality for the negative of a point *P*, in affine coordinates, is $-P = -(x,y) = (x, -y)$

The operations will not provide useful outputs if the inputs are not valid points on the curve, i.e., if they are not solutions to the curve equation E.

38.6.7.2 Point math over a binary field (F_{2^m})

The ECC_F2M family of functions perform add, double, and scalar multiply operations on points on a curve defined by the equation:

E: $y^2 + xy = x^3 + ax^2 + b$

These operations are available in Affine Coordinates (*x,y*). All inputs and output values of polynomial values are in polynomial basis. For example, x^5+x+1 is represented as 23h

The modulus (value in N memory) for these functions is *q*, the field-defining irreducible polynomial for the curve. Other documents use other symbols, including *p(t)*, *f(t)*, and *f*.

The equality for the negative of a point, in affine coordinates, is $-P = -(x,y) = (x, x+y)$.

The operations will not provide useful outputs if the inputs are not valid points on the curve, i.e., if they are not solutions to the curve equation E.

Because of the way the point operations are performed over a binary field, these functions require as an input the value *c* rather than *b*. The relationship between these two values is:

$b = c^4 \bmod q$

and

$c = b^{2^{m-2}} \bmod q$, where *m* is the degree (the power of its highest-power term) of *q*.

This *c* value is referred to as *b'* in the ECC Public-Key protocols for ECDSA sign, and so on. The calculation of *c* is expensive, so it is obviously an advantage to calculate it only once or have it precomputed. See Special values for common ECC domains for these values for common ECC domains.

38.6.7.3 About the Point at Infinity

The point at infinity is a possible result for point math operations. Knowing its representation is important for programming, though the PIZ bit in Operation Status Register can be used to determine when the result of an operation is the point at infinity.

The representation of the point at infinity, in affine coordinates, depends upon the type of curve and the value of the *b* term of the curve's equation:

- For F_p curves, where *b* is equal to 0: (0, 1)
- For F_p and F_{2^m} curves, where *b* is not equal to 0: (0, 0)

38.6.8 PKHA Mode Register

The following tables list the valid PKHA_MODE values for all PKHA functions:

PKHA Clear Memory Functions: Table 38-28

PKHA Modular Arithmetic functions: Table 38-19

PKHA Copy Memory functions: Table 38-25

NOTE

Use of any PKHA_MODE value not listed in these tables results in an invalid mode error.

38.6.9 PKHA functions

The various PKHA functions are described in the following subsections. The following information applies to all PKHA functions.

- Mode Register bits that may be either 1's or 0's for the given function are represented with *x*.
- For convenience, in all the descriptions below the output is shown as the default B, although the actual output destination can be specified via the Class 1 Mode Register[OutSel] field to be either the B RAM or the A RAM.
- For each PKHA function, the specified mode bits are in the Class 1 Mode Register[PKHA_MODE_LS] field.
- For all of the PKHA functions, the Class 1 Mode Register[PKHA_MODE_MS] field is set to 8h.
- The descriptions specify the output register(s) and any other registers that might be modified. Note that the default output register is still modified but the output is placed into the specified destination register(s).
- Note that any parameter underlined is in Montgomery form (for example, $\underline{A}$ = AR mod N).
- Errors reported by PKHA are written to the Job Ring Output Status Register and termination status word (Job termination status/error codes). They are encoded in the ERRID field.
- Three flags in the CCB Status Register may be set by PKHA: PIZ, PIO, and PRM. These flags can be tested by the JUMP (HALT) command They are
  - PIZ is set to indicate that PKHA generated a result equal to zero, or, in the case of ECC functions, the point at infinity.
  - PIO is set whenever a GCD routine finds that the Greatest Common Denominator of two numbers is the number 1. For other general non-ECC functions, it means that the result is equal to one. This may also be referred to as the GCD flag.
  - PRM is set by the PRIME_TEST routine if it finds that a candidate integer is probably prime (that is, passes the Miller-Rabin primality test).
- It is important to note that the PKHA mathematical functions work in terms of "digits"; that is, the arithmetic unit is pipelined to work on a digit of data at a time. For PKHA-32 a digit = 32 bits (4 bytes) of data, for PKHA-64 a digit = 64 bits (8 bytes), and for PKHA-128 a digit = 128 bits (16 bytes). Therefore, the term 'digit' refers to 32, 64, or 128 bits of data in the input and/or output values used by the PKHA arithmetic unit.

38.6.9.1 Clear Memory (CLEAR_MEMORY) function

This function clears the specified registers or quadrants of registers in the PKHA. This includes the A, B, N and E. All registers or quadrants of registers are written with zeros.

A detailed description may be found in PKHA MODE: clear memory function.

Table 38-27. CLEAR_MEMORY function properties

Property	Notes
Mode value	*ABEN*_0000_00 *QQ_QQ* 00_0001, with the following restrictions on the combinations of ABEN and QQQQ: At least one of ABEN must be on. If E is on, all Q must be zero. Some example encodings are in the table below.
Input	None
Output	A = 0, B = 0, E = 0, N = 0, or some quadrant(s) thereof, as specified by ABEN and QQQQ. Each *Q* specifies a quadrant, in order from 3 through 0.
Requirements	The Mode Register specifies which registers or quadrants of registers to clear. If no quadrants are selected, then all quadrants of the specified register(s) are cleared.
Side effects	None
Errors reported	Invalid Mode, if no registers are selected, or E with one or more quadrants is selected
Flags set	None

Table 38-28. Example mode values for PKHA clear memory functions

Function name	Register selects				Quadrant selects				Brief description	Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex)
	A	B	E	N	3	2	1	0		
CLEAR_MEMORY	1	1	1	1	0	0	0	0	Clear registers A, B, E, N	F0001
	1	1	1	0	0	0	0	0	Clear registers A, B, E	E0001
	1	1	0	1	0	0	0	0	Clear registers A, B, N	D0001
	1	0	1	0	0	0	0	0	Clear registers A, E	A0001
	1	0	0	1	0	0	0	0	Clear registers A, N	90001
	0	1	0	1	0	0	0	0	Clear registers B, N	50001
	0	1	0	0	0	0	0	0	Clear register B	40001
	0	0	1	0	0	0	0	0	Clear register E	20001
	0	0	0	1	0	0	0	0	Clear register N	10001
	1	0	0	0	1	1	0	0	Clear quadrants 2 and 3 of register A	80301
	0	1	0	1	0	0	0	1	Clear quadrant 0 of registers B, N	50041
	1	1	0	0	1	0	0	0	Clear quadrant 3 of registers A, B	C0201

1. PKHA_MODE_MS concatenated with 0h concatenated with PKHA_MODE_LS

38.6.9.2 Integer modular addition (MOD_ADD) function

Table 38-29. MOD_ADD function properties

Property	Notes
Mode value	0000_0000_000*x*_0000_0010
Input	• N = modulus and data size, any integer • A = first addend, any integer less than N • B = second addend, any integer less than N
Output	B (or A, if selected) = (A + B) mod N
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes • A and B are each < N.
Side effects	None
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • A Size Error is set if the size of A is greater than size of N. • B Size Error is set if the size of B is greater than size of N. If (A + B) >= N, N will be subtracted just once from the sum. That is, if (A + B) >= 2N, then the result will not be mod N.
Flags set	PIZ is set if the result is zero. PIO is set if the result is one.

38.6.9.3 Integer Modular Subtraction (MOD_SUB_1) function

Modular subtraction can be described as follows. If A ≥ B or A = B = 0, then B = A - B. Otherwise, if A < B, then B = A + N - B. The result is always positive and less than N.

Table 38-30. MOD_SUB_1 function properties

Property	Notes
Mode value	0000_0000_000*x*_0000_0011
Input	• N = modulus, any integer • A = minuend, any integer less than N • B = subtrahend, any integer less than N
Output	B (or A, if selected) = (A - B) mod N
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes • A and B are less than N.
Side effects	None
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • A Size Error is set if the size of A is greater than size of N. • B Size Error is set if the size of B is greater than size of N.
Flags set	PIZ is set if the result is zero. PIO is set if the result is one.

38.6.9.4 Integer Modular Subtraction (MOD_SUB_2) function

Table 38-31. MOD_SUB_2 function properties

Property	Notes
Mode value	0000_0000_000*x*_0000_0100
Input	• N = modulus, any integer • B = minuend, any integer less than or equal to N • A = subtrahend, any integer less than or equal to N
Output	B (or A, if selected) = (B - A) mod N
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes • A and B are < N
Side effects	None
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • A Size Error is set if the size of A is greater than size of N. • B Size Error is set if the size of B is greater than size of N.
Flags set	PIZ is set if the result is zero. PIO is set if the result is one.

38.6.9.5 Integer Modular Multiplication (MOD_MUL)

The (AB) mod N computation is provided to assist in algorithms and protocols where a single modular multiplication is required and not as a chaining of multiplications. In the latter case, Montgomery format multiplication routines (that is, MOD_MUL_IM or MOD_MUL_IM_OM) are more efficient. This function first computes R^2 mod N, then multiples one factor to produce AR, then multiplies AR*B to produce AB.

Table 38-32. MOD_MUL function properties

Property	Notes
Mode value	0000_0000_000*x*_0000_0101
Input	• N = modulus, any odd integer • A = multiplicand, any integer less than N • B = multiplier, any integer less than N
Output	B (or A, if selected) = (AxB) mod N
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes • A and B are < N.
Side effects	A and E are modified.
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • Modulus Even Error is set if N is even. • A Size Error is set if the size of A is greater than size of N.

Table continues on the next page...

Table 38-32. MOD_MUL function properties (continued)

Property	Notes
	• B Size Error is set if the size of B is greater than size of N. • Divide-By-Zero Error is set if the most significant digit of the modulus is all zeros.
Flags set	PIZ is set if the result is zero. PIO is set if the result is one.

38.6.9.6 Integer Modular Multiplication with Montgomery Inputs (MOD_MUL_IM)

This function takes its inputs, integers, in Montgomery form, multiplies them modulo the value in the N register and returns the result as a field value. To do this, it performs two multiplications: AR*BR => ABR and ABR*1 => AB.

Table 38-33. MOD_MUL_IM function properties

Property	Notes
Mode value	1000_0000_000*x*_0000_0101
Input	• N = modulus, any odd integer • A = multiplicand, a value in Montgomery format • B = multiplier, a value in Montgomery format
Output	B (or A if selected) = A x B mod N, the non-Montgomery product of the inputs
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes • A and B are less than modulus N
Side effects	None
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • Modulus Even Error is set if N is even. • A Size Error is set if the size of A is greater than size of N. • B Size Error is set if the size of B is greater than size of N.
Flags set	PIZ is set if the result is zero. PIO is set if the result is one.

38.6.9.7 Integer Modular Multiplication with Montgomery Inputs and Outputs (MOD_MUL_IM_OM) Function

This function performs the calculation A*B/R mod N, where R is the Montgomery factor for N. This can be used in several ways:

- If one value is a normal value, and the other is R2 mod N, then the result is the normal value converted to Montgomery.

- If A and B are both Montgomery values, then the result is the product of A and B as a Montgomery value.
- If only one of (A,B) is a Montgomery value, then the result is the product as a normal value.
- If one of (A,B) is a Montgomery value and the other is the value one, then the result is Montgomery value converted to a normal value.

Table 38-34. MOD_MUL_IM_OM function properties

Property	Notes
Mode value	1100_0000_000*x*_0000_0101
Input	• N = modulus, any odd integer • A = multiplicand, an integer 0 ≤ A < N • B = multiplier, an integer 0 ≤ B < N
Output	B (or A, if selected) = (AxB) mod N
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes
Side effects	None
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • Modulus Even Error is set if N is even. • A Size Error is set if the size of A is greater than size of N. • B Size Error is set if the size of B is greater than size of N.
Flags set	PIZ is set if the result is zero.

38.6.9.8 Integer Modular Exponentiation (MOD_EXP)

This function is commonly used to perform a single-step RSA operation. It computes R^2 mod N and converts A to Montgomery form before beginning the exponentiation.

Table 38-35. MOD_EXP function properties

Property	Notes
Mode value	0000_0000_000*x*_0000_0110
Input	• N = modulus, any odd integer • A = an integer 0 ≤ A < N • E = exponent, any integer
Output	B (or A, if selected) = (A^E) mod N, a an integer 0 ≤ A < N
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes • Maximum key (exponent) size = 256 bytes • A < N
Side effects	None
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • Modulus Even Error is set if N is even. • Key Size Error is set if size of E = 0 or size of E > 256. • A Size Error is set if the size of A is greater than size of N.

Table continues on the next page...

Table 38-35. MOD_EXP function properties (continued)

Property	Notes
Flags set	PIZ is set if the result is zero.

38.6.9.9 Integer Modular Exponentiation, Timing Equalized (MOD_EXP_TEQ) Function

MOD_EXP_TEQ performs the same operation as Integer Modular Exponentiation (MOD_EXP) but with an added timing equalization security feature. Its exponentiation run-time, for a given modulus and size of exponent, is constant.

38.6.9.10 Integer Modular Exponentiation with Montgomery Input (MOD_EXP_IM) Function

The input data (base) to be exponentiated must be provided in the Montgomery format. That is $\underline{A}$ = AR (mod N), where R is the Montgomery constant). The result will be returned in the output in normal integer (non-Montgomery) representation.

Table 38-36. MOD_EXP_IM function properties

Property	Notes
Mode value	1000_0000_000*x*_0000_0110
Input	• N = modulus, any odd integer • A = a value 0 ≤ A < N, in Montgomery format • E = exponent, any integer (normal integer representation)
Output	B (or A, if selected) = (A^E) mod N
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes • Maximum key (exponent) size = 256 bytes • A < N
Side effects	None
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • Modulus Even Error is set if N is even. • Key Size Error is set if size of E = 0 or size of E > 256. • A Size Error is set if the size of A is greater than size of N.
Flags set	PIZ is set if the result is zero. PIO is set if the result is one.

38.6.9.11 Integer Modular Exponentiation with Montgomery Input, Timing Equalized (MOD_EXP_IM_TEQ) Function

MOD_EXP_IM_TEQ performs the same operation as Integer Modular Exponentiation with Montgomery Input (MOD_EXP_IM) Function but with an added timing equalization security feature. Its computation run-time, for a given modulus and size of exponent, is constant.

38.6.9.12 Integer Modulo Reduction (MOD_AMODN)

A and N can be of any size and it is not required that A > N, but N must be non-zero. This function computes the remainder of A divided by N.

Table 38-37. MOD_AMODN function properties

Property	Notes
Mode value	0000_0000_000*x*_0000_0111
Input	• N = modulus, any non-zero integer • A = any integer
Output	B (or A, if selected) = A mod N, A reduced modulo N
Requirements	• N = non-zero value • Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes
Side effects	None
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • Divide By Zero Error is set if N = 0.
Flags set	PIZ is set if the result is zero. PIO is set if the result is one.

38.6.9.13 Integer Modular Inversion (MOD_INV)

If the modulus, N, is prime, then all values of A, 1 ≤ A < N, are guaranteed to have an inverse mod N. If N is not prime, A may or may not have an inverse. It will have one only if GCD(A, N) == 1.

Table 38-38. MOD_INV function properties

Property	Notes
Mode value	0000_0000_000*x*_0000_1000
Input	• N = modulus, any non-zero integer • A = any non-zero integer lass than N
Output	B (or A, if selected) = A^{-1} mod N, an integer, the multiplicative inverse of A

Table continues on the next page...

Table 38-38. MOD_INV function properties (continued)

Property	Notes
Requirements	• Neither A or N can be zero. • A must be less than N. • Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes
Side effects	A and E are modified.
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • A Size Error is set if the size of A is greater than size of N. • Divide-By-Zero Error is set if N or A = 0, or if the most significant digit of N = 0, or if there is no inverse.
Flags set	None

38.6.9.14 Integer Montgomery factor computation (MOD_R2)

This function is used to compute a constant to assist in converting operands into the Montgomery residue system representation. The constant R^2(mod N) is dependent upon the digit size of the PKHA and the value in N.

MUL, EXP, and ECC functions which do not have "IM" (Montgomery inputs) or an R2 input will internally invoke this routine to determine the constant and do the conversions before other operations.

If the modulus N is a protocol- or system-wide parameter that does not change frequently, such as in ECC operations for a specific curve, save this computed constant, because this routine takes a not-insignificant amount of time to complete.

Table 38-39. MOD_R2 function properties

Property	Notes
Mode value	0000_0000_000*x*_0000_1100
Input	N = modulus, any odd integer
Output	B (or A, if selected) = R2 mod N, where R = 2^{SD} where S is size of a digit in bits and D is the number of digits of N; in other words, D = ceiling [sizeof(N) in bits / S]
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes
Side effects	None
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • Modulus Even Error is set if N is even.
Flags set	None

38.6.9.15 Integer Greatest Common Divisor (MOD_GCD)

Table 38-40. MOD_GCD function properties

Property	Notes
Mode value	0000_0000_000*x*_0000_1110
Input	• N = any integer. The most-significant digit of N must be non-zero. • A = any integer less than or equal to N
Output	B (or A, if selected) = GCD(A,N), an integer less than or equal to A that divides both A and N
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes • A and N may not both be zero
Side effects	A is modified
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • A Size Error is set if the size of A is greater than size of N. • Divide-By-Zero Error is set if N or A = 0, or if the most significant digit of N = 0.
Flags set	PIO is set if the result is 1.

38.6.9.16 Miller_Rabin primality test (PRIME_TEST)

Table 38-41. PRIME_TEST function properties

Property	Notes
Mode value	0000_0000_000*x*_0000_1111
Input	• N[1] = Candidate prime integer • A = An initial random seed for the base value of exponentiation; can be any integer 2 < A < N - 2 • B = "t" parameter, which is the number of trial runs. By default, it is set at 1 or B[7:0], whichever is bigger. Only the lowest byte of the supplied value is used.
Output	B (or A, if selected) = 1 if candidate is believed to be prime, otherwise 0
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes • A and N may not both be zero
Side effects	N and A are modified
Errors reported	• N Size Error is set if size of N = 0 or size of N > 256. • B Size Error is set if B[7:5] is nonzero and N size > 128 • Divide-By-Zero Error is set if no seed can be found that is in the legal range of 2 < A < N-2. This occurs if N = 1 or N = 3
Flags set	PRM is set if the candidate is believed to be prime

1. If the most significant digit of N is zero, the result is always composite, the output is the value zero, and the PRM flag is not set, regardless of the primality of the value of N.

38.6.9.17 Binary Polynomial (F_{2m}) addition (F2M_ADD) function

This function performs binary polynomial modular addition without any modulo reduction, as the value in the N register is ignored. Only its size is used, to determine the size of the result.

This type of addition is the equivalent of a bitwise XOR and this function may be used for that purpose.

This function could as easily be labeled F2M_SUB, as it is mathematically equivalent.

Table 38-42. F2M_ADD function properties

Property	Notes
Mode value	0010_0000_000*x*_0000_0010
Input	• Size of N (modulus size) • A = first addend, a binary polynomial • B = second addend, a binary polynomial
Output	B (or A, if selected) = A xor B, a binary polynomial
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes • The N need not be written, because its contents are ignored, but the size of N must be written. This size is needed because inputs A and B are considered binary polynomials modulo some irreducible polynomial N.
Side effects	None
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • A Size Error is set if the size of A is greater than size of N. • B Size Error is set if the size of B is greater than size of N.
Flags set	PIZ is set if the result is zero. PIO is set if the result is one.

38.6.9.18 Binary Polynomial (F2m) Modular Multiplication (F2M_MUL)

The (AB) mod N computation is provided to assist in algorithms and protocols where a single modular multiplication is required and not as a chaining of multiplications. In the latter case, Montgomery format multiplication routines (that is, F2M_MUL_IM or F2M_MUL_IM_OM) are more efficient. This function first computes R^2 mod N, then multiples $A*R^2$ to produce AR, then multiplies AR*B to produce AB.

Table 38-43. F2M_MUL function properties

Property	Notes
Mode value	0010_0000_000*x*_0000_0101
Input	• N = modulus, an irreducible polynomial • A = multiplicand, a field element • B = multiplier, a field element

Table continues on the next page...

Table 38-43. F2M_MUL function properties (continued)

Property	Notes
Output	B (or A, if selected) = (AB) mod N, a field element
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes • A and B are field elements modulo N.
Side effects	A and E are modified.
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • Modulus Even Error is set if N is even. • A Size Error is set if the size of A is greater than size of N. • B Size Error is set if the size of B is greater than size of N.
Flags set	PIZ is set if the result is zero. PIO is set if the result is one.

38.6.9.19 Binary Polynomial (F_{2m}) modular multiplication with Montgomery inputs (F2M_MUL_IM) Function

This function takes its inputs, binary polynomials, in Montgomery form, multiplies them modulo the value in the N register, used as a reduction polynomial, and returns the result as a field value. To do this, it performs two multiplications: AR*BR => ABR and ABR*1 => AB.

Table 38-44. F2M_MUL_IM function properties

Property	Notes
Mode value	1010_0000_000*n*_0000_0101
Input	• N = modulus, an "odd" binary polynomial of order m • A = multiplicand, a binary polynomial < 2^m, in Montgomery format • B = multiplicand, a binary polynomial < 2^m, in Montgomery format
Output	B (or A, if selected) = (AxB) mod N, a a binary polynomial < 2^m, non-Montgomery format
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes • A and B are field elements in Montgomery format and must be modulo reduced by irreducible polynomial N.
Side effects	None
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • Modulus Even Error is set if N is even. • A Size Error is set if the size of A is greater than size of N. • B Size Error is set if the size of B is greater than size of N.
Flags set	PIZ is set if the result is zero. PIO is set if the result is one.

38.6.9.20 Binary Polynomial (F_{2^m}) Modular Multiplication with Montgomery Inputs And outputs (F2M_MUL_IM_OM) Function

This function performs the calculation A*B/R mod N, where R is the Montgomery factor for N. This can be used in several ways:

- If one value is a normal value, and the other is R2 mod N, then the result is the normal value converted to Montgomery.
- If A and B are both Montgomery values, then the result is the product of A and B as a Montgomery value.
- If only one of (A,B) is a Montgomery value, then the result is the product as a normal value.
- If one of (A,B) is a Montgomery value and the other is the value one, then the result is Montgomery value converted to a normal value.

Table 38-45. F2M_MUL_IM_OM function properties

Property	Notes
Mode value	1110_0000_000*x*_0000_0101
Input	• N = modulus, an "odd" binary polynomial of order m • A = a binary polynomial 0 < 2^m • B = a binary polynomial 0 < 2^m
Output	B (or A, if selected) = (AxB) mod N, in Montgomery format
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes • A and B are field elements in Montgomery format.
Side effects	None
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • Modulus Even Error is set if N is even. • A Size Error is set if the size of A is greater than size of N. • B Size Error is set if the size of B is greater than size of N.
Flags set	PIZ is set if the result is zero. PIO is set if the result is one.

38.6.9.21 Binary Polynomial (F_{2m}) Modular Exponentiation (F2M_EXP)

This function is similar to MOD_EXP but works on binary polynomials. It is provided mainly to assist in the computation of elliptic curve parameter "c", where c = $b^{2^{m-2}} \bmod n$) given an elliptic curve parameter "b" and the field-defining polynomial in N. It computes R^2 mod N and converts A to Montgomery form before beginning the exponentiation.

Table 38-46. F2M_EXP function properties

Property	Notes
Mode value	0010_0000_000*x*_0000_0110
Input	• N = modulus, an "odd" binary polynomial of order m • A = a binary polynomial < 2^m • E = exponent, any integer
Output	B (or A, if selected) = (A^E) mod N, a binary polynomial
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes • Maximum key (exponent) size = 256 bytes
Side effects	None
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • Modulus Even Error is set if N is even. • Key Size Error is set if size of E = 0 or size of E > 256. • A Size Error is set if the size of A is greater than size of N.
Flags set	PIZ is set if the result is zero. PIO is set if the result is one.

38.6.9.22 Binary Polynomial (F_{2m}) Modular Exponentiation, Timing Equalized (F2M_EXP_TEQ) Function

F2M_EXP_TEQ performs the same operation as Binary Polynomial (F_{2m}) Modular Exponentiation (F2M_EXP) but with an added timing equalization security feature. Its exponentiation run-time, for a given modulus and size of exponent, is constant.

Table 38-47. F2M_EXP_TEQ function properties

Property	Notes
Mode value	0010_0000_010*x*_0000_0110
Input	• N = modulus, an "odd" binary polynomial of order m • A = a binary polynomial 0 ≤ A < N • E = exponent, any integer
Output	B (or A, if selected) = (A^E) mod N, a binary polynomial field element
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes • Maximum key (exponent) size = 256 bytes
Side effects	None
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • Modulus Even Error is set if N is even. • Key Size Error is set if size of E = 0 or size of E > 256. • A Size Error is set if the size of A is greater than size of N.
Flags set	PIZ is set if the result is zero. PIO is set if the result is one.

38.6.9.23 Binary Polynomial (F_{2m}) modulo reduction (F2M_AMODN)

A and N can be of any size, and it is not required that A > N, but N must be non-zero. This is the equivalent of the MOD_AMODN routine applied to binary polynomials.

Table 38-48. F2M_AMODN function properties

Property	Notes
Mode value	0010_0000_000*x*_0000_0111
Input	• N = modulus, any non-zero polynomial /> • A = any polynomial
Output	B (or A, if selected) = A mod N, a polynomial, binary element modulo N
Requirements	• N = non-zero value • Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes
Side effects	None
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • Divide By Zero Error is set if N = 0.
Flags set	PIZ is set if the result is zero. PIO is set if the result is one.

38.6.9.24 Binary Polynomial (F_{2m}) modular inversion (F2M_INV)

Table 38-49. F2M_INV function properties

Property	Notes
Mode value	0010_0000_000*x*_0000_1000
Input	• N = modulus, an irreducible polynomial • A = a field element
Output	B (or A, if selected) = A^{-1} mod N, a field element, the multiplicative inverse of A
Requirements	• A is an element of the binary polynomial field. • Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes
Side effects	A and E are modified.
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • Modulus Even Error is set if N is even. • A Size Error is set if the size of A is greater than size of N. • Divide-By-Zero Error is set if N or A = 0, or if the most significant digit of N = 0.
Flags set	None

38.6.9.25 Binary Polynomial (F_2m) R^2 mod N (F2M_R2) Function

This function is used to compute the Montgomery Conversion Factor, which is used to convert operands into the Montgomery residue system representation. The constant R^2(mod N) is dependent upon the digit size of the PKHA and the value of N. If this value is not available, then this routine (function) is called to determine the constant before other operations. If N contains a protocol- or system-wide parameter that does not change frequently, such as in ECC operations for a specific curve, save this computed constant, because this routine takes a considerable amount of time to complete.

Table 38-50. F2M_R2 function properties

Property	Notes
Mode value	0010_0000_000*x*_0000_1100
Input	N = modulus, an irreducible polynomial
Output	B (or A, if selected) = R2 mod N, where R = 2^{SD} where S is size of a digit in bits and D is the number of digits of an irreducible polynomial, in other words D = ceiling [sizeof(N) in bits / S]
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes
Side effects	None
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • Modulus Even Error is set if N is even.
Flags set	None

38.6.9.26 Binary Polynomial (F_{2m}) Greatest Common Divisor (F2M_GCD) Function

Table 38-51. F2M_GCD function properties

Property	Notes
Mode value	0010_0000_000*x*_0000_1110
Input	• N = any polynomial. The most-significant digit of N must be non-zero. • A = any polynomial with degree less than or equal to N
Output	B (or A, if selected) = BINARY_GCD(A,N), a polynomial with degree less than or equal to polynomial A that divides both A and N
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 256 bytes • A and N may not both be zero
Side effects	A is modified
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 256. • A Size Error is set if the size of A is greater than size of N. • Divide-By-Zero Error is set if N or A = 0, or if the most significant digit of N = 0.
Flags set	PIO is set if the result is 1.

38.6.9.27 ECC F_p Point Add - Affine Coordinates (ECC_MOD_ADD) Function

Table 38-52. ECC_MOD_ADD function properties

Property	Notes
Mode value	0000_0000_000*x*_0000_1001
Input	• N = modulus, a prime number. The most significant digit of N must be non-zero The most significant digit of N must be non-zero • [A0, A1] = first addend in affine coordinates • A2 = ignored • A3 = elliptic curve "a" parameter • B0 = elliptic curve "b" parameter • [B1, B2] = second addend in affine coordinates • B3 = ignored
Output	[B1, B2] (or [A0, A1], if A output selected) = [A0, A1] + [B1, B2], where "+" represents an elliptic curve point addition. Output is in affine coordinates.
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 64 bytes • Point coordinates A0, A1, B1 and B2, and elliptic curve parameters A3 and B0 are elements of the prime field and therefore are less than the modulus N.
Side effects	A0, A1, A2, A3 and B3 are modified.
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 64. • Modulus Even Error is set if N is even. • A Size Error is set if the size of A is greater than size of N. • B Size Error will be set if size of B is greater than size of N. • Divide-By-Zero Error is set if the most-significant digit of N = 0.
Flags set	None

38.6.9.28 ECC F_p Point Add - Affine Coordinates, R2modN Input, (ECC_MOD_ADD_R2) Function

This function is more efficient than (ECC F_p Point Add - Affine Coordinates (ECC_MOD_ADD) Function), which first must compute R^2 mod N

Table 38-53. ECC_MOD_ADD_R2 function properties

Property	Notes
Mode value	0001_0000_000*x*_0000_1001
Input	• N = modulus, a prime number. The most significant digit of N must be non-zero The most significant digit of N must be non-zero • [A0, A1] = first addend point in affine coordinates (x,y) • A2 = ignored • A3 = elliptic curve "a" parameter • B0 = elliptic curve "b" parameter • [B1, B2] = second addend point in affine coordinates (x,y) • B3 = R2 (R^2 mod N) input

Table continues on the next page...

Table 38-53. ECC_MOD_ADD_R2 function properties (continued)

Property	Notes
Output	[B1, B2] (or [A0, A1], if A output selected) = [A0, A1] + [B1, B2], where "+" represents an elliptic curve point addition. Output is in affine coordinates.
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 64 bytes • Point coordinates A0, A1, B1 and B2, and elliptic curve parameters A3 and B0 are elements of the prime field formed by N.
Side effects	A0, A1, A2, A3 and B3 are modified.
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 64. • Modulus Even Error is set if N is even. • A Size Error is set if the size of A is greater than size of N. • B Size Error will be set if size of B is greater than size of N. • Divide-By-Zero Error is set if the most-significant digit of N = 0.
Flags set	None

38.6.9.29 ECC F_p Point Double - Affine Coordinates (ECC_MOD_DBL) Function

Table 38-54. ECC_MOD_DBL function properties

Property	Notes
Mode value	0000_0000_000*x*_0000_1010
Input	• N = modulus, a prime number. The most significant digit of N must be non-zero The most significant digit of N must be non-zero • [A0, A1, A2] = ignored • A3 = elliptic curve "a" parameter • B0 = elliptic curve "b" parameter • [B1, B2] = input point in affine coordinates • B3 = ignored
Output	[B1, B2] (or [A0, A1], if A output selected) = [B1, B2] + [B1, B2], where "+" represents an elliptic-curve point addition. Output is in affine coordinates (x, y).
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 64 bytes • Point coordinates B1 and B2, and elliptic curve parameters A3 and B0 are elements of the prime field formed by N.
Side effects	A0, A2, A3 and B3 are modified.
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 64. • Modulus Even Error is set if N is even. • A Size Error is set if size of A is greater than size of N. • B Size Error is set if size of B is greater than size of N. • Divide-by-Zero Error is set if the most significant digit of N = 0.
Flags set	PIZ is set if the result is the point at infinity.

38.6.9.30 ECC F_p Point Multiply - Affine Coordinates (ECC_MOD_MUL) Function

Table 38-55. ECC_MOD_MUL function properties

Property	Notes
Mode value	0000_0000_000*x*_0000_1011
Input	• N = modulus, a prime number. The most significant digit of N must be non-zero The most significant digit of N must be non-zero • E = scalar multiplier (k), any integer • [A0, A1] = multiplicand, an input point in affine coordinates (x,y) • A2 = ignored • A3 = elliptic curve "a" parameter • B0 = elliptic curve "b" parameter • B1 = ignored • B2 = ignored • B3 = ignored • A0-A3 and B0-B3 are four, equally size segments of A and B memory locations.
Output	• P[B1, B2] (or P[A0, A1], if A output selected) = E x P[A0, A1], where "x" denotes elliptic curve scalar point multiplication. Output is in affine coordinates (x,y). • B0 = undefined • B3 = undefined
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 64 bytes • The point (A0, A1) must be on the elliptic curve formed by (N, A3, B0).
Side effects	A0, A1, A2, A3 and B3 are modified.
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 64. • Key Size Error is set if size of E = 0 or size of E > 256. • Modulus Even Error is set if N is even. • A Size Error is set if size of A is greater than size of N. • B Size Error is set if size of B is greater than size of N. • Divide-by-Zero Error is set if the most significant digit of N = 0.
Flags set	PIZ is set if the result is the point at infinity.

The following special cases should be noted:

- For k = 0, this function returns a point at infinity; that is (0,0) if curve parameter "b" is nonzero and (0,1) otherwise.
- For k < 0, (that is, a negative scalar multiplication is required), its absolute value should be provided to the PKHA; that is, k = abs(-k). After the computation is complete, the formula -P= (x,-y) can be used to compute the "y" coordinate of the effective final result, and other coordinates are the same.

38.6.9.31 ECC F_p Point Multiply - Affine Coordinates, Timing Equalized (ECC_MOD_MUL_TEQ) Function

ECC_MOD_MUL_TEQ performs the same operation as ECC F_p Point Multiply - Affine Coordinates (ECC_MOD_MUL) Function but with an added timing equalization security feature. Its computation run-time is, for a given curve (N, A3, B0), constant for a given size of E.

38.6.9.32 ECC F_p Point Multiply with R2modN Input - Affine Coordinates (ECC_MOD_MUL_R2) Function

This function is more efficient than (ECC F_p Point Multiply - Affine Coordinates (ECC_MOD_MUL) Function), which first must compute R^2 mod N.

Table 38-56. ECC_MOD_MUL_R2 function properties

Property	Notes
Mode value	0001_0000_000*x*_0000_1011
Input	• N = modulus, a prime number. The most significant digit of N must be non-zero The most significant digit of N must be non-zero • E = key, scalar multiplier (k), any integer • [A0, A1] = multiplicand, an input point in affine coordinates (x,y) • A2 = ignored • A3 = elliptic curve "a" parameter • B0 = elliptic curve "b" parameter • B1 = R2 mod N, pre-computed as described in MOD_R2 • B2 = ignored • B3 = ignored • A0-A3 and B0-B3 are four, equally size segments of A and B memory locations.
Output	• P[B1, B2] (or P[A0, A1], if A output selected) = E x P[A0, A1], where "x" denotes elliptic curve scalar point multiplication. Output is in affine coordinates (x,y). • B0 = undefined • B3 = undefined
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 64 bytes • Point coordinates A0 and A1 and elliptic curve parameters A3 and B0 are elements of the prime field formed by the modulus N.
Side effects	A0, A1, A2, A3 and B3 are modified.
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 64. • Key Size Error is set if size of E = 0 or size of E > 256. • Modulus Even Error is set if N is even. • A Size Error is set if size of A is greater than size of N. • B Size Error is set if size of B is greater than size of N. • Divide-by-Zero Error is set if the most significant digit of N = 0.
Flags set	PIZ is set if the result is the point at infinity.

The following special cases should be noted:

- For k = 0, this function returns a point at infinity; that is, (0,0) if curve parameter "b" is nonzero and (0,1) otherwise.
- For k < 0, (that is, a negative scalar multiplication is required), its absolute value should be provided to the PKHA; that is, k = abs(- k). After the computation is complete, the formula -P = (x,-y) can be used to compute the "y" coordinate of the effective final result, and other coordinate is the same.

38.6.9.33 ECC F_p Point Multiply with R2modN Input - Affine Coordinates, Timing Equalized (ECC_MOD_MUL_R2_TEQ) Function

ECC_MOD_MUL_R2_TEQ performs the same operation as ECC F_p Point Multiply with R2modN Input - Affine Coordinates (ECC_MOD_MUL_R2) Function but with an added timing equalization security feature. Its computation run-time is, for a given curve (N, A3, B0), constant for a given size of E.

NOTE

The properties and special cases of ECC_MOD_MUL_R2_TEQ are identical to that of ECC F_p Point Multiply with R2modN Input - Affine Coordinates (ECC_MOD_MUL_R2) Function.

Table 38-57. ECC_MOD_MUL_R2_TEQ function

Property	Notes
Mode value	0001_0000_010*x*_0000_1011

38.6.9.34 ECC F_{2m} Point Add - Affine Coordinates (ECC_F2M_ADD) Function

Table 38-58. ECC_F2M_ADD function properties

Property	Notes
Mode value	0010_0000_000*x*_0000_1001
Input	• N = modulus, an irreducible polynomial. The most significant digit of N must be non-zero. • [A0, A1] = first addend in affine coordinates • A2 = ignored • A3 = elliptic curve "a" parameter • B0 = elliptic curve "c" parameter, where c = $b^{2^{m-2}}$ mod *n* • B1, B2] = second addend in affine coordinates • B3 = ignored
Output	P[B1, B2] (or P[A0, A1], if A output selected) = P[A0, A1] + P[B1, B2], where "+" represents an elliptic curve point addition. Output is in affine coordinates.

Table continues on the next page...

Table 38-58. ECC_F2M_ADD function properties (continued)

Property	Notes
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 64 bytes • Point coordinates A0, A1, B1, and B2 and elliptic curve parameters A3 and B0 are elements of the binary polynomial field N.
Side effects	A0, A1, A2, A3 and B3 are modified.
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 64. • Modulus Even Error is set if N is even. • A Size Error is set if the size of A is greater than size of N. • B Size Error will be set if size of B is greater than size of N. • Divide By Zero Error is set if the most significant digit of N = 0. • C is Zero Error if B3 is zero.
Flags set	None

38.6.9.35 ECC F_{2^m} Point Add - Affine Coordinates, R2modN Input (ECC_F2M_ADD_R2) Function

This function is more efficient than (ECC F_{2^m} Point Add - Affine Coordinates (ECC_F2M_ADD) Function), which first must compute R^2 mod N

Table 38-59. ECC_F2M_ADD_R2 function properties

Property	Notes
Mode value	0011_0000_000*x*_0000_1001
Input	• N = modulus, an irreducible polynomial. The most significant digit of N must be non-zero. • [A0, A1] = first addend in affine coordinates • A2 = ignored • A3 = elliptic curve "a" parameter • B0 = elliptic curve "c" parameter, where c = $b^{2^{m-2}}$ mod *n*. Must not be zero. • [B1, B2] = second addend in affine coordinates • B3 = R2 input
Output	P[B1, B2] (or P[A0, A1], if A output selected) = P[A0, A1] + P[B1, B2], where "+" represents an elliptic curve point addition. Output is in affine coordinates.
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 64 bytes • Point coordinates A0, A1, B1 and B2, and elliptic curve parameters A3 and B0 are elements of the binary polynomial field.
Side effects	A0, A1, A2, A3 and B3 are modified.
Errors reported	• Data Size Error is set if size of N > 64. • Modulus Even Error is set if N is even. • A Size Error is set if the size of A is greater than size of N. • B Size Error will be set if size of B is greater than size of N. • Divide By Zero Error is set if the most significant digit of N = 0. • C is Zero Error if B3 is zero.
Flags set	None

38.6.9.36 ECC F_{2m} Point Double - Affine Coordinates (ECC_F2M_DBL) Function

Table 38-60. ECC_F2M_DBL function properties

Property	Notes
Mode value	0010_0000_000*x*_0000_1010
Input	• N = modulus, an irreducible polynomial. The most significant digit of N must be non-zero. • A0, A1, A2 = ignored • A3 = elliptic curve "a" parameter • B0 = elliptic curve "c" parameter where c = $b^{2^{m-2}}$ mod *n* • [B1, B2] = input point in affine coordinates • B3 = ignored
Output	P[B1, B2] (or P[A0, A1], if A output selected) = P[B1, B2] + P[B1, B2], where "+" represents an elliptic-curve point addition. Output is in affine coordinates.
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 64 bytes • Point coordinates B1 and B2, and elliptic curve parameters A3 and B0 are elements of the binary polynomial field formed by N.
Side effects	A0, A2, A3 and B3 are modified.
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 64. • Modulus Even Error is set if N is even. • A Size Error is set if size of A is greater than size of N. • B Size Error is set if size of B is greater than size of N. • Divide-by-Zero Error is set if the most significant digit of N = 0. • C is Zero Error if B3 is zero.
Flags set	PIZ is set if the result is the point at infinity.

38.6.9.37 ECC F_{2m} Point Multiply - Affine Coordinates (ECC_F2M_MUL) Function

Table 38-61. ECC_F2M_MUL function properties

Property	Notes
Mode value	0010_0000_000*x*_0000_1011
Input	• N = modulus, an irreducible polynomial. The most significant digit of N must be non-zero. • E = key, scalar multiplier (k), any integer • [A0, A1] = multiplicand, an input point in affine coordinates (x,y) • A2 = ignored • A3 = elliptic curve "a" parameter • B0 = elliptic curve "c" parameter where c = $b^{2^{m-2}}$ mod *n* • B1 = ignored • B2 = ignored • B3 = ignored
Output	• P[B1, B2] (or P[A0, A1], if A output selected) = E x P[A0, A1], where "x" denotes elliptic curve scalar point multiplication. Output is in affine coordinates (x,y).

Table continues on the next page...

Table 38-61. ECC_F2M_MUL function properties (continued)

Property	Notes
	• B0 = undefined • B3 = undefined
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 64 bytes (irreducible polynomial of maximum degree 511) • Point coordinates A0 and A1 and elliptic curve parameters A3 and B0 must be elements of the binary polynomial field.
Side effects	A0, A1 and B3 are modified.
Errors reported	• Data Size Error is set if size of N = 0or size of N > 64. • Key Size Error is set if size of E = 0 or size of E > 256. • Modulus Even Error is set if N is even. • A Size Error is set if size of A is greater than size of N. • B Size Error is set if size of B is greater than size of N. • Divide-by-Zero Error is set if the most significant digit of N = 0. • C is Zero Error if B3 is zero.
Flags set	PIZ is set if the result is the point at infinity.

The following special cases should be noted:

- For E = 0, this function returns a point at infinity (0,0).
- For E < 0, (that is, a negative scalar multiplication is required), its absolute value should be provided to the PKHA; that is, `k = -E`). After the multiplication is complete, the formula `-P= (x, x+y)` can be used to compute the y coordinate of the effective final result; the x coordinate stays the same.

38.6.9.38 ECC F_{2^m} Point Multiply - Affine Coordinates, Timing Equalized (ECC_F2M_MUL_TEQ) Function

ECC_F2M_MUL_TEQ performs the same operation as ECC F_{2^m} Point Multiply - Affine Coordinates (ECC_F2M_MUL) Function but with an added timing equalization security feature. Its computation run-time is, for a given curve (N, A3, B0), constant for a given size of E.

NOTE

The properties of ECC_F2M_MUL_TEQ and its special cases are identical to that of ECC F_{2^m} Point Multiply - Affine Coordinates (ECC_F2M_MUL) Function.

38.6.9.39 ECC F_{2^m} Point Multiply with R2modN Input- Affine Coordinates (ECC_F2M_MUL_R2) Function

This function is more efficient than (ECC F_{2^m} Point Multiply - Affine Coordinates (ECC_F2M_MUL) Function), which first must compute R^2 mod N.

Table 38-62. ECC_F2M_MUL_R2 function properties

Property	Notes
Mode value	0011_0000_000*x*_0000_1011
Input	• N = modulus, an irreducible polynomial. The most significant digit of N must be non-zero. • E = key, scalar multiplier (k), any integer • [A0, A1] = multiplicand, an input point in affine coordinates (x,y) • A2 = ignored • A3 = elliptic curve "a" parameter • B0 = elliptic curve "c" parameter where c = $b^{2^{m-2}}$ mod *n* and m = degree of polynomial M • B1 = (f2m) R2 mod N, pre-computed as described in F2M_R2MODN (0Eh) • B2 = ignored • B3 = ignored • A0-A3 and B0-B3 are four, equally size segments of A and B memory locations.
Output	• P[B1, B2] (or P[A0, A1], if A output selected) = E x P[A0, A1], where "x" denotes elliptic curve scalar point multiplication. Output is in affine coordinates (x,y). • B0 = undefined • B3 = undefined
Requirements	• Minimum modulus size = 1 byte • Maximum modulus size = 64 bytes (irreducible polynomial of maximum degree 511) • Point coordinates A0 and A1 and elliptic curve parameters A3 and B0 must be elements of the binary polynomial field.
Side effects	A0, A1, A2, and B3 are modified.
Errors reported	• Data Size Error is set if size of N = 0 or size of N > 64. • Key Size Error is set if size of E = 0 or size of E > 256. • Modulus Even Error is set if N is even. • A Size Error is set if size of A is greater than size of N. • B Size Error is set if size of B is greater than size of N. • Divide-by-Zero Error is set if the most significant digit of N = 0. • C is Zero Error if B3 is zero.
Flags set	PIZ is set if the result is the point at infinity.

The following special cases should be noted:

- For k = 0, this function returns a point at infinity (0,0).
- For k < 0, (that is, a negative scalar multiplication is required), its absolute value should be provided to the PKHA; that is, k = abs(- k). After the computation is complete, the formula -P = (x, x+y) can be used to compute the "y" coordinate of the effective final result, and other coordinate is the same.

38.6.9.40 ECC F_{2^m} Point Multiply with R2modN Input - Affine Coordinates, timing equalized (ECC_F2M_MUL_R2_TEQ) Function

ECC_F2M_MUL_R2_TEQ performs the same operation as ECC F_{2^m} Point Multiply with R2modN Input- Affine Coordinates (ECC_F2M_MUL_R2) Function but with an added timing equalization security feature. Its computation run-time is, for a given curve (N, A3, B0), constant for a given size of E.

Table 38-63. ECC_F2M_MUL_R2_TEQ function

Property	Notes
Mode value	0011_0000_010*x*_0000_1011

38.6.9.41 Copy memory, N-Size (COPY_NSZ)

This function copies the amount of data specified by the N Size register from the specified source register or register quadrant to the specified destination register or register quadrant. The source and destination are specified in the Mode Value. The source can be A, B or N. The destination can be A, B, E or N, but not the same as the source.

In a quadrant copy, when NSZ exceeds the length of a quadrant, the copy will carry on into the next (higher-numbered) quadrant(s).

When the copy operation has completed, the size register for the destination register will be updated to contain the size of the modulus.

Table 38-64. COPY_NSZ function properties

Property	Notes
Mode value	*xxxx*_0000_*xxxx*_*xx*01_0000
Input	None
Output	None
Requirements	N-size Register must contain a valid value
Side effects	Destination register size register is updated to contain N-size.
Errors reported	None
Flags set	None

38.6.9.42 Copy memory, source-size (COPY_SSZ)

This function copies the amount of data specified in the size register for the source register from the specified source register or register quadrant to the specified destination register or register quadrant. The source and destination are specified in the Mode Value. The source can be A, B or N. The destination can be A, B, E or N, but not the same as the source.

In a quadrant copy, when SSZ exceeds the length of a quadrant, the copy will carry on into the next (higher-numbered) quadrant(s).

When the copy operation has completed, the size for the destination register will be updated to contain the source register size.

Table 38-65. COPY_SSZ function properties

Property	Notes
Mode value	*xxxx*_0000_*xxxx*_*xx*01_0001
Input	None
Output	None
Requirements	The size register of the source register must contain a valid value.
Side effects	Destination register size register is updated to contain the source register size.
Errors reported	None
Flags set	None

38.6.10 Special values for common ECC domains

Software can sometimes use the PKHA more effectively if the Montgomery Conversion Factor (R^2 mod N) is either provided or previously used to convert other inputs into Montgomery form. For convenience, the conversion factors for common ECC domains have been computed and published here. Some of the other domain values are provided to aid in definite identification of the domain, in the case that the name is not found or is not an exact match.

The following tables give these values for the *q* and *r* modulus values found in ECC domains. These associated Montgomery values are dependent upon the PKHA digit size (16, 32, 64, 128). These tables are for a PKHA with a 16-bit digit.

ECC F_{2^m} requires a *c* (also called *b′*) parameter for the elliptic curve in place of the *b* value. Table 38-67 provides these values in addition to the Montgomery values. The *b′* values are universal and do not change with PKHA digit size.

The following variable definitions apply to both tables. Variable names (*q, r, b, c*) follow the conventions of IEEE Std 1363.

Name

The names in these tables are associated with, or named in, various published standards. Neither the names nor the domains are guaranteed to be complete. Two values of the domain parameters are provided for purposes of identification.

- Those beginning with "P-", "K-", and "B-" are in FIPS 186 from NIST, found at www.csrc.nist.gov
- Those beginning with "ansix9" are names from ANS X9.62-2005; those beginning with "prime" or "c2pnb" are from an earlier ANSI document
- Those beginning with "sec" are from SEC 2 from the Standards for Efficient Cryptography group, found at www.secg.org
- Those beginning with "wtls" are taken from Wireless Transport Layer Security / Wireless Access Protocol, Version 06-Apr-2001, WAP-261-WTLS-20010406-a. Not all software libraries agree with the mapping of these names to values; care has been taken to identify the values based upon the source documentation.
- Those beginning with "ECDSA", "ECP", "EC2N", "ecp_group", and "Oakley" are from various RFCs found at www.ietf.org
- Those beginning with "GOST" are from the Russian standard GOST R 3410-2001
- Those beginning with "brainpool" are from ECC Brainpool, found at www.ecc-brainpool.org and republished in RFC 5639

R

R is the Montgomery factor. Its value is 2^{SD}, where *D* is the PKHA digit size in bits, and *S* is the minimum number of digits needed to hold the modulus. As an example, for a modulus of nine bytes (72 bits), *R* would be

- 2^{80} for a PKHA with digit size of 16 bits
- 2^{96} for a PKHA with digit size of 32 bits
- 2^{128} for a PKHA with digit size of 64 or 128 bits

q

This is the *field-defining* value for the elliptic curve. For F_p curves, it is the prime number used as the modulus for all point arithmetic; it is named *p* in some other publications. For F_{2^m} curves, it is the irreducible binary polynomial used as the modulus for all point arithmetic. It is not, as usually defined, $q = 2^m$, i.e. the size of the field.

L

This is the number of bytes needed to hold *q* and each of its associated values: *R2modq*, *a*,*b*,*c*, the point coordinates *x* and *y*, the result of an ECDH key agreement, etc.

R2modq

This is $R^2 \bmod q$, the Montgomery Conversion Factor when *q* is the modulus.

r

This is the (usually prime) number which is the order of *G*, the generator point. It is also usually used as the modulus for the non-ECC-related arithmetic in an ECC primitive. This variable is named *n* in some other publications.

N

This is the number of bytes needed to hold *r* and each of its associated values: *R2modr*, private keys, each of the two components of an ECDSA signature, etc.

R2modr

This is $R^2 \ mod \ r$, the Montgomery Conversion Factor when *r* is the modulus.

b / c (b')

b is the coefficient for the x^0 (ones) term in an F_{2^m} elliptic curve equation. Its relationship with *c* is $b = c^4$. *c* is sometimes referred to as *b'* in Freescale documentation.

The domains in the table are ordered by size.

Table 38-66. Special Values for common ECC F_p domains when PKHA digit size is 16 bits

Name	L	N	
	var		Value (hex, decimal, sums of powers)
secp112r1	14	14	
wtls6	q		0xDB7C2ABF62E35E668076BEAD208B 4451685225093714772084598273548427
	R2modq		0x00030000000000000000000000
	r		0xDB7C2ABF62E35E7628DFAC6561C5 44516852250937147764918915425489 33
	R2modr		0x6A1DD1858198B5BADC6CFD628D0A
secp112r2	14	14	
	q		0xDB7C2ABF62E35E668076BEAD208B 4451685225093714772084598273548427
	R2modq		0x00030000000000000000000000
	r		0x36DF0AAFD8B8D7597CA10520D04B 1112921306273428674967732714786891
	R2modr		0x2C19866CFA502191682A07EBB1DF
wtls8	14	15	
	q		0xFFFFFFFFFFFFFFFFFFFFFFFFDE7 519229685853482762853049632921955 9 $2^{112} - 2^9 - 2^4 - 2^3 - 1$
	R2modq		0x00000000000000000000046671
	r		0x0100000000000001ECEA551AD837E9 519229685853482776727383611436029 7
	R2modr		0x00E074FD104C86569DB6C204A52932
secp128r1	16	16	
	q		0xFFFFFFFDFFFFFFFFFFFFFFFFFFFFFFFF

Table continues on the next page...

Table 38-66. Special Values for common ECC F_p domains when PKHA digit size is 16 bits (continued)

Name	L	N	
	var		Value (hex, decimal, sums of powers)
			340282366762482138434845932244680310783
			$2^{128} - 2^{97} - 1$
	R2modq		0x00000024000000040000000800000011
	r		0xFFFFFFFE0000000075A30D1B9038A115
			340282366762482138443322565580356624661
	R2modr		0x71875047CDD8151626BC6448FADE9BED
secp128r2	16	16	
	q		0xFFFFFFFDFFFFFFFFFFFFFFFFFFFFFFFF
			340282366762482138434845932244680310783
			$2^{128} - 2^{97} - 1$
	R2modq		0x00000024000000040000000800000011
	r		0x3FFFFFFF7FFFFFFFBE0024720613B5A3
			85070591690620534603955721926813660579
	R2modr		0x0EFCA409C09D126A99CD2E9404A3B434
secp160k1	20	21	
ansix9p160k1	q		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFAC73
			1461501637330902918203684832716283019651637554291
			$2^{160} - 2^{32} - 2^{14} - 2^{12} - 2^9 - 2^8 - 2^7 - 2^3 - 2^2 - 1$
	R2modq		0x0000000000000000000000010000A71A1B44BBA9
	r		0x0100000000000000000001B8FA16DFAB9ACA16B6B3
			1461501637330902918203686915170869725397159163571
	R2modr		0x008CCA75EBCDCF2BABDFE44FB4D282030A56ECBB7D
secp160r1	20	21	
ansix9p160r1	q		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7FFFFFFF
wtls7			1461501637330902918203684832716283019653785059327
			$2^{160} - 2^{31} - 1$
	R2modq		0x0000000000000000000000004000000100000001
	r		0x0100000000000000000001F4C8F927AED3CA752257
			1461501637330902918203687197606826779884643492439
	R2modr		0x00436AB204A0E626837A98A22C8D8455E5563AADF1
secp160r2	20	21	
ansix9p160r2	q		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFAC73
			1461501637330902918203684832716283019651637554291
			$2^{160} - 2^{32} - 2^{14} - 2^{12} - 2^9 - 2^8 - 2^7 - 2^3 - 2^2 - 1$
	R2modq		0x0000000000000000000000010000A71A1B44BBA9
	r		0x0100000000000000000000351EE786A818F3A1A16B
			1461501637330902918203685083571792140653176136043

Table continues on the next page...

Table 38-66. Special Values for common ECC F_p domains when PKHA digit size is 16 bits (continued)

Name	L	N	
	var		Value (hex, decimal, sums of powers)
	R2modr		0x00D68E04EB76E5A181476A1C23DB8912D0E66BA378
wtls9	20	21	
	q		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC808F 1461501637330902918203684832716283019655932313743 $2^{160} - 2^{17} - 2^{16} - 2^{14} - 2^{13} - 2^{12} - 2^{11} - 2^{10} - 2^{9} - 2^{8} - 2^{6} - 2^{5} - 2^{4} - 1$
	R2modq		0x0000000000000000000000000000C3C174FE1
	r		0x0100000000000000000001CDC98AE0E2DE574ABF33 1461501637330902918203687013445034429194588307251
	R2modr		0x00519041ADCC3AB9A731EC45CB8499428E993A39C5
brainpoolP160r1	20	20	
	q		0xE95E4A5F737059DC60DFC7AD95B3D8139515620F 1332297598440044874827085558802491743757193798159
	R2modq		0x6CF12F81C0CA7EF8FED717E0B333F8D625BC14FF
	r		0xE95E4A5F737059DC60DF5991D45029409E60FC09 1332297598440044874827085038830181364212942568457
	R2modr		0x2BC73851FC9BE6F69E31FE16FC61D4351FDF90EA
brainpoolP160t1	20	20	
	q		0xE95E4A5F737059DC60DFC7AD95B3D8139515620F 1332297598440044874827085558802491743757193798159
	R2modq		0x6CF12F81C0CA7EF8FED717E0B333F8D625BC14FF
	r		0xE95E4A5F737059DC60DF5991D45029409E60FC09 1332297598440044874827085038830181364212942568457
	R2modr		0x2BC73851FC9BE6F69E31FE16FC61D4351FDF90EA
P-192 secp192r1 ansix9p192r1 prime192v1 ECPRGF192Random	24	24	
	q		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFF 6277101735386680763835789423207666416083908700390324961279 $2^{192} - 2^{64} - 1$
	R2modq		0x0000000000000001000000000000000020000000000000001
	r		0xFFFFFFFFFFFFFFFFFFFFFFFF99DEF836146BC9B1B4D22831 6277101735386680763835789423176059013767194773182842284081
	R2modr		0x28BE5677EA0581A24696EA5BBB3A6BEECE66BACCDEB35961
secp192k1 ansix9p192k1	24	24	
	q		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFEE37 6277101735386680763835789423207666416102355444459739541047 $2^{192} - 2^{32} - 2^{12} - 2^{8} - 2^{7} - 2^{6} - 2^{3} - 1$
	R2modq		0x00000000000000000000000000000100002392013C4FD1
	r		0xFFFFFFFFFFFFFFFFFFFFFFFE26F2FC170F69466A74DEFD8D

Table continues on the next page...

Table 38-66. Special Values for common ECC F_p domains when PKHA digit size is 16 bits (continued)

Name	L	N	
	var		Value (hex, decimal, sums of powers)
			6277101735386680763835789423061264271957123915200845512077
	R2modr		0x6A21191C2EC4B2B1F0F4F172195E97E2461C1989250F0702
prime192v2	24	24	
	q		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFF 6277101735386680763835789423207666416083908700390324961279 $2^{192} - 2^{64} - 1$
	R2modq		0x0000000000000001000000000000000020000000000000001
	r		0xFFFFFFFFFFFFFFFFFFFFFFFE5FB1A724DC80418648D8DD31 62771017353866807638357894230788259361921005375843850560 49
	R2modr		0xA4FEB8C277C030E139DA8CFB4E35E1F62814A261001BE8FF
prime192v3	24	24	
	q		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFF 6277101735386680763835789423207666416083908700390324961279 $2^{192} - 2^{64} - 1$
	R2modq		0x0000000000000001000000000000000020000000000000001
	r		0xFFFFFFFFFFFFFFFFFFFFFFFF7A62D031C83F4294F640EC13 6277101735386680763835789423166314882687165660350679936019
	R2modr		0x45BCB42FF1CC05D0194D076B366D09BF0305982367330969
brainpoolP192r1	24	24	
	q		0xC302F41D932A36CDA7A3463093D18DB78FCE476DE1A86297 4781668983906166242955001894344923773259119655253013193367
	R2modq		0xB6225126EED34F1033BF484602C3FE69E2474C6972C7B21A
	r		0xC302F41D932A36CDA7A3462F9E9E916B5BE8F1029AC4ACC1 4781668983906166242955001894269038308119863659119834868929
	R2modr		0x98769B9CE772102BBF4AFD5DBF53AFF0B4727C80E407E8F8
brainpoolP192t1	24	24	
	q		0xC302F41D932A36CDA7A3463093D18DB78FCE476DE1A86297 4781668983906166242955001894344923773259119655253013193367
	R2modq		0xB6225126EED34F1033BF484602C3FE69E2474C6972C7B21A
	r		0xC302F41D932A36CDA7A3462F9E9E916B5BE8F1029AC4ACC1 4781668983906166242955001894269038308119863659119834868929
	R2modr		0x98769B9CE772102BBF4AFD5DBF53AFF0B4727C80E407E8F8
P-224	28	28	
secp224r1	q		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000000000000000000001
ansix9p224r1			26959946667150639794667015087019630673557916260026308143510066298881
wtls12	R2modq		0x00000000FFFFFFFFFFFFFFFFFFFFFFFE000000000000000000000001
ECPRGF224Random	r		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFF16A2E0B8F03E13DD29455C5C2A3D

Table continues on the next page...

Table 38-66. Special Values for common ECC F_p domains when PKHA digit size is 16 bits (continued)

Name	L	N	
	var		Value (hex, decimal, sums of powers)
			26959946667150639794667015087019625940457807714424391721682722368061
	R2modr		0xD4BAA4CF1822BC47B1E979616AD09D9197A545526BDAAE6C3AD01289
secp224k1	28	29	
ansix9p224k1	q		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFE56D 26959946667150639794667015087019630673637144422540572481099315275117 $2^{224} - 2^{32} - 2^{12} - 2^{11} - 2^{9} - 2^{7} - 2^{4} - 2^{1} - 1$
	R2modq		0x0000000000000000000000000000000000000010000352602C23069
	r		0x0100000000000000000000000001DCE8D2EC6184CAF0A971769FB1F7 26959946667150639794667015087019640346510327083120074548994958668279
	R2modr		0x0004DF3DA0993FF72BB882BD88BBFF3BF810C92B3D43C82C98C348E90E
brainpoolP224r1	28	28	
	q		0xD7C134AA264366862A18302575D1D787B09F075797DA89F57EC8C0FF 22721622932454352787552537995910928073340732145944992304435472941311
	R2modq		0x0578FD592E6A6CE43FE8A2AA96AF774C43C20E727867CA8064DCD04F
	r		0xD7C134AA264366862A18302575D0FB98D116BC4B6DDEBCA3A5A7939F 22721622932454352787552537995910923612567546342330757191396560966559
	R2modr		0x4A73A6563211A5611E9CAE249F24919B9399652CADDAF8AA486CA401
brainpoolP224t1	28	28	
	q		0xD7C134AA264366862A18302575D1D787B09F075797DA89F57EC8C0FF 22721622932454352787552537995910928073340732145944992304435472941311
	R2modq		0x0578FD592E6A6CE43FE8A2AA96AF774C43C20E727867CA8064DCD04F
	r		0xD7C134AA264366862A18302575D0FB98D116BC4B6DDEBCA3A5A7939F 22721622932454352787552537995910923612567546342330757191396560966559
	R2modr		0x4A73A6563211A5611E9CAE249F24919B9399652CADDAF8AA486CA401
prime239v1	30	30	
	q		0x7FFFFFFFFFFFFFFFFFFFFFF7FFFFFFFFFFF8000000000007FFFFFFFFFF 883423532389192164791648750360308885314476597252960362792450860609699839
	R2modq		0x00000000000000000000005000000000005FFFFFFFFFFC000000000008
	r		0x7FFFFFFFFFFFFFFFFFFFFFF7FFFFF9E5E9A9F5D9071FBD1522688909D0B 883423532389192164791648750360308884807550341691627752275345424702807307
	R2modr		0x17789087ABE4B1BE15BDEB5DDB90D9E27D9E8C4D1C2BB940279628B08300
prime239v2	30	30	
	q		0x7FFFFFFFFFFFFFFFFFFFFFF7FFFFFFFFFFF8000000000007FFFFFFFFFF 883423532389192164791648750360308885314476597252960362792450860609699839
	R2modq		0x00000000000000000000005000000000005FFFFFFFFFFC000000000008

Table continues on the next page...

Table 38-66. Special Values for common ECC F_p domains when PKHA digit size is 16 bits (continued)

Name	L	N	
	var		Value (hex, decimal, sums of powers)
	r		0x7FFFFFFFFFFFFFFFFFFFFFFF800000CFA7E8594377D414C03821BC582063 88342353238919216479164875036030888639268765754699385514776573245129 5331
	R2modr		0x3CB60034F6CF025EBF73DDE8691B5F3A7A21FC865937164FCB06304ACE1E
prime239v3	30	30	
	q		0x7FFFFFFFFFFFFFFFFFFFFFFF7FFFFFFFFFFF8000000000007FFFFFFFFFF 88342353238919216479164875036030888531447659725296036279245086060969 9839
	R2modq		0x000000000000000000000005000000000005FFFFFFFFFFC000000000008
	r		0x7FFFFFFFFFFFFFFFFFFFFFFF7FFFF975DEB41B3A6057C3C432146526551 88342353238919216479164875036030888477119036976592255051796717105803 4001
	R2modr		0x2441C5CD91500EB94E46F167137CF1B742FF58158145184FF0BE2809E6AD
P-256 secp256r1 ansix9p256r1 prime256v1 ECDSA-256 ecp_group_19 ECPRGF256Random	32	32	
	q		0xFFFFFFFF00000001000000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFF 11579208921035624876269744694940757353008614341529031419553363130886 7097853951
	R2modq		0x00000004FFFFFFFDFFFFFFFFFFFFFFFFEFFFFFFFBFFFFFFFF000000000000000 3
	r		0xFFFFFFFF00000000FFFFFFFFFFFFFFFFBCE6FAADA7179E84F3B9CAC2FC632 551 11579208921035624876269744694940757352999695522413576034242225906106 8512044369
	R2modr		0x66E12D94F3D956202845B2392B6BEC594699799C49BD6FA683244C95BE79EEA 2
secp256k1 ansix9p256k1	32	32	
	q		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFF C2F 11579208923731619542357098500868790785326998466564056403945758400790 8834671663 $2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$
	R2modq		0x000000000000000000000000000000000000000000000001000007A2000E90A1
	r		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD036 4141 11579208923731619542357098500868790785283756427907490438260516314151 8161494337
	R2modr		0x9D671CD581C69BC5E697F5E45BCD07C6741496C20E7CF878896CF21467D7D1 40
brainpoolP256r1	32	32	
	q		0xA9FB57DBA1EEA9BC3E660A909D838D726E3BF623D52620282013481D1F6E537 7

Table continues on the next page...

Table 38-66. Special Values for common ECC F_p domains when PKHA digit size is 16 bits (continued)

Name	L	N	
	var		Value (hex, decimal, sums of powers)
			76884956397045344220809746629001649093037950200943055203735601445031516197751
	R2modq		0x4717AA21E5957FA8A1ECDACD6B1AC8075CCE4C26614D4F4D8CFEDF7BA6465B6C
	r		0xA9FB57DBA1EEA9BC3E660A909D838D718C397AA3B561A6F7901E0E82974856A7 76884956397045344220809746629001649092737531784414529538755519063063536359079
	R2modr		0x0B25F1B9C32367629B7F25E76C815CB0F35D176A1134E4A0E1D8D8DE3312FCA6
brainpoolP256t1	32	32	
	q		0xA9FB57DBA1EEA9BC3E660A909D838D726E3BF623D52620282013481D1F6E5377 76884956397045344220809746629001649093037950200943055203735601445031516197751
	R2modq		0x4717AA21E5957FA8A1ECDACD6B1AC8075CCE4C26614D4F4D8CFEDF7BA6465B6C
	r		0xA9FB57DBA1EEA9BC3E660A909D838D718C397AA3B561A6F7901E0E82974856A7 76884956397045344220809746629001649092737531784414529538755519063063536359079
	R2modr		0x0B25F1B9C32367629B7F25E76C815CB0F35D176A1134E4A0E1D8D8DE3312FCA6
GOSTR3410-CryptoPro-A	32	32	
	q		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD97 115792089237316195423570985008687907853269984665640564039457584007913129639319 $2^{256} - 2^9 - 2^6 - 2^5 - 2^3 - 1$
	R2modq		0x000000000000000000000000000000000000000000000000000000005CF11
	r		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6C611070995AD10045841B09B761B893 115792089237316195423570985008687907853073762908499243225378155805079068850323
	R2modr		0x551FE9CB451179DBF74885D08A3714C6FB07F8222E76DD529AC2D7858E79A469
GOSTR3410-CryptoPro-B	32	32	
	q		0x8000000000000000000000000000000000000000000000000000000000000C99 57896044618658097711785492504343953926634992332820282019728792003956564823193
	R2modq		0x000000000000000000000000000000000000000000000000000000027ACDC4
	r		0x8000000000000000000000000000015F700CFFF1A624E5E497161BCC8A198F

Table continues on the next page...

Table 38-66. Special Values for common ECC F_p domains when PKHA digit size is 16 bits (continued)

Name	L	N	
	var		Value (hex, decimal, sums of powers)
			57896044618658097711785492504343953927102133160255826820068844496087732066703
	R2modr		0x09D1D2C4E50824664A2E7E2F6882CF102A3104A7EA43E85529B721F4E6CD7823
GOSTR3410-CryptoPro-C	32	32	
	q		0x9B9F605F5A858107AB1EC85E6B41C8AACF846E86789051D37998F7B9022D759B 70390085352083305199547718019018437841079516630045180471284346843705633502619
	R2modq		0x807A394EDE097652186304212849C07B1017BB39C2D346C5409973B4C427FCEA
	r		0x9B9F605F5A858107AB1EC85E6B41C8AA582CA3511EDDFB74F02F3A6598980BB9 70390085352083305199547718019018437840920882647164081035322601458352298396601
	R2modr		0x7AA61B49A49D4759C67E5D0EE96E8ED304FDA8694AFDA24BE94FAAB66ABA180E
brainpoolP320r1	40	40	
	q		0xD35E472036BC4FB7E13C785ED201E065F98FCFA6F6F40DEF4F92B9EC7893EC28FCD412B1F1B32E27 176359332223916635416190984244601952088951277271951519277296041528864086880214981809550149990352 7
	R2modq		0xA259BA4A6C2D92525455A964E614D6D21F4C881F30C5B676C2478A8D906978EF994EE88A743B52F9
	r		0xD35E472036BC4FB7E13C785ED201E065F98FCFA5B68F12A32D482EC7EE8658E98691555B44C59311 176359332223916635416190984244601952088951277271768606376068612401678478484584346835685258203921
	R2modr		0x31EC87C73200B14FE30D35244E6390FE86B330BCAF86C40991C3001BE0E16805679D29DF2513E4CD
brainpoolP320t1	40	40	
	q		0xD35E472036BC4FB7E13C785ED201E065F98FCFA6F6F40DEF4F92B9EC7893EC28FCD412B1F1B32E27 176359332223916635416190984244601952088951277271951519277296041528864086880214981809550149990352 7
	R2modq		0xA259BA4A6C2D92525455A964E614D6D21F4C881F30C5B676C2478A8D906978EF994EE88A743B52F9
	r		0xD35E472036BC4FB7E13C785ED201E065F98FCFA5B68F12A32D482EC7EE8658E98691555B44C59311 176359332223916635416190984244601952088951277271768606376068612401678478484584346835685258203921
	R2modr		0x31EC87C73200B14FE30D35244E6390FE86B330BCAF86C40991C3001BE0E16805679D29DF2513E4CD

Table continues on the next page...

Table 38-66. Special Values for common ECC F_p domains when PKHA digit size is 16 bits (continued)

Name	L	N	
	var		Value (hex, decimal, sums of powers)
P-384 secp384r1 ansix9p384r1 ECDSA-384 ecp_group_20 ECPRGF384Random	48	48	
	q		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFFFF0000000000000000FFFFFFFF 39402006196394479212279040100143613805079739270465446667948293404245721771496870329047266088258938001861606973112319
	R2modq		0x0000000000000000000000000000000100000002000000000FFFFFFFE00000000 00000200000000FFFFFFFE00000001
	r		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC7634D81F4372 DDF581A0DB248B0A77AECEC196ACCC52973 39402006196394479212279040100143613805079739270465446667946905279627659399113263569398956308152294913554433653942643
	R2modr		0x0C84EE012B39BF213FB05B7A28266895D40D49174AAB1CC5BC3E483AFCB82947FF3D81E5DF1AA4192D319B2419B409A9
brainpoolP384r1	48	48	
	q		0x8CB91E82A3386D280F5D6F7E50E641DF152F7109ED5456B412B1DA197FB71123ACD3A729901D1A71874700133107EC53 21659270770119316173069236842332604979796116387017648600081618503821089934025961822236561982844534088440708417973331
	R2modq		0x36BF6883178DF842D5C6EF3BA57E052C621401919918D5AF8E28F99CC9940899535283343D7FD965087CEFFF40B64BDE
	r		0x8CB91E82A3386D280F5D6F7E50E641DF152F7109ED5456B31F166E6CAC0425A7CF3AB6AF6B7FC3103B883202E9046565 21659270770119316173069236842332604979796116387017648600075645274821611501358515537962695117368903252229601718723941
	R2modr		0x0CE8941A614E97C28F886DC965165FDB574A74CB52D748FF2A927E3B9802688A37264E202F2B6B6EAC4ED3A2DE771C8E
brainpoolP384t1	48	48	
	q		0x8CB91E82A3386D280F5D6F7E50E641DF152F7109ED5456B412B1DA197FB71123ACD3A729901D1A71874700133107EC53 21659270770119316173069236842332604979796116387017648600081618503821089934025961822236561982844534088440708417973331
	R2modq		0x36BF6883178DF842D5C6EF3BA57E052C621401919918D5AF8E28F99CC9940899535283343D7FD965087CEFFF40B64BDE
	r		0x8CB91E82A3386D280F5D6F7E50E641DF152F7109ED5456B31F166E6CAC0425A7CF3AB6AF6B7FC3103B883202E9046565 21659270770119316173069236842332604979796116387017648600075645274821611501358515537962695117368903252229601718723941
	R2modr		0x0CE8941A614E97C28F886DC965165FDB574A74CB52D748FF2A927E3B9802688A37264E202F2B6B6EAC4ED3A2DE771C8E
brainpoolP512r1	64	64	
	q		0xAADD9DB8DBE9C48B3FD4E6AE33C9FC07CB308DB3B3C9D20ED6639CCA703308717D4D9B009BC66842AECDA12AE6A380E62881FF2F2D82C68528AA6056583A48F3

Table continues on the next page...

Table 38-66. Special Values for common ECC F_p domains when PKHA digit size is 16 bits (continued)

Name	L	N	
	var		Value (hex, decimal, sums of powers)
			894896220765023255165660281515915342216260964409835451134459718720005 701041355243991793430419195694276544653038642734593796389430992392853 6070534607816947
	R2modq		0x3C4C9D05A9FF6450202E19402056EECCA16DAA5FD42BFF8319486FD8D58980 57E0C19A7783514A2553B7F9BC905AFFD3793FB1302715790549AD144A6158F205
	r		0xAADD9DB8DBE9C48B3FD4E6AE33C9FC07CB308DB3B3C9D20ED6639CCA7033 0870553E5C414CA92619418661197FAC10471DB1D381085DDADDB58796829CA90 069 894896220765023255165660281515915342216260964409835451134459718720005 701041341852837898173064352495985745139837002928058309421561388204397 3354392115544169
	R2modr		0xA794586A718407B095DF1B4C194B2E56723C37A22F16BBDFD7F9CC263B790D E3A6F230C72F0207E83EC64BD033B7627F0886B75895283DDDD2A3681ECDA816 71
brainpoolP512t1	64	64	
	q		0xAADD9DB8DBE9C48B3FD4E6AE33C9FC07CB308DB3B3C9D20ED6639CCA7033 08717D4D9B009BC66842AECDA12AE6A380E62881FF2F2D82C68528AA6056583A 48F3 894896220765023255165660281515915342216260964409835451134459718720005 701041355243991793430419195694276544653038642734593796389430992392853 6070534607816947
	R2modq		0x3C4C9D05A9FF6450202E19402056EECCA16DAA5FD42BFF8319486FD8D58980 57E0C19A7783514A2553B7F9BC905AFFD3793FB1302715790549AD144A6158F205
	r		0xAADD9DB8DBE9C48B3FD4E6AE33C9FC07CB308DB3B3C9D20ED6639CCA7033 0870553E5C414CA92619418661197FAC10471DB1D381085DDADDB58796829CA90 069 894896220765023255165660281515915342216260964409835451134459718720005 701041341852837898173064352495985745139837002928058309421561388204397 3354392115544169
	R2modr		0xA794586A718407B095DF1B4C194B2E56723C37A22F16BBDFD7F9CC263B790D E3A6F230C72F0207E83EC64BD033B7627F0886B75895283DDDD2A3681ECDA816 71

Table 38-67. Special Values for common ECC F_{2^m} domains when PKHA digit size is 16 bits

Name	L	N	
	var		Value (hex, decimal, sums of powers)
sect113r1	15	15	
wtls4	q		0x020000000000000000000000000201 $x^{113} + x^9 + 1$
	R2modq		0x00000000000000000001000040000000
	b		0xE8BEE4D3E2260744188BE0E9C723
	c		0x0173E834AF28EC76CB83BD8DFEB2D5

Table continues on the next page...

Table 38-67. Special Values for common ECC F_{2^m} domains when PKHA digit size is 16 bits (continued)

Name	L	N	
	var		Value (hex, decimal, sums of powers)
	r		0x0100000000000000D9CCEC8A39E56F
			5192296858534827689835882578830703
	R2modr		0x002D02609ABE76F866BDCE5B3F9BCC
sect113r2	15	15	
	q		0x020000000000000000000000000201
			$x^{113} + x^9 + 1$
	R2modq		0x00000000000000000001000040000000
	b		0x95E9A9EC9B297BD4BF36E059184F
	c		0x0054D9F03957174A32329167D7FE71
	r		0x0100000000000000108789B2496AF93
			5192296858534827702972497909952403
	R2modr		0x00471CB662E29CB41ABC888E16FF49
wtls1	15	14	
	q		0x020000000000000000000000000201
			$x^{113} + x^9 + 1$
	R2modq		0x00000000000000000001000040000000
	b		0x01
	c		0x00000000000000000000000000000001
	r		0xFFFFFFFFFFFFFFFFDBF91AF6DEA73
			5192296858534827627896703833467507
	R2modr		0x000511F09A1AB7E0A600B6A46FA9
sect131r1	17	17	
	q		0x080000000000000000000000000000010D
			$x^{131} + x^8 + x^3 + x^2 + 1$
	R2modq		0x00000000000000000000000040144000000
	b		0x0217C05610884B63B9C6C7291678F9D341
	c		0x03DB89B405E491160E3B2F07B0CE20B37E
	r		0x040000000000000000023123953A9464B54D
			136112946768375385389393275685365560653
	R2modr		0x03CE363344739BBCD370FB96ADB81025B9
sect131r2	17	17	
	q		0x080000000000000000000000000000010D
			$x^{131} + x^8 + x^3 + x^2 + 1$
	R2modq		0x00000000000000000000000040144000000
	b		0x04B8266A46C55657AC734CE38F018F2192
	c		0x07CBB9920D71A48E099C38D71DA6490EB1
	r		0x04000000000000000016954A233049BA98F

Table continues on the next page...

Table 38-67. Special Values for common ECC F_{2^m} domains when PKHA digit size is 16 bits (continued)

Name	L	N	
	var		Value (hex, decimal, sums of powers)
			136112946768375385387953504341281286798 3
	R2modr		0x02EBB3EA65BB896320D34767E38BE77465
Oakley 3	20	-	
	q		0x0800000000000000000000004000000000000001 $x^{155} + x^{62} + 1$
	R2modq		0x0000004000000000000000000000000000000400
	b		0x07338F
	c		0x00311000000223A000C4474000088E8000111D1D
B-163	21	21	
ansix9t163r2	q		0x0800000000000000000000000000000000000000C9 $x^{163} + x^7 + x^6 + x^3 + 1$
sect163r2			
EC2NGF163Random	R2modq		0x0000000000000000000000000000014104000000
	b		0x020A601907B8C953CA1481EB10512F78744A3205FD
	c		0x072C4E1EF7CB2F3A035D3310429415960913 8BB404
	r		0x040000000000000000000292FE77E70C12A4234C33 584600654932361167281474244287639068925684320 1587
	R2modr		0x011E0B46683488BE6C9C55E513CB679874C4E0E0B1
K-163	21	21	
ansix9t163k1	q		0x0800000000000000000000000000000000000000C9 $x^{163} + x^7 + x^6 + x^3 + 1$
sect163k1			
EC2NGF163Koblitz	R2modq		0x0000000000000000000000000000014104000000
wtls3	b		0x01
	c		0x0000000000000000000000000000000000000001
	r		0x040000000000000000020108A2E0CC0D99F8A5EF 584600654932361167281474175359844834832911 8574063
	R2modr		0x013877C8B9719E20D16A359191E004B3D7105A3DB4
sect163r1	21	21	
ansix9t163r1	q		0x0800000000000000000000000000000000000000C9 $x^{163} + x^7 + x^6 + x^3 + 1$
	R2modq		0x0000000000000000000000000000014104000000
	b		0x0713612DCDDCB40AAB946BDA29CA91F73AF958AFD9
	c		0x05ED403ED58EB45B1CCECA0F4F61655549861BE052
	r		0x03FFFFFFFFFFFFFFFFFFF48AAB689C29CA710279B 584600654932361167281473846509879898 1304420411291
	R2modr		0x030E95D72BF45AD7608E54BE73B44B4B1C327B65C1
wtls5	21	21	
	q		0x0800000000000000000000000000000000000000107

Table continues on the next page...

Table 38-67. Special Values for common ECC F_{2^m} domains when PKHA digit size is 16 bits (continued)

Name	L	N	
	var		Value (hex, decimal, sums of powers)
			$x^{163} + x^8 + x^2 + x^1 + 1$
	R2modq		0x0000000000000000000000000000040054000000
	b		0xC9517D06D5240D3CFF38C74B20B6CD4D6F9DD4D9
	c		0x0453E1E4B7291F5C2D53CE18483F007081E7EA26EC
	r		0x040000000000000000001E60FC8821CC74DAEAFC1
			584600654932361167281474162622639205657383263840 1
	R2modr		0x007218FA96704CFBABEA28DE5A57F5A5C73FE73E87
Oakley 4	24	-	
	q		0x020000000000000000000000000002000000000000000001
			$x^{185} + x^{69} + 1$
	R2modq		0x000000001000000000000000000000000000000004000
	b		0x1EE9
	c		0x000000000003000000018000C00C000000638030001C0009
sect193r1	25	25	
ansix9t193r1	q		0x020000000000000000000000000000000000000000008001
			$x^{193} + x^{15} + 1$
	R2modq		0x000000000000000000000000000000001000000040000000
	b		0xFDFB49BFE6C3A89FACADAA7A1E5BBC7CC1C2E5D831478814
	c		0x0167B35EB4313F263D0F7A3D5036F0A0A3C980D40E5A053ED2
	r		0x010000000000000000000000C7F34A778F443ACC920EBA49
			6277101735386680763835789423269548053691575186051040197193
	R2modr		0x00555F659B9F0A6812CD5A09619A2E7446C7F16EDAA6C25C24
sect193r2	25	25	
ansix9t193r2	q		0x020000000000000000000000000000000000000000008001
			$x^{193} + x^{15} + 1$
	R2modq		0x000000000000000000000000000000001000000040000000
	b		0xC9BB9E8927D4D64C377E2AB2856A5B16E3EFB7F61D4316AE
	c		0x006989FE6BFE30EDDC3244269F3AAD18D66CF3DB3E3302FAA8
	r		0x010000000000000000000000015AAB561B005413CCD4EE99D5
			6277101735386680763835789423314955362437298222279840143829
	R2modr		0x0021EBDCE9B24356750478EA3C9B8611E749E5446745FCDD4F
B-233	30	30	
sect233r1	q		0x020000000000000000000000000000000000000004000000000000000001
ansix9t233r1			$x^{233} + x^{74} + 1$
EC2NGF233Random	R2modq		0x000000000000000000040000000000000000000000000000000000004000
wtls11	b		0x66647EDE6C332C7F8C0923BB58213B333B20E9CE4281FE115F7D8F90AD
	c		0x0007D5EF4389DFF11ECDBA39C30970D3CE35CEBBA58473F64B4DC0F2686C

Table continues on the next page...

Table 38-67. Special Values for common ECC F_{2^m} domains when PKHA digit size is 16 bits (continued)

Name	L	N	
	var		Value (hex, decimal, sums of powers)
	r		0x0100000000000000000000000000013E974E72F8A6922031D2603CFE0D7 690174634679056378743475586227702555583981273734501355537938363448546 3
	R2modr		0x00405345696AB044AA57CDD6D0CC913DB0DBA856F036B34D7EDB8C1383DC
K-233	30	29	
sect233k1 ansix9t233k1 EC2NGF233Koblitz wtls10	q		0x020000000000000000000000000000000000000004000000000000000001 $x^{233} + x^{74} + 1$
	R2modq		0x000000000000000000040000000000000000000000000000000004000
	b		0x01
	c		0x000000000000000000000000000000000000000000000000000000000001
	r		0x800000000000000000000000000069D5BB915BCD46EFB1AD5F173ABDF 345087317339528189371737793113851276057094098886225212632808702474134 3
	R2modr		0x193B9A7B59BEBED80293C813EEB5B6D7DB65FF0B4A6545982B006BBF08
sect239k1 ansix9t239k1	30	30	
	q		0x800000000000000000004000000000000000000000000000000000000001 $x^{239} + x^{158} + 1$
	R2modq		0x200000000000000000000000000000000000008000000000000000004
	b		0x01
	c		0x000000000000000000000000000000000000000000000000000000000001
	r		0x2000000000000000000000000005A79FEC67CB6E91F1C1DA800E478A5 220855883097298041197912187592864814948216561321709848887480219215362 213
	R2modr		0x1E805199383E8C975E5EA68E20339652293F129E7CF3AB36516F788CE5A5
B-283	36	36	
sect283r1 ansix9t283r1 EC2NGF283Random	q		0x0800000000000000000000000000000000000000000000000000000000000000000 010A1 $x^{283} + x^{12} + x^7 + x^5 + 1$
	R2modq		0x0000000000000000000000000000000000000000000000000000000000000000401100400
	b		0x027B680AC8B8596DA5A4AF8A19A0303FCA97FD7645309FA2A581485AF6263E313B79A2F5
	c		0x03D8C93D3B0EA81D9294034D7EE3135D0AC5FC8D9CB0276F7211F880F0D81CA4C6E87B38
	r		0x03FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEF90399660FC938A90165B042A7CEFADB307 77706755689029162836778476272940756265696259243769048891091965267700442777787378692871
	R2modr		0x0299ADD3FE013DB2E23755FAA9545A49222D8461643773D41D288BCA1EBB695767D7CA78

Table continues on the next page...

Table 38-67. Special Values for common ECC F_{2^m} domains when PKHA digit size is 16 bits (continued)

Name	L	N	
	var		Value (hex, decimal, sums of powers)
K-283 sect283k1 ansix9t283k1 EC2NGF283Koblitz	36	36	
	q		0x0800000000000000000000000000000000000000000000000000000000000010A1 $x^{283} + x^{12} + x^7 + x^5 + 1$
	R2modq		0x0000000000000000000000000000000000000000000000000000000000000401100400
	b		0x01
	c		0x0000000000000000000000000000000000000000000000000000000000000000000001
	r		0x01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE9AE2ED07577265DFF7F94451E061E163C61 3885337784451458141838923813647037813284811733793061324295874997529815829704422603873
	R2modr		0x00868DB4F6424B4B297831F5EBA11AE2B1AC177F33C6C133859892757E41B7CAE8927BDB
B-409 sect409r1 ansix9t409r1 EC2NGF409Random	52	52	
	q		0x020000000000000000000000000000000000000000000000000000000000000000000000000000000000000000800000000000000000001 $x^{409} + x^{87} + 1$
	R2modq		0x000000000000000000000000000000000000000000000000000000000000001000000000000000000000000000000000000000000000004000
	b		0x21A5C2C8EE9FEB5C4B9A753B7B476B7FD6422EF1F3DD674761FA99D6AC27C8A9A197B272822F6CD57A55AA4F50AE317B13545F
	c		0x0149B8B7BEBD9B63653EF1CD8C6A5DD105A2AAAC36FE2EAE43CF28CE1CB7C830C1ECDBFA413AB07FE35A57811AE4F88D30AC63FB
	r		0x010000000000000000000000000000000000000000000000001E2AAD6A612F33307BE5FA47C3C9E052F838164CD37D9A21173 661055968790248598951915308032771039828404682964281219284648798304157774827374805208143723762179110965979867288366567526771
	R2modr		0x0007C24B27E70A941D4738F415F186A66A9FF8783C798E99D4A152BF0CE0C0CC273CDA3D70AA82C43A336EFBE1479034DB8EF936
K-409 sect409k1 ansix9t409k1 EC2NGF409Koblitz	52	51	
	q		0x020000000000000000000000000000000000000000000000000000000000000000000000000000000000000000800000000000000000001 $x^{409} + x^{87} + 1$
	R2modq		0x000000000000000000000000000000000000000000000000000000000000001000000000000000000000000000000000000000000000004000
	b		0x01
	c		0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001
	r		0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE5F83B2D4EA20400EC4557D5ED3E3E7CA5B4B5C83B8E01E5FCF

Table continues on the next page...

Table 38-67. Special Values for common ECC F_{2^m} domains when PKHA digit size is 16 bits (continued)

Name	L	N	
	var		Value (hex, decimal, sums of powers)
			330527984395124299475957654016385519914202341482140609642324395022880 711289249191050673258457777458014096366590617731358671
	R2modr		0x50DC9B805D4BEBA0701EDA0D529DAD74A3ED9914801EFC3F5D0760180600F3 725B714A1C6E7B3C68A06DF3709E9354226F8D6C

38.7 LTC AES Examples

Example AES ECB Operation:

1. Write key to Primary Key Register.
2. Write key size to Primary Key Size Register.
   - Key Size is 16 bytes (00000010h)
   - Key Size is 24 bytes (00000018h)
   - Key Size is 32 bytes (00000020h)
3. Write Mode to Primary Mode Register. (0010020Dh)
4. Write Size of data to encrypt/decrypt to Data Size Register.
5. Write data into the Input FIFO.
6. Read data from the Output FIFO.
7. Interrupt is generated after final word is pushed to output FIFO.

Example AES CTR Operation:

1. Write key to Primary Key Register.
2. Write key size to Primary Key Size Register.
   - Key Size is 16 bytes (00000010h)
   - Key Size is 24 bytes (00000018h)
   - Key Size is 32 bytes (00000020h)
3. Write initial counter value to Context Words 4-7 in the Context Register.
4. Write Mode to Primary Mode Register. (0010000Dh)
5. Write Size of data to encrypt/decrypt to Data Size Register.
6. Write data into the Input FIFO.
7. Read data from the Output FIFO.
8. Interrupt is generated after final word is pushed to output FIFO.

Example AES CCM or CCM* Operation:

1. Write key to Primary Key Register.
2. Write key size to Primary Key Size Register.

- Key Size is 16 bytes (00000010h)
- Key Size is 24 bytes (00000018h)
- Key Size is 32 bytes (00000020h)

3. Write B0 value to Context Words 0-3 in the Context Register.
4. Write initial counter value to Context Words 4-7 in the Context Register.
5. Write Mode to Primary Mode Register. (0010080Dh) CCM and CCM* both use the same mode value.
   - CCM and CCM* both use the same mode value.
6. Write Size of Authentication Only Data to the AAD Size Register.
7. Write Authentication Only data to the Input FIFO.
   - Authentication data needs to be padded to a 16 byte boundary with zeros.
   - For example if there is 8 bytes of AAD then (00000008h) should be written to AAD Size register and 8 bytes of AAD data followed by 8 bytes of Zero should be written into the Input FIFO.
8. Write Size of data to encrypt/decrypt and authenticate to Data Size Register.
9. Write data into the Input FIFO.
10. Read data from the Output FIFO.
11. Interrupt is generated after final word is pushed to output FIFO.
12. Read MAC from Context Registers
    - MAC is read from Context Registers 0-3.
    - Encrypted MAC is read from Context Registers 8-11.

Example AES CCM or CCM* Authentication Only Operation:

1. Write key to Primary Key Register.
2. Write key size to Primary Key Size Register.
   - Key Size is 16 bytes (00000010h)
   - Key Size is 24 bytes (00000018h)
   - Key Size is 32 bytes (00000020h)
3. Write B0 value to Context Words 0-3 in the Context Register.
4. Write initial counter value to Context Words 4-7 in the Context Register.
5. Write Mode to Primary Mode Register. (0010080Dh).
   - CCM and CCM* both use the same mode value.
6. Write Size of Authentication Only Data to the AAD Size Register.
   - The AL bit needs to be set in the AAD Size Register. This tells the AES core engine that it will recieve only Authentication Data. Note for encryption only the mechanism is handled automatically.
7. Write Authentication Only data to the Input FIFO.
   - Authentication data needs to be padded to a 16 byte boundary with zeros.
   - For example if there is 8 bytes of AAD then (00000008h) should be written to AAD Size register and 8 bytes of AAD data followed by 8 bytes of Zero should be written into the Input FIFO.

8. Write data into the Input FIFO.
9. Interrupt is generated after final word is processed from input FIFO.
10. Read MAC from Context Registers
    - MAC is read from Context Registers 0-3.
    - Encrypted MAC is read from Context Registers 8-11.

Example AES CCM or CCM* Encryption Only Operation:

1. Write key to Primary Key Register.
2. Write key size to Primary Key Size Register.
    - Key Size is 16 bytes (00000010h)
    - Key Size is 24 bytes (00000018h)
    - Key Size is 32 bytes (00000020h)
3. Write B0 value to Context Words 0-3 in the Context Register.
4. Write initial counter value to Context Words 4-7 in the Context Register.
5. Write Mode to Primary Mode Register. (0010080Dh).
    - CCM and CCM* both use the same mode value.
6. Write Size of data to encrypt/decrypt to Data Size Register.
7. Write data into the Input FIFO.
8. Read data from the Output FIFO.
9. Interrupt is generated after final word is pushed into the output FIFO.

38.8 LTC DES Examples

Example DES ECB Decrypt Operation:

1. Write key to Primary Key Register.
2. Write key size to Primary Key Size Register. (00000008h)
3. Write Mode to Primary Mode Register. (0020020Ch)
4. Write Size of data to encrypt/decrypt to Data Size Register.
5. Write data into the Input FIFO.
6. Read data from the Output FIFO.
7. Interrupt is generated after final word is pushed to output FIFO.

Example 3DES CBC Encrypt Operation:

1. Write key to Primary Key Register.
2. Write key size to Primary Key Size Register.
    - Key Size is 16 bytes (00000010h)
    - Key Size is 24 bytes (00000018h)
3. Write IV to Context Words 0-1 in the Context Register.
4. Write Mode to Primary Mode Register. (0021010Dh)

5. Write Size of data to encrypt/decrypt to Data Size Register.
6. Write data into the Input FIFO.
7. Read data from the Output FIFO.
8. Interrupt is generated after final word is pushed to output FIFO.

38.9 LTC MDHA Examples

Example MDHA SHA256 Mode:

1. Write SHA256 Mode and Init/Final Mode to Primary Mode Register.
2. Write data size to Data Size Register.
3. Write data to Input FIFO.
4. Interrupt is generated after final word is written to Input FIFO.
5. Read final digest from the Context Register.

Example MDHA Multiple Groups of Data:

1. Write SHA256 Mode and Init Mode to Primary Mode Register.
2. Write data size to Data Size Register.
3. Write data to Input FIFO.
4. Interrupt is generated after final word of this group is written.
5. Read and save intermediate digest from Context Register.
6. Write SHA256 Mode and Update Mode to Primary Mode Register.
7. Load intermediate digest to Context Register.
8. Write data size to Data Size Register.
9. Write data to Input FIFO.
10. Interrupt is generated after final word of this group is written.
11. Read and save intermediate digest from Context Register.
12. Write SHA256 Mode and Final Mode to Primary Mode Register.
13. Load intermediate digest to Context Register.
14. Write data size to Data Size Register.
15. Write data to Input FIFO.
16. Interrupt is generated after final word of this group is written.
17. Read final digest from Context Register.

38.10 LTC PKHA Examples

Example PKHA MOD_ADD Operation:

1. Write N size to PKHA N Size Register.
2. Write A Size to PKHA A Size Register.

3. Write B Size to PKHA B Size Register.
4. Write Modulus to PKHA N0 Register.
5. Write first addend to PKHA A0 Register.
6. Write second addend to PKHA B0 Register.
7. Write Mode to Primary Mode Register. (00800002h)
8. Wait for interrupt to indicate 'done'.
9. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register.
10. Read the result of MOD_ADD from PKHA B0 Register.
11. If desired, read LTC Status Register, bits 30 and 29, to check PKZ, PKO flags.

Example PKHA F2M_MUL, result to PKHA A, Operation:

1. Write N size to PKHA N Size Register.
2. Write A Size to PKHA A Size Register.
3. Write B Size to PKHA B Size Register.
4. Write Modulus to PKHA N0 Register.
5. Write first multiplicand to PKHA A0 Register.
6. Write second multiplicand to PKHA B0 Register.
7. Write Mode to Primary Mode Register. (00820105h)
8. Wait for interrupt to indicate 'done'.
9. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register.
10. Read the result of F2M_MUL from PKHA A0 Register.
11. If desired, read LTC Status Register, bits 30 and 29, to check PKZ, PKO flags.

Example PKHA ECC_MOD_MUL Operation:

1. Write N size to PKHA N Size Register.
2. Write A Size to PKHA A Size Register.
3. Write B Size to PKHA B Size Register.
4. Write E Size to PKHA E Size Register.
5. Write Modulus to PKHA N0 Register.
6. Write multiplicand X coordinate to PKHA A0 Register.
7. Write multiplicand Y coordinate to PKHA A1 Register.
8. Write elliptic curve "a" parameter to PKHA A3 Register.
9. Write elliptic curve "b" parameter to PKHA B0 Register.
10. Write scalar multiplier "k" to PKHA E Register.
11. Write Mode to Primary Mode Register. (0080000Bh)
12. Wait for interrupt to indicate 'done'.
13. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register.
14. Read the resulting point X from PKHA B1 Register.

15. Read the resulting point Y from PKHA B2 Register.
16. Read LTC Status Register bit 30, to check PKZ flag. If set, this indicates the result is point at infinity.

Example PKHA MOD_R2 Operation:

1. Write N size to PKHA N Size Register.
2. Write Modulus to PKHA N0 Register.
3. Write Mode to Primary Mode Register. (0080000Ch)
4. Wait for interrupt to indicate 'done'.
5. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register.
6. Read the result of MOD_R2 from PKHA B0 Register. This is R^2modN.

Example PKHA ECC_F2M_MUL_R2_TEQ, result to PKHA A, Operation:

1. Write N size to PKHA N Size Register.
2. Write A Size to PKHA A Size Register.
3. Write B Size to PKHA B Size Register.
4. Write E Size to PKHA E Size Register.
5. Write Modulus to PKHA N0 Register.
6. Write multiplicand X coordinate to PKHA A0 Register.
7. Write multiplicand Y coordinate to PKHA A1 Register.
8. Write elliptic curve "a" parameter to PKHA A3 Register.
9. Write elliptic curve "c" parameter to PKHA B0 Register.
10. Write pre-computed R^2modN to PKHA B1 Register.
11. Write scalar multiplier "k" to PKHA E Register.
12. Write Mode to Primary Mode Register. (0083050Bh)
13. Wait for interrupt to indicate 'done'.
14. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register.
15. Read the resulting point X from PKHA A0 Register.
16. Read the resulting point Y from PKHA A1 Register.
17. Read LTC Status Register bit 30, to check PKZ flag. If set, this indicates the result is point at infinity.

Example PKHA Miller-Rabin Primality Test Operation:

1. Write N size to PKHA N Size Register.
2. Write A Size to PKHA A Size Register.
3. Write B Size to the value 1, to the PKHA B Size Register.
4. Write the prime candidate to PKHA N0 Register.
5. Write the random seed, range (1 < A < N-1) to PKHA A0 Register.
6. Write the number of trials PKHA B0[7:0] Register.

7. Write Mode to Primary Mode Register. (0080000Fh)
8. Wait for interrupt to indicate 'done'.
9. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register.
10. Read the result of the primality test from PKHA B0 Register. It will be "1" if the candidate is believed to be prime, else 0.
11. Alternatively, read LTC Status Register bit 28, to check the PKP flag, which will be set if the candidate is believed to be prime.

Example PKHA Performing ECDSA Sign Protocol.

In this example, a series of operations are run using PKHA to perform ECDSA Signature Generation (Sign). The sequence of operations in this example is:

1. Compute u = Nonce mod r (reduce Nonce random value).
2. Compute h = 1/u mod r.
3. Compute V = u*G, the public key for u.
4. Compute c = V_x mod r.
5. Compute s*c mod r.
6. Compute f + s*c mod r
7. Compute d = (h * (f + s*c)) mod r.

And the PKHA operations to be performed for ECDSA Sign Protocol are:

1. Write size of r to PKHA N Size Register.
2. Write size of Nonce to PKHA A Size Register.
3. Write value r to PKHA N Register.
4. Write the random Nonce value to PKHA A0 Register.
5. Write Mode command for AmodN, result to A, to Primary Mode Register (00800107h). This computes u = Nonce mod r.
6. Wait for interrupt to indicate 'done'.
7. Clear the interrupt by writing 10000h to LTC Status Register. PKHA A now contains u.
8. Write Mode command for COPY NSZ, Source A0, Destination N2, to Primary Mode Register (00800C90h). This saves u in PKHA N2.
9. Wait for interrupt to indicate 'done'.
10. Clear the interrupt by writing 10000h to LTC Status Register. PKHA N2 now contains u.
11. Write Mode command for MOD_INV to Primary Mode Register (00800008h). This computes h = 1/u mod r.
12. Wait for interrupt to indicate 'done'.
13. Clear the interrupt by writing 10000h to LTC Status Register. The also clears the Primary Mode Register. PKHA B now contains h.

14. Read the value h from PKHA B0 Register and save it for later use. The size of h is the r size, so this determines the number of bytes to read.
15. Write size of q to PKHA N Size Register.
16. Write value q to PKHA N Register.
17. Write size of q to PKHA A Size Register.
18. Write value a to PKHA A3 Register.
19. Write value G_x to PKHA A0 Register.
20. Write value G_y to PKHA A1 Register.
21. Write Mode command for COPY NSZ, Source N2, Destination B0, to Primary Mode Register (00800610h).
22. Wait for interrupt to indicate 'done'.
23. Clear the interrupt by writing 10000h to LTC Status Register. PKHA B0 now contains u.
24. Write Mode command for COPY NSZ, Source B, Destination E, to Primary Mode Register (00820810h).
25. Wait for interrupt to indicate 'done'.
26. Clear the interrupt by writing 10000h to LTC Status Register. PKHA E now contains u.
27. Write value b to PKHA B0 Register.
28. Write Mode command for ECC_MOD_MUL_TEQ to Primary Mode Register (0080040Bh), else if this is an F2m domain, the command is ECC_F2M_MUL_TEQ (0082040Bh).
29. Wait for interrupt to indicate 'done'.
30. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register. PKHA B1/B2 now contain V_x/V_y.
31. Write Mode command for COPY SSZ, Source B1, Destination A0, to Primary Mode Register (00820111h).
32. Wait for interrupt to indicate 'done'.
33. Clear the interrupt by writing 10000h to LTC Status Register. PKHA A0 now contains V_x.
34. Write the size of r to PKHA N Size Register.
35. Write the value r to PKHA N Register.
36. Write Mode command for AmodN to Primary Mode Register (00800007h).
37. Wait for interrupt to indicate 'done'.
38. Clear the interrupt by writing 10000h to LTC Status Register. PKHA B now contains the Signature result $c = V_x \bmod r$.
39. Read the value c from PKHA B0 Register and save it as the c result. This size of this value is r size.
40. Write the size of r to PKHA A Size Register.
41. Write the value s (Private Key) to PKHA A0 Register.
42. Write Mode command for MOD_MUL to Primary Mode Register (00800005h).

43. Wait for interrupt to indicate "done".
44. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register. PKHA B0 now contains (s*c) mod r.
45. Write the size of the message representative "f" to PKHA A Size Register.
46. Write the value f to PKHA A0 Register.
47. Write Mode command for MOD_ADD to Primary Mode Register (00800002h).
48. Wait for interrupt to indicate "done".
49. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register. PKHA B0 now contains (f + (s*c)) mod r.
50. Write the size of r to PKHA A Size Register.
51. Write the value h (saved earlier) to PKHA A0 Register.
52. Write Mode command for MOD_MUL to Primary Mode Register (00800005h).
53. Wait for interrupt to indicate "done".
54. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register. PKHA B0 now contains the Signature result d = (h * (f + (s*c))) mod r.
55. Read the value d from PKHA B0 Register and save it as the d result. This size of this value is r size.

Example PKHA Performing ECDSA Verify Protocol.

In this example, a series of operations are run using PKHA to perform ECDSA Verify. The sequence of operations in this example is:

1. Compute h = 1/d mod r.
2. Compute u1 = m*h mod r.
3. Compute u2 = c*h mod r.
4. Compute J = u2*W.
5. Compute K = u1*G.
6. Compute P = u1*G + u2*W.
7. Compute c' = P_x mod r.
8. Compare c' = c, if they match, verification is successful.

And the PKHA operations to be performed for ECDSA Verify Protocol are:

1. Write size of r to PKHA N Size Register.
2. Write size of r to PKHA A Size Register.
3. Write value r to PKHA N Register.
4. Write value d to PKHA A0 Register.
5. Write Mode command for MOD_INV to Primary Mode Register (00800008h).
6. Wait for interrupt to indicate 'done'.
7. Clear the interrupt by writing 10000h to LTC Status Register. PKHA B now contains h = 1/d mod r.

8. Read the value h from PKHA B0 Register and save it for later use. The size of h is the r size, so this determines the number of bytes to read.
9. Write Mode command for CLEAR_MEM A to Primary Mode Register (00880001h). This clears A in case r size > message representative size.
10. Wait for interrupt to indicate 'done'.
11. Clear the interrupt by writing 10000h to LTC Status Register.
12. Write size of m (message representative) to PKHA A Size Register.
13. Write value m to PKHA A0 Register.
14. Write Mode command for MOD_MUL to Primary Mode Register (00800005h).
15. Wait for interrupt to indicate 'done'.
16. Clear the interrupt by writing 10000h to LTC Status Register. PKHA B0 now contains u1 = m*h mod r.
17. Read the value u1 from PKHA B0 Register and save it for later use. The size of u1 is the r size, so this determines the number of bytes to read.
18. Write size of r to PKHA A Size Register.
19. Write value h (saved earlier) to PKHA B0 Register.
20. Write value c to PKHA A0 Register.
21. Write Mode command for MOD_MUL to Primary Mode Register (00800005h).
22. Wait for interrupt to indicate 'done'.
23. Clear the interrupt by writing 10000h to LTC Status Register. PKHA B now contains u2 = c*h mod r.
24. Write Mode command for COPY SSZ, Source B, Destination E, to Primary Mode Register (00820810h).
25. Wait for interrupt to indicate 'done'.
26. Clear the interrupt by writing 10000h to LTC Status Register. PKHA E now contains u2.
27. Write Mode command for CLEAR_MEM A, N to Primary Mode Register (00890001h). This clears A and N in case r size > q size.
28. Wait for interrupt to indicate 'done'.
29. Clear the interrupt by writing 10000h to LTC Status Register.
30. Write size of q to PKHA N Size Register.
31. Write value q to PKHA N Register.
32. Write size of q to PKHA A Size Register.
33. Write value a to PKHA A3 Register.
34. Write size of q to PKHA B Size Register.
35. Write value b to PKHA B0 Register.
36. Write public key x value to PKHA A0 Register.
37. Write public key y value to PKHA A1 Register.
38. Write Mode command for ECC_MOD_MUL, Destination A, to Primary Mode Register (0080010Bh), else if this is an F2m domain, the command is ECC_F2M_MUL, DST A (0082010Bh).

39. Wait for interrupt to indicate 'done'.
40. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register. PKHA A0/A1 now contain J_x/J_y = u2*W.
41. Read the value J_x from PKHA A0 Register and save it for later use. This size of this value is q size.
42. Read the value J_y from PKHA A1 Register and save it for later use. This size of this value is q size.
43. Write size of r to PKHA E Size Register.
44. Write value u1 (saved earlier) to PKHA E Register.
45. Write value a to PKHA A3 Register.
46. Write value b to PKHA B0 Register.
47. Write G_x value to PKHA A0 Register.
48. Write G_y value to PKHA A1 Register.
49. Write Mode command for ECC_MOD_MUL, Destination A, to Primary Mode Register (0080010Bh), else if this is an F2m domain, the command is ECC_F2M_MUL, DST A (0082010Bh).
50. Wait for interrupt to indicate 'done'.
51. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register. PKHA A0/A1 now contain K_x/K_y = u1*G.
52. Write size of q to PKHA B Size Register.
53. Write value J_x (saved earlier) to PKHA B1 Register.
54. Write value J_y (saved earlier) to PKHA B2 Register.
55. Write Mode command for ECC_MOD_ADD, Destination A, to Primary Mode Register (00800109h), else if this is an F2m domain, the command is ECC_F2M_ADD, DST A (00820109h).
56. Wait for interrupt to indicate 'done'.
57. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register. PKHA A0/A1 now contain P_x/P_y = J + K.
58. Write size of r to PKHA N Size Register.
59. Write the value r to PKHA N Register.
60. Write Mode command for AmodN to Primary Mode Register (00800007h).
61. Wait for interrupt to indicate 'done'.
62. Clear the interrupt by writing 10000h to LTC Status Register. PKHA B now contains the Verify result c' = P_x mod r.
63. Write the size of r to PKHA B Size Register.
64. Write the value c to PKHA A0 Register.
65. Write Mode command for F2M_ADD to Primary Mode Register (00820002h).
66. Wait for interrupt to indicate "done".
67. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register. PKHA B0 now contains zero if the verification is successful.

68. Read the result of the verification from PKHA B0 Register. It will be "0" if the verification is successful.
69. Alternatively, read LTC Status Register bit 30, to check the PKZ flag, which will be set if verification is successful.

Example PKHA Performing RSA Encryption Protocol.

In this example, a series of operations are run using PKHA to perform RSA Encryption from n, e. This is done by computing $g = f^e \bmod n$.

The PKHA operations to be performed for RSA Encryption Protocol are:

1. Write size of n to PKHA N Size Register.
2. Write size of e to PKHA E Size Register.
3. Write size of f (secret) to PKHA A Size Register.
4. Write value n to PKHA N Register.
5. Write value e to PKHA E Register.
6. Write value f to PKHA A0 Register.
7. Write Mode command for MOD_EXP, to Primary Mode Register (00800006h). This computes $g = f^e \bmod n$.
8. Wait for interrupt to indicate 'done'.
9. Clear the interrupt by writing 10000h to LTC Status Register. PKHA B now contains g.
10. Read the value g from PKHA B0 Register and save it as the RSA encrypt result. This size of this value is n size.

Example PKHA Performing RSA Decryption Format 1 (n, d) Protocol.

In this example, a series of operations are run using PKHA to perform RSA Decryption Format 1 (n, d). This is done by computing $f = g^d \bmod n$.

The PKHA operations to be performed for RSA Decryption (n, d) Protocol are:

1. Write size of n to PKHA N Size Register.
2. Write size of d to PKHA E Size Register.
3. Write size of g (encrypted secret) to PKHA A Size Register.
4. Write value n to PKHA N Register.
5. Write value d to PKHA E Register.
6. Write value g to PKHA A0 Register.
7. Write Mode command for MOD_EXP_TEQ, to Primary Mode Register (00800406h). This computes $f = g^d \bmod n$.
8. Wait for interrupt to indicate 'done'.
9. Clear the interrupt by writing 10000h to LTC Status Register. PKHA B now contains f.

10. Read the value f from PKHA B0 Register and save it as the RSA decrypt result. This size of this value is n size.

Example PKHA Performing RSA Decryption Format 2 (pqd) Protocol.

In this example, a series of operations are run using PKHA to perform RSA Decryption Format 2 (pqd). The sequence of operations in this example is:

1. Compute d2 = d mod (q - 1).
2. Compute j2 = g^{d2} mod q.
3. Compute d1 = d mod (p - 1).
4. Compute j1 = g^{d1} mod q.
5. Compute h = ((1 / q) * (j1 - j2)) mod p.
6. Compute j = j2 + (h * q)

And the PKHA operations to be performed for RSA Decryption Format 2 (pqd) Protocol are:

1. Write size of q to PKHA N Size Register.
2. Write value q to PKHA N Register.
3. Write Mode command for COPY NSZ, Source N0, Destination A0, to Primary Mode Register (00860010h). This saves q in PKHA A.
4. Wait for interrupt to indicate 'done'.
5. Clear the interrupt by writing 10000h to LTC Status Register. PKHA A now contains q.
6. Write the value 1 to PKHA B Size Register.
7. Write value 1 (one single byte) to PKHA B0 Register.
8. Write Mode command for MOD_SUB_1 to Primary Mode Register (00800003h). This computes q - 1.
9. Wait for interrupt to indicate 'done'.
10. Clear the interrupt by writing 10000h to LTC Status Register. The also clears the Primary Mode Register. PKHA B now contains q - 1.
11. Write Mode command for COPY NSZ, Source B0, Destination N0, to Primary Mode Register (00820C10h). This saves q - 1 in PKHA N.
12. Wait for interrupt to indicate 'done'.
13. Clear the interrupt by writing 10000h to LTC Status Register. PKHA N now contains q - 1.
14. Write size of d to PKHA A Size Register.
15. Write value d to PKHA A Register.
16. Write Mode command for AmodN to Primary Mode Register (00800007h).
17. Wait for interrupt to indicate 'done'.
18. Clear the interrupt by writing 10000h to LTC Status Register. PKHA B0 now contains d2 = d mod (q - 1).

19. Write Mode command for COPY NSZ, Source B, Destination E, to Primary Mode Register (00820810h).
20. Wait for interrupt to indicate 'done'.
21. Clear the interrupt by writing 10000h to LTC Status Register. PKHA E now contains d2.
22. Write value q to PKHA N Register.
23. Write Mode command for CLEAR_MEM A to Primary Mode Register (00880001h), in case size < q size.
24. Wait for interrupt to indicate 'done'.
25. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register.
26. Write the size of g to PKHA A Size Register.
27. Write the value g to PKHA A Register.
28. Write Mode command for AmodN, destination A, to Primary Mode Register (00800107h).
29. Wait for interrupt to indicate 'done'.
30. Clear the interrupt by writing 10000h to LTC Status Register. PKHA A now contains g mod q.
31. Write Mode command for MOD_EXP_TEQ to Primary Mode Register (00800406h).
32. Wait for interrupt to indicate "done".
33. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register. PKHA B0 now contains $j2 = g^{d2} \bmod q$.
34. Read the value j2 from PKHA B0 Register and save it for later use. This size of this value is q size.
35. Write the size of p to PKHA N Size Register.
36. Write the value p to PKHA N Register.
37. Write Mode command for COPY NSZ, Source N0, Destination A0, to Primary Mode Register (00860010h). This saves p in PKHA A.
38. Wait for interrupt to indicate 'done'.
39. Clear the interrupt by writing 10000h to LTC Status Register. PKHA A now contains p.
40. Write Mode command for CLEAR_MEM B to Primary Mode Register (00840001h).
41. Wait for interrupt to indicate 'done'.
42. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register.
43. Write the value 1 to PKHA B Size Register.
44. Write value 1 (one single byte) to PKHA B0 Register.
45. Write Mode command for MOD_SUB_1 to Primary Mode Register (00800003h). This computes p - 1.
46. Wait for interrupt to indicate 'done'.

47. Clear the interrupt by writing 10000h to LTC Status Register. The also clears the Primary Mode Register. PKHA B now contains p - 1.
48. Write Mode command for COPY NSZ, Source B0, Destination N0, to Primary Mode Register (00820C10h). This saves p - 1 in PKHA N.
49. Wait for interrupt to indicate 'done'.
50. Clear the interrupt by writing 10000h to LTC Status Register. PKHA N now contains p - 1.
51. Write size of d to PKHA A Size Register.
52. Write value d to PKHA A Register.
53. Write Mode command for AmodN to Primary Mode Register (00800007h).
54. Wait for interrupt to indicate 'done'.
55. Clear the interrupt by writing 10000h to LTC Status Register. PKHA B0 now contains d1 = d mod (p - 1).
56. Write Mode command for COPY NSZ, Source B, Destination E, to Primary Mode Register (00820810h).
57. Wait for interrupt to indicate 'done'.
58. Clear the interrupt by writing 10000h to LTC Status Register. PKHA E now contains d1.
59. Write value p to PKHA N Register.
60. Write size of g to PKHA A Size Register.
61. Write value g to PKHA A Register.
62. Write Mode command for AmodN, Destination A, to Primary Mode Register (00800107h).
63. Wait for interrupt to indicate "done".
64. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register. PKHA A now contains g mod p.
65. Write Mode command for MOD_EXP_TEQ to Primary Mode Register (00800406h).
66. Wait for interrupt to indicate "done".
67. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register. PKHA B0 now contains $j1 = g^{d1} \bmod p$.
68. Read the value j1 from PKHA B0 Register and save it for later use. This size of this value is p size.
69. Write the size of q to PKHA A Size Register.
70. Write the value j2 (saved earlier) to PKHA A0 Register.
71. Write Mode command for AmodN to Primary Mode Register (00800007h).
72. Wait for interrupt to indicate "done".
73. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register. PKHA B0 now contains j2 mod p.
74. If q size > p size, write Mode command for CLEAR_MEM A to Primary Mode Register (00880001h).
75. If q size > p size, wait for interrupt to indicate 'done'.

76. If q size > p size, clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register.
77. Write size of p to PKHA A Size Register.
78. Write value j1 (saved earlier) to PKHA A0 Register.
79. Write Mode command for MOD_SUB_1 to Primary Mode Register (00800003h).
80. Wait for interrupt to indicate "done".
81. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register. PKHA B0 now contains (j1 - j2) mod p.
82. Read the value (j1 - j2) mod p from PKHA B0 Register and save it for later use. This size of this value is p size.
83. Write size of q to PKHA A Size Register.
84. Write value q to PKHA A0 Register.
85. Write Mode command for AmodN, Destination A, to Primary Mode Register (00800107h).
86. Wait for interrupt to indicate "done".
87. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register. PKHA A0 now contains q mod p.
88. Write Mode command for MOD_INV to Primary Mode Register (00800008h).
89. Wait for interrupt to indicate "done".
90. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register. PKHA A0 now contains c = 1/q mod p.
91. Write the value (j1 - j2) mod p (saved earlier) to PKHA A0 Register.
92. Write Mode command for MOD_MUL to Primary Mode Register (00800005h).
93. Wait for interrupt to indicate "done".
94. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register. PKHA B0 now contains h = (c * (j1 - j2)) mod p.
95. Write Mode command for CLEAR_MEM A to Primary Mode Register (00880001h).
96. Wait for interrupt to indicate 'done'.
97. Clear the interrupt by writing 10000h to LTC Status Register. This also clears the Primary Mode Register.
98. Write the value 1 to PKHA A Size Register. This is needed to avoid an error on the next operation which would occur if A size = 0.
99. Write Mode command for COPY NSZ, Source B, Destination A, to Primary Mode Register (00820010h).
10 Wait for interrupt to indicate 'done'.
0.
10 Clear the interrupt by writing 10000h to LTC Status Register. This also clears the
1. Primary Mode Register.
10 Write Mode command for CLEAR_MEM B to Primary Mode Register (00840001h).
2.

10 Wait for interrupt to indicate 'done'.
3.
10 Clear the interrupt by writing 10000h to LTC Status Register. This also clears the
4. Primary Mode Register.
10 Write PKHA N size with the n size, rounded up to the next 8-byte value
5.
10 Write all ones to PKHA N Register, up to the size written in the previous step.
6.
10 Write size of q to PKHA B Size Register.
7.
10 Write the value q to PKHA B Register.
8.
10 Write Mode command for MOD_MUL_IM_OM to Primary Mode Register
9. (008C0005h).
11 Wait for interrupt to indicate "done".
0.
11 Clear the interrupt by writing 10000h to LTC Status Register. This also clears the
1. Primary Mode Register. PKHA B0 now contains h * q.
11 Write Mode command for CLEAR_MEM A to Primary Mode Register (00880001h).
2. This clears leftover bytes in PKHA A Register.
11 Wait for interrupt to indicate 'done'.
3.
11 Clear the interrupt by writing 10000h to LTC Status Register. This also clears the
4. Primary Mode Register.
11 Write size of q to PKHA A Size Register.
5.
11 Write the value j2 (saved earlier) to PKHA A Register.
6.
11 Write Mode command for MOD_ADD to Primary Mode Register (00800002h).
7.
11 Wait for interrupt to indicate "done".
8.
11 Clear the interrupt by writing 10000h to LTC Status Register. This also clears the
9. Primary Mode Register. PKHA B0 now contains the final result j = j2 + (h * q).
12 Read the value j from PKHA B0 Register and save it as the j result. This size of this
0. value is n size.

38.11 LTC General Examples

Writing and Reading Data from the FIFOs:

1. Writing and reading by polling operations.
   - The FIFO Status Register(LTCFIFOSTA) shows the number of entries in both the input and output FIFOs. It also shows when the FIFOs are full.
   - The input and output FIFOs support 4x32bit entries each.
   - Whenever there is space in the input FIFO the user can write a word into the Input FIFO.
   - Whenever there is a word in the output FIFO then the user can read a word from the Output FIFO.
2. Writing and reading FIFOs by DMA operations.
   - The on chip DMA will handle all reads and writes of the FIFOs.
   - The IDE and ODE bits in the Control Register must be written to enable the DMA handshake.
     - IDE will enable dma transfers to the input FIFO when there is space available.
     - ODE will enable dma transfers from the output FIFO when there are words in the FIFO.
   - The on chip DMA should then be programmed to write data to the input FIFO and read data from the output FIFO.

38.12 LTC Register Descriptions

All reads of write-only addresses always return zero. Writes to read-only addresses are ignored. LTC will generate a transfer error whenever an undefined address is read or written to on the register bus. Although many of the LTC registers hold more than 32 bits, the register addresses shown in the Memory Map below represent how these registers are accessed over the register bus as 32-bit words.

NOTE

The reset value of some registers differs between different versions of LTC. To ensure driver compatibility across different versions of LTC, when updating fields within registers, the registers should first be read, the required fields updated, and then the register should be written. This will avoid inadvertently changing the settings of other fields in the same register.

38.12.1 LTC Memory Map

Offset	Register	Width (In bits)	Access	Reset value
0h	LTC Mode (non-PKHA/non-RNG use) (LTC0_MD)	32	RW	00000000h
0h	LTC Mode (PublicKey) (LTC0_MDPK)	32	RW	00000000h
8h	LTC Key Size (LTC0_KS)	32	RW	00000000h
10h	LTC Data Size (LTC0_DS)	32	RW	00000000h
18h	LTC ICV Size (LTC0_ICVS)	32	RW	00000000h
30h	LTC Command (LTC0_COM)	32	WO	00000000h
34h	LTC Control (LTC0_CTL)	32	RW	00000000h
40h	LTC Clear Written (LTC0_CW)	32	WO	00000000h
48h	LTC Status (LTC0_STA)	32	W1C	01000000h
4Ch	LTC Error Status (LTC0_ESTA)	32	RO	00000000h
58h	LTC AAD Size (LTC0_AADSZ)	32	RW	00000000h
60h	LTC IV Size (LTC0_IVSZ)	32	RW	00000000h
68h	LTC DPA Mask Seed (LTC0_DPAMS)	32	WO	00000000h
80h	LTC PKHA A Size (LTC0_PKASZ)	32	RW	00000000h
88h	LTC PKHA B Size (LTC0_PKBSZ)	32	RW	00000000h
90h	LTC PKHA N Size (LTC0_PKNSZ)	32	RW	00000000h
98h	LTC PKHA E Size (LTC0_PKESZ)	32	RW	00000000h
100h - 13Ch	LTC Context (LTC0_CTX_0 - LTC0_CTX_15)	32	RW	00000000h
200h - 21Ch	LTC Keys (LTC0_KEY_0 - LTC0_KEY_7)	32	RW	00000000h
4F0h	LTC Version ID (LTC0_VID1)	32	RO	00340100h
4F4h	LTC Version ID 2 (LTC0_VID2)	32	RO	00000101h
4F8h	LTC CHA Version ID (LTC0_CHAVID)	32	RO	40450252h
7C0h	LTC FIFO Status (LTC0_FIFOSTA)	32	RO	00000000h
7E0h	LTC Input Data FIFO (LTC0_IFIFO)	32	WO	00000000h
7F0h	LTC Output Data FIFO (LTC0_OFIFO)	32	RO	00000000h
800h - 8FCh	LTC PKHA A (LTC0_PKA_0 - LTC0_PKA_63)	32	RW	00000000h
800h - 83Ch	LTC PKHA A0 (LTC0_PKA0_0 - LTC0_PKA0_15)	32	RW	00000000h
840h - 87Ch	LTC PKHA A1 (LTC0_PKA1_0 - LTC0_PKA1_15)	32	RW	00000000h
880h - 8BCh	LTC PKHA A2 (LTC0_PKA2_0 - LTC0_PKA2_15)	32	RW	00000000h
8C0h - 8FCh	LTC PKHA A3 (LTC0_PKA3_0 - LTC0_PKA3_15)	32	RW	00000000h
A00h - A3Ch	LTC PKHA B0 (LTC0_PKB0_0 - LTC0_PKB0_15)	32	RW	00000000h
A00h - AFCh	LTC PKHA B (LTC0_PKB_0 - LTC0_PKB_63)	32	RW	00000000h
A40h - A7Ch	LTC PKHA B1 (LTC0_PKB1_0 - LTC0_PKB1_15)	32	RW	00000000h
A80h - ABCh	LTC PKHA B2 (LTC0_PKB2_0 - LTC0_PKB2_15)	32	RW	00000000h
AC0h - AFCh	LTC PKHA B3 (LTC0_PKB3_0 - LTC0_PKB3_15)	32	RW	00000000h
C00h - C3Ch	LTC PKHA N0 (LTC0_PKN0_0 - LTC0_PKN0_15)	32	RW	00000000h
C00h - CFCh	LTC PKHA N (LTC0_PKN_0 - LTC0_PKN_63)	32	RW	00000000h
C40h - C7Ch	LTC PKHA N1 (LTC0_PKN1_0 - LTC0_PKN1_15)	32	RW	00000000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
C80h - CBCh	LTC PKHA N2 (LTC0_PKN2_0 - LTC0_PKN2_15)	32	RW	00000000h
CC0h - CFCh	LTC PKHA N3 (LTC0_PKN3_0 - LTC0_PKN3_15)	32	RW	00000000h
E00h - EFCh	LTC PKHA E (LTC0_PKE_0 - LTC0_PKE_63)	32	WO	00000000h

38.12.2 LTC Mode (PublicKey) (LTC0_MDPK)

38.12.2.1 Address

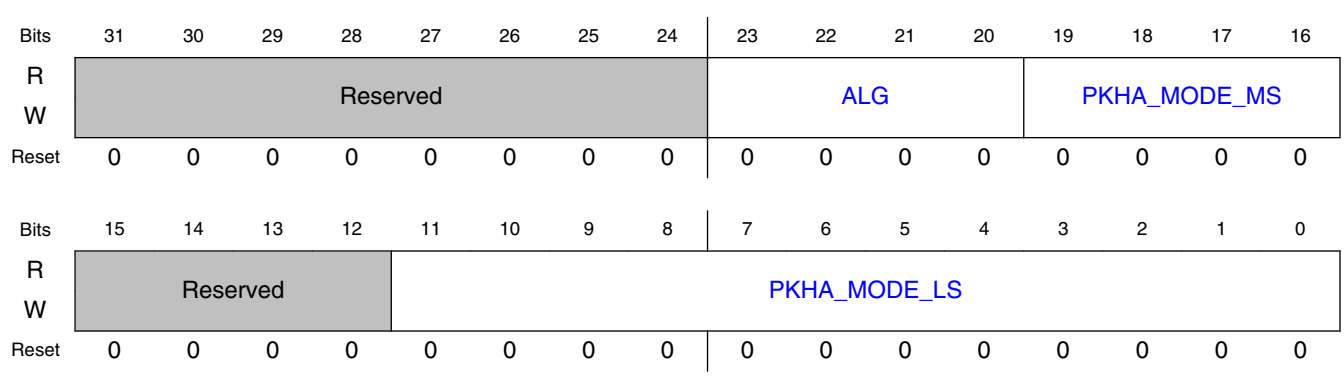
Register	Offset
LTC0_MDPK	0h

38.12.2.2 Function

The Mode Register is used to tell the CHAs which operation is being requested. The interpretation of this register is for public key operations (PKHA) only, and is described herein.

The bit assignments for the PKHA_MODE will be different depending on which of the three types of PKHA functions is being called. The three function types are: 1) Clear Memory, 2) Modular Arithmetic, and 3) Copy Memory. Detailed descriptions of their mode formats can be found in Table PKHA MODE: clear memory function, Table PKHA MODE: Arithmetic Functions and Table PKHA MODE: copy memory functions.

38.12.2.3 Diagram



38.12.2.4 Fields

Field	Function
31-24 —	Options bits.
23-20 ALG	Algorithm. This field specifies which algorithm is being selected. 1000b - PKHA
19-16 PKHA_MODE_MS	PKHA_MODE most-significant 4 bits. The format of the PKHA_MODE field differs depending on which of the three types of PKHA functions is being executed. The three function types are: 1) Clear Memory, 2) Modular Arithmetic, and 3) Copy Memory. Detailed descriptions of their mode formats can be found in Table PKHA MODE: clear memory function, Table PKHA MODE: Arithmetic Functions and Table PKHA MODE: copy memory functions.
15-12 —	Reserved
11-0 PKHA_MODE_LS	PKHA_MODE least significant 12 bits. The format of the PKHA_MODE field differs depending on which of the three types of PKHA functions is being executed. The three function types are: 1) Clear Memory, 2) Modular Arithmetic, and 3) Copy Memory. Detailed descriptions of their mode formats can be found in Table PKHA MODE: clear memory function, Table PKHA MODE: Arithmetic Functions and Table PKHA MODE: copy memory functions.

38.12.3 LTC Mode (non-PKHA/non-RNG use) (LTC0_MD)

38.12.3.1 Address

Register	Offset
LTC0_MD	0h

38.12.3.2 Function

The Mode Register is used to tell the cryptographic engines which operation is being requested. The interpretation of this register will be unique for each CHA.

This section defines the format of the Mode Register when used with non-public-key algorithms and non-RNG operations.

38.12.3.3 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved			AAI									AS		ICV_ TEST	ENC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.12.3.4 Fields

Field	Function
31-24 —	Reserved. Must be 0.
23-16 ALG	Algorithm. This field specifies which algorithm is being selected. 00010000b - AES 00100000b - DES 00100001b - 3DES 01000001b - MDHA - SHA-1 01000010b - MDHA - SHA-224 01000011b - MDHA - SHA-256
15-13 —	Reserved. Must be 0.
12-4 AAI	Additional Algorithm information. This field contains additional mode information that is associated with the algorithm that is being executed. See also the section describing the appropriate CHA. **NOTE:** Some algorithms do not require additional algorithm information and in those cases this field should be all 0s.

Table 38-68. AAI Interpretation for AES Modes

[For AES the MSB of AAI is the DK (Decrypt Key) bit.]				
Code[1]	Interpretation		Code	Interpretation
00h	CTR (mod 2^{128})		80h	CCM, CCM*
10h	CBC		90h	GCM
20h	ECB		A0h	Reserved
30h	Reserved		B0h	Reserved
40h	Reserved		C0h	Reserved
50h	Reserved		D0h	Reserved

Table continues on the next page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
Table 38-68. AAI Interpretation for AES Modes (continued)
<table>
<tr><th colspan="5">[For AES the MSB of AAI is the DK (Decrypt Key) bit.]</th></tr>
<tr><th>Code[1]</th><th>Interpretation</th><th></th><th>Code</th><th>Interpretation</th></tr>
<tr><td>60h</td><td>CMAC</td><td></td><td>E0h</td><td>Reserved</td></tr>
<tr><td>70h</td><td>XCBC-MAC</td><td colspan="3"></td></tr>
<tr><td colspan="5">Setting the DK bit (i.e. ORing 100h with any AES code above) causes Key Register to be loaded with the AES Dcrypt key, rather than the AES Encrypt key.</td></tr>
</table>
1. The codes are mutually exclusive (i.e. they cannot be ORed with each other).
Table 38-69. AAI Interpretation for DES
<table>
<tr><th>Code[1]</th><th>Interpretation</th><th></th><th>Code[1]</th><th>Interpretation</th></tr>
<tr><td>10h</td><td>CBC</td><td></td><td>30h</td><td>CFB</td></tr>
<tr><td>20h</td><td>ECB</td><td></td><td>40h</td><td>OFB</td></tr>
<tr><td colspan="5">80h ORed with any DES code above: Check odd parity</td></tr>
</table>
1. The codes are mutually exclusive (i.e. they cannot be ORed with each other).
</td></tr>
<tr><td>3-2
AS</td><td>Algorithm State. This field defines the state of the algorithm that is being executed. This may not be used by every algorithm.
00b - Update
01b - Initialize
10b - Finalize
11b - Initialize/Finalize</td></tr>
<tr><td>1
ICV_TEST</td><td>ICV Checking / Test AES fault detection.
For algorithms other than AES ECB mode: ICV Checking
This bit selects whether the current algorithm should compare the known ICV versus the calculated ICV. This bit will be ignored by algorithms that do not support ICV checking.
0 - Don't compare
1 - Compare
For AES ECB mode: Test AES fault detection
In AES ECB mode, this bit activates fault detection testing by injecting bit level errors into AES core logic as defined in the first 128 bits of the context.
0 - Don't inject bit errors
1 - Inject bit errors</td></tr>
<tr><td>0
ENC</td><td>Encrypt/Decrypt. This bit selects encryption or decryption.
0b - Decrypt.
1b - Encrypt.</td></tr>
</table>

1. The codes are mutually exclusive (i.e. they cannot be ORed with each other).

38.12.4 LTC Key Size (LTC0_KS)

38.12.4.1 Address

Register	Offset
LTC0_KS	8h

38.12.4.2 Function

The Key Size Register is used to tell the crypto engine(AES) the size of the key that was loaded into the Key Register. The Key Size Register must be written after the key is written into the Key Register. Writing to the Key Size Register will prevent the user from modifying the Key Register. Only 16, 24, and 32 byte keys are supported.

38.12.4.3 Diagram

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

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

38.12.4.4 Fields

Field	Function
31-6 —	Reserved.
5-0 KS	Key Size. This is the size of a Key measured in bytes

38.12.5 LTC Data Size (LTC0_DS)

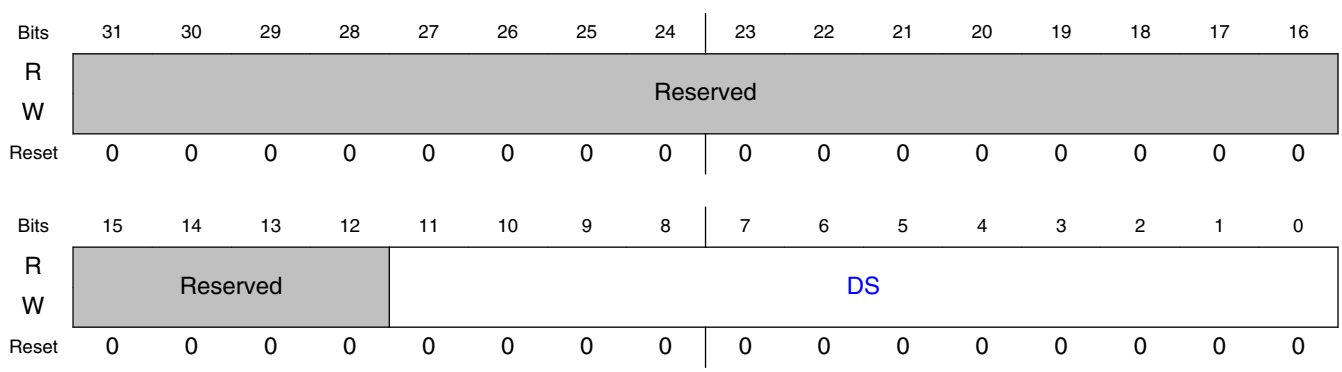
38.12.5.1 Address

Register	Offset
LTC0_DS	10h

38.12.5.2 Function

The Data Size Register is used to tell the AES the amount of data that will be loaded into the Input Data FIFO. This register should only be written to once during a single operation. Note that writing to the LTC AAD Size (LTC0_AADSZ) , or the LTC IV Size (LTC0_IVSZ), will cause this register to also update. When this register is then written directory to then the new value will be added to the previous value in the register. That is, if the DS field currently has the value 16, writing 2 to the least-significant half of the Data Size register (i.e. the DS field) will result in a value of 18 in the DS field. Note that AES decrements this register, so reading the register may return a value less than sum of the values that were written into it. This register is cleared whenever a key is decrypted or encrypted.

38.12.5.3 Diagram



38.12.5.4 Fields

Field	Function
31-12 —	Reserved.
11-0 DS	Data Size. This is the number of whole bytes of data that will be consumed by the CHA. Note that writing the AAD Size Register or IV Size Register will result in this register also being written to.

38.12.6 LTC ICV Size (LTC0_ICVS)

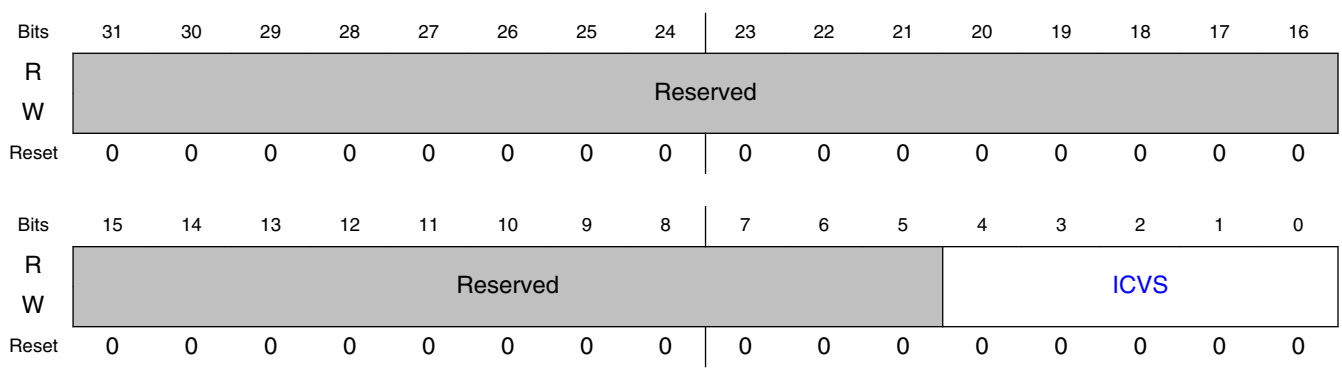
38.12.6.1 Address

Register	Offset
LTC0_ICVS	18h

38.12.6.2 Function

The ICV Size Register indicates how much of the last block of ICV is valid when performing AES integrity check modes (e.g. AES-CMAC, AES-XCBC-MAC). This register must be written prior to the corresponding word of data being consumed by AES. In practical terms, this means the register must be written prior to the corresponding data being written to the Input Data FIFO.

38.12.6.3 Diagram



38.12.6.4 Fields

Field	Function
31-5 —	Reserved.
4-0 ICVS	ICV Size, in Bytes.

38.12.7 LTC Command (LTC0_COM)

38.12.7.1 Address

Register	Offset
LTC0_COM	30h

38.12.7.2 Function

The LTC Command Register is used to send control signals to the Crypto Engines.

38.12.7.3 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved			Reserved		Reserved	Reserved	Reserved			Reserved	Reserved	Reserved			
									MD	PK				DES	AES	ALL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.12.7.4 Fields

Field	Function
31-29 —	Reserved. To preserve software compatibility with other versions of LTC, 0 should be written to all reserved bits.
28 —	Reserved.
27-13 —	Reserved.
12-11 —	Reserved
10 —	Reserved
9 —	Reserved
8 —	Reserved
7 MD	Reset MDHA. Writing a 1 to this bit resets the Message Digest Hardware Accelerator. 0b - Do Not Reset 1b - Reset Message Digest Hardware Accelerator
6 PK	Reset PKHA. Writing a 1 to this bit resets the Public Key Hardware Accelerator. 0b - Do Not Reset 1b - Reset Public Key Hardware Accelerator
5 —	Reserved
4 —	Reserved
3 —	Reserved.
2 DES	Reset DESA. Writing a 1 to this bit resets the DES Accelerator. 0b - Do Not Reset 1b - Reset DES Accelerator
1 AES	Reset AESA. Writing a 1 to this bit resets the AES Accelerator core engine. 0b - Do Not Reset 1b - Reset AES Accelerator
0 ALL	Reset All Internal Logic. Writing to this bit will reset all accelerator engines and as well as all the internal registers. 0b - Do Not Reset 1b - Reset all CHAs in use by this CCB.

38.12.8 LTC Control (LTC0_CTL)

38.12.8.1 Address

Register	Offset
LTC0_CTL	34h

38.12.8.2 Function

This register is used for some of the internal controls of the LTC block.

38.12.8.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	KAL	Reserved							COS	CIS	KOS	KIS	Reserved		OFS	IFS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		OFR	OFE	Reserved		IFR	IFE	Reserved			PDE	Reserved			IM
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.12.8.4 Fields

Field	Function
31 KAL	Key Register Access Lock. Read access to the key register is blocked. Any reads of the key register will only return zero. Once this bit is set, it can only be cleared by hard reset. 0b - Key Register is readable. 1b - Key Register is not readable.
30-24 —	Reserved.
23 COS	Context Register Output Byte Swap. Byte swap all data that is read from the context register. Data is byte swapped only within a single word. 0b - Do Not Byte Swap Data. 1b - Byte Swap Data.

Table continues on the next page...

Field	Function
22 CIS	Context Register Input Byte Swap. Byte swap all data that is written to the context register. Data is byte swapped only within a single word. 0b - Do Not Byte Swap Data. 1b - Byte Swap Data.
21 KOS	Key Register Output Byte Swap. Byte swap all data that is read from the key register. Data is byte swapped only within a single word. 0b - Do Not Byte Swap Data. 1b - Byte Swap Data.
20 KIS	Key Register Input Byte Swap. Byte swap all data that is written to the key register. Data is byte swapped only within a single word. 0b - Do Not Byte Swap Data. 1b - Byte Swap Data.
19-18 —	Reserved.
17 OFS	Output FIFO Byte Swap. Byte swap all data that is read from the Onput FIFO. 0b - Do Not Byte Swap Data. 1b - Byte Swap Data.
16 IFS	Input FIFO Byte Swap. Byte swap all data that is written to the Input FIFO. 0b - Do Not Byte Swap Data. 1b - Byte Swap Data.
15-14 —	Reserved.
13 OFR	Output FIFO DMA Request Size. The DMA request logic will only request data if the OUTPUT FIFO has enough data to satisfy the request. 0b - DMA request size is 1 entry. 1b - DMA request size is 4 entries.
12 OFE	Output FIFO DMA Enable. 0b - DMA Request and Done signals disabled for the Output FIFO. 1b - DMA Request and Done signals enabled for the Output FIFO.
11-10 —	Reserved.
9 IFR	Input FIFO DMA Request Size. The DMA request logic will only request data if the INPUT FIFO has enough space for the request size. 0b - DMA request size is 1 entry. 1b - DMA request size is 4 entries.
8 IFE	Input FIFO DMA Enable. 0b - DMA Request and Done signals disabled for the Input FIFO. 1b - DMA Request and Done signals enabled for the Input FIFO.
7-5 —	Reserved.
4 PDE	PKHA Register DMA Enable. 0b - DMA Request and Done signals disabled for the PKHA Registers. 1b - DMA Request and Done signals enabled for the PKHA Registers.
3-1	Reserved.

Table continues on the next page...

Field	Function
—	
0 IM	Interrupt Mask. Once this bit is set, it can only be cleared by hard reset. 0b - Interrupt not masked. 1b - Interrupt masked

38.12.9 LTC Clear Written (LTC0_CW)

38.12.9.1 Address

Register	Offset
LTC0_CW	40h

38.12.9.2 Function

The Clear Written Register is used to clear many of the internal registers. All fields of this register are self-clearing.

38.12.9.3 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R					Reserved							Reserved			Reserved	
W	CPKE	CPKN	CPKB	CPKA						CKR	CCR		CICV	CDS		CM
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.12.9.4 Fields

Field	Function
31	Clear Input FIFO. Writing a 1 to this bit causes the Input Data FIFO.

Table continues on the next page...

Field	Function
CIF	
30 COF	Clear Output FIFO. Writing a 1 to this bit causes the Output FIFO to be cleared.
29-28 —	Reserved.
27-16 —	Reserved.
15 CPKE	Clear the PKHA E Size Register. Writing a one to this bit causes the PKHA E Size Register to be cleared.
14 CPKN	Clear the PKHA N Size Register. Writing a one to this bit causes the PKHA N Size Register to be cleared.
13 CPKB	Clear the PKHA B Size Register. Writing a one to this bit causes the PKHA B Size Register to be cleared.
12 CPKA	Clear the PKHA A Size Register. Writing a one to this bit causes the PKHA A Size Register to be cleared.
11-7 —	Reserved.
6 CKR	Clear the Key Register. Writing a one to this bit causes the Key and Key Size Registers to be cleared.
5 CCR	Clear the Context Register. Writing a one to this bit causes the Context Register to be cleared.
4 —	Reserved.
3 CICV	Clear the ICV Size Register. Writing a one to this bit causes the ICV Size Register to be cleared.
2 CDS	Clear the Data Size Register. Writing a one to this bit causes the Data Size Register to be cleared. This clears AAD Size as well.
1 —	Reserved.
0 CM	Clear the Mode Register. Writing a one to this bit causes the Mode Register to be cleared.

38.12.10 LTC Status (LTC0_STA)

38.12.10.1 Address

Register	Offset
LTC0_STA	48h

38.12.10.2 Function

The LTC Status Register shows the status of the internal Crypto engine and its internal registers.

38.12.10.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	PKZ	PKO	PKP	Reserved			DPARRN	Reserved			EI	Reserved			DI
W																w1c
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved			Reserved		Reserved	Reserved	Reserved	MB	PB	Reserved	Reserved	Reserved	DB	AB	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.12.10.4 Fields

Field	Function
31 —	Reserved.
30 PKZ	Public Key Operation is Zero. For Finite Field operations the result of a Public Key operation is zero. For ECC operations, the result is Point at infinity.
29 PKO	Public Key Operation is One. The greatest common divisor of two numbers is one (that is, the two numbers are relatively prime) for GCD operation, or PKHA result = 1 for Finite Field operations.
28 PKP	Public Key is Prime. The given number is probably prime (that is, it passes the Miller-Rabin primality test).
27-25 —	Reserved.

Table continues on the next page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>24
DPARRN</td><td>This bit is asserted after POR and after every 50K blocks processed by AESA and DESA to indicate it is advisable for added security to write a new seed to LTC DPA Mask Seed (LTC0_DPAMS).</td></tr>
<tr><td>23-21
—</td><td>Reserved.</td></tr>
<tr><td>20
EI</td><td>Error Interrupt. The Error Interrupt has been asserted. This error can only be cleared by resetting LTC.
0b - Not Error.
1b - Error Interrupt.</td></tr>
<tr><td>19-17
—</td><td>Reserved.</td></tr>
<tr><td>16
DI</td><td>Done Interrupt. The Done Interrupt has been asserted.
<table>
<tr><th>Value</th><th>Read</th><th>Write</th></tr>
<tr><td>0</td><td>No Done Interrupt</td><td>No change</td></tr>
<tr><td>1</td><td>Done Interrupt asserted</td><td>Clear the Done Interrupt</td></tr>
</table></td></tr>
<tr><td>15-13
—</td><td>Reserved.</td></tr>
<tr><td>12-11
—</td><td>Reserved</td></tr>
<tr><td>10
—</td><td>Reserved</td></tr>
<tr><td>9
—</td><td>Reserved</td></tr>
<tr><td>8
—</td><td>Reserved</td></tr>
<tr><td>7
MB</td><td>MDHA Busy. This bit indicates that the MDHA is busy. The CHA can either be busy processing data or resetting.
0b - MDHA Idle
1b - MDHA Busy</td></tr>
<tr><td>6
PB</td><td>PKHA Busy. This bit indicates that the Public Key Hardware Accelerator is busy. The CHA can either be busy processing data or resetting.
0b - PKHA Idle
1b - PKHA Busy.</td></tr>
<tr><td>5
—</td><td>Reserved</td></tr>
<tr><td>4
—</td><td>Reserved</td></tr>
<tr><td>3
—</td><td>Reserved.</td></tr>
<tr><td>2</td><td>DESA Busy. This bit indicates that the DES Accelerator is busy. The CHA can either be busy processing data or resetting.</td></tr>
</table>

Table continues on the next page...

Field	Function
DB	0b - DESA Idle 1b - DESA Busy.
1 AB	AESA Busy. This bit indicates that the AES Accelertor is busy. The CHA can either be busy processing data or resetting. 0b - AESA Idle 1b - AESA Busy.
0 —	Reserved.

38.12.11 LTC Error Status (LTC0_ESTA)

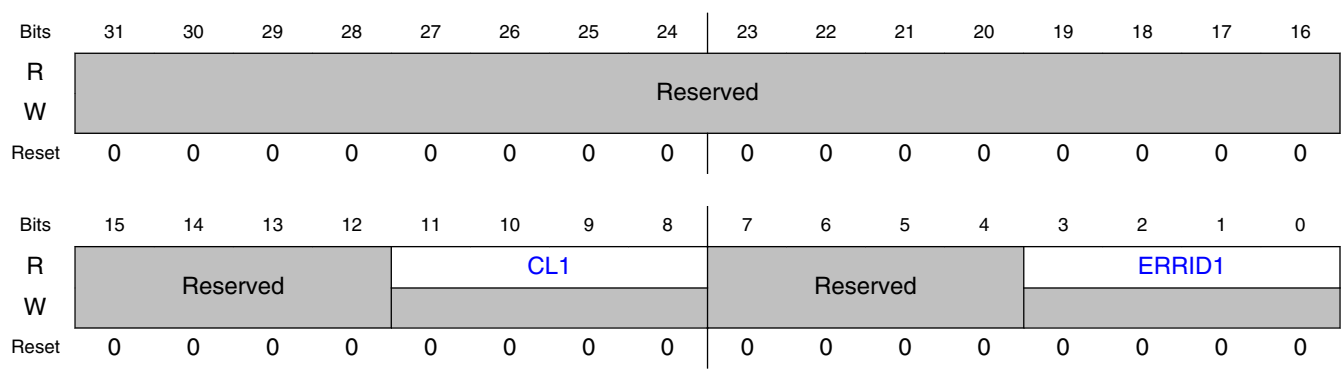
38.12.11.1 Address

Register	Offset
LTC0_ESTA	4Ch

38.12.11.2 Function

The LTC Error Register shows the status of the internal Crypto Engine and its internal registers.

38.12.11.3 Diagram



38.12.11.4 Fields

Field	Function
31-12 —	Reserved.
11-8 CL1	algorithms. The algorithms field indicates which algorithm is asserting an error. Others reserved 0000b - LTC General Error 0001b - AES 0010b - DES 0100b - MDHA 1000b - Public Key
7-4 —	Reserved
3-0 ERRID1	Error ID 1. These bits indicate the type of error that was found while processing the Descriptor. The Algorithm that is associated with the error can be found in the CL1 field. Others reserved. 0001b - Mode Error 0010b - Data Size Error, including PKHA N Register Size Error 0011b - Key Size Error, including PKHA E Register Size Error 0100b - PKHA A Register Size Error 0101b - PKHA B Register Size Error 0110b - Data Arrived out of Sequence Error 0111b - PKHA Divide by Zero Error 1000b - PKHA Modulus Even Error 1001b - DES Key Parity Error 1010b - ICV Check Failed 1011b - Internal Hardware Failure 1100b - CCM AAD Size Error (either 1. AAD flag in B0 =1 and no AAD type provided, 2. AAD flag in B0 = 0 and AAD povided, or 3. AAD flag in B0 =1 and not enough AAD provided - expecting more based on AAD size.) 1111b - Invalid Crypto Engine Selected

38.12.12 LTC AAD Size (LTC0_AADSZ)

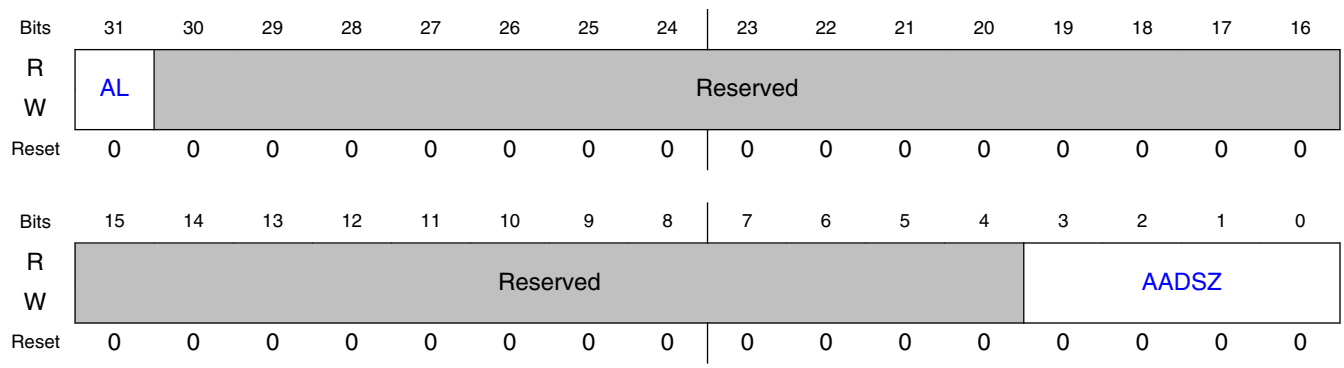
38.12.12.1 Address

Register	Offset
LTC0_AADSZ	58h

38.12.12.2 Function

The AAD Size Register is used by AESA to determine how much of the last block of AAD is valid. The write to this register should be the entire size of the AAD as it is also added directly to the Data Size Register. The size added to the Data Size Register is the AAD size rounded up to the next 16 byte boundary. For instance a size of 20 bytes written to the AAD size register will cause 32 bytes to be added to the Data Size Register. The size stored in the AADSZ field represents the number of bytes valid in the final block of AAD. However the entire size of AAD should be written to the LTC AAD Size (LTC0_AADSZ) Register address location. When authentication only is being done then the AL bit needs to be written to tell the AES engine that this is the last of the data.

38.12.12.3 Diagram



38.12.12.4 Fields

Field	Function
31 AL	AAD Last. Only AAD data will be written into the Input FIFO.
30-4 —	Reserved.
3-0 AADSZ	AAD size in Bytes, mod 16.

38.12.13 LTC IV Size (LTC0_IVSZ)

38.12.13.1 Address

Register	Offset
LTC0_IVSZ	60h

38.12.13.2 Function

The IV Size Register is used by AESA to determine how much of the last block of IV is valid. The write to this register should be the entire size of the IV as it is also added directly to the Data Size Register. The size added to the Data Size Register is the IV size rounded up to the next 16 byte boundary. For instance a size of 20 bytes written to the IV size register will cause 32 bytes to be added to the Data Size Register. The size stored in the IVSZ field represents the number of bytes valid in the final block of IV. However the entire size of IV should be written to the LTC IV Size (LTC0_IVSZ) Register address location. When IV only is being done then the IL bit needs to be written to tell the AES engine that this is the last of the data.

38.12.13.3 Diagram

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

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

38.12.13.4 Fields

Field	Function
31 IL	IV Last. Only IV data will be written into the Input FIFO.
30-4 —	Reserved.

Table continues on the next page...

Field	Function
3-0 IVSZ	IV size in Bytes, mod 16.

38.12.14 LTC DPA Mask Seed (LTC0_DPAMS)

38.12.14.1 Address

Register	Offset
LTC0_DPAMS	68h

38.12.14.2 Function

The DPA Mask Seed Register is used to reseed the mask that provides resistance against Differential Power Analysis attacks on AES, DES, and 3DES keys.

38.12.14.3 Diagram

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

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

38.12.14.4 Fields

Field	Function
31-0 DPAMS	Differential Power Analysis Mask Seed.This resistance uses a randomly changing mask that introduces "noise" into the power consumed by the AESA and DESA. This reduces the signal to noise ratio that differential power analysis attacks use to "guess" bits of the AES, DES, and 3DES key. This randomly

Field	Function
	changing mask should be seeded at POR, and will continue to provide DPA resistance thereafter. However, to provide even more DPA protection it is recommended that the DPA Mask be reseeded after every 50,000 AESA and DESA blocks have been processed. DPARRN in LTC Status (LTC0_STA) will assert after POR and after 50,000 AES, DES, and 3DES blocks have been processed to indicate that it is time to write DPAMS with a new random seed. At that time software can opt to write a new seed (preferably obtained from an RNG) into the DPA Mask Seed Register (DPAMS), or software can instead opt to provide new seed earlier or later, or not at all. DPA resistance continues even if the DPA Mask is never reseeded.

38.12.15 LTC PKHA A Size (LTC0_PKASZ)

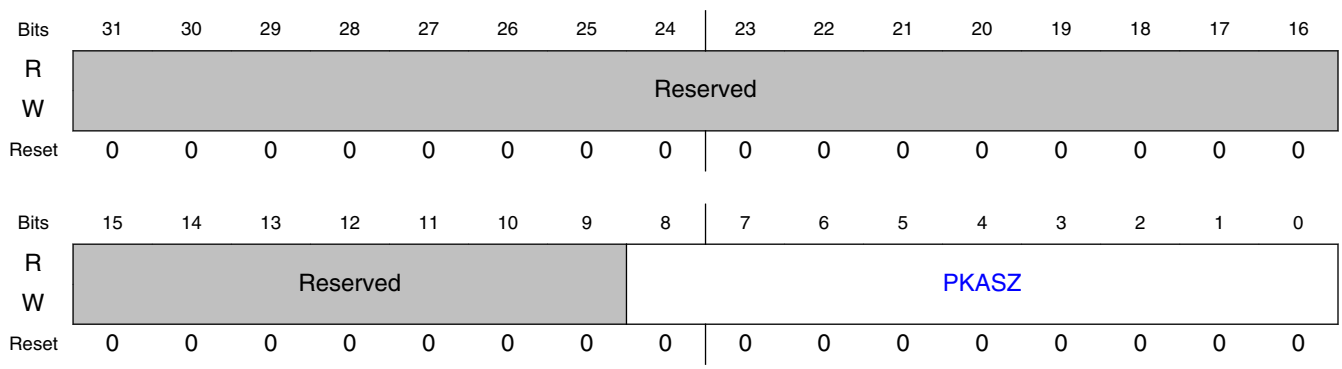
38.12.15.1 Address

Register	Offset
LTC0_PKASZ	80h

38.12.15.2 Function

The LTC PKHA A Size Register is used to indicate the number of bytes of valid data in the PKHA A Register. That is, it indicates the size, in bytes, of the numeric value contained in the PKHA A Register. Depending on the PKHA operation to be run, this register may need to be loaded prior to starting the PKHA operation. Note that some PKHA operations update this value at the completion of the operation.

38.12.15.3 Diagram



38.12.15.4 Fields

Field	Function
31-9 —	Reserved.
8-0 PKASZ	PKHA A Size. This is the size of the numeric value, in bytes, contained within the PKHA A Register.

38.12.16 LTC PKHA B Size (LTC0_PKBSZ)

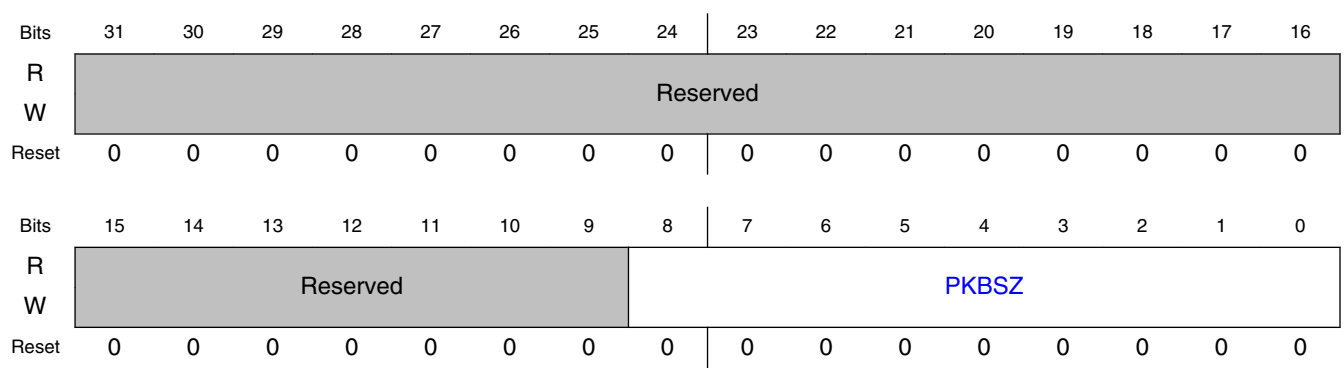
38.12.16.1 Address

Register	Offset
LTC0_PKBSZ	88h

38.12.16.2 Function

The LTC PKHA B Size Register is used to indicate the number of bytes of valid data in the PKHA B Register. That is, it indicates the size, in bytes, of the numeric value contained in the PKHA B Register. Depending on the PKHA operation to be run, this register may need to be loaded prior to starting the PKHA operation. Note that some PKHA operations update this value at the completion of the operation.

38.12.16.3 Diagram



38.12.16.4 Fields

Field	Function
31-9 —	Reserved.
8-0 PKBSZ	PKHA B Size. This is the size of the numeric value, in bytes, contained within the PKHA B Register.

38.12.17 LTC PKHA N Size (LTC0_PKNSZ)

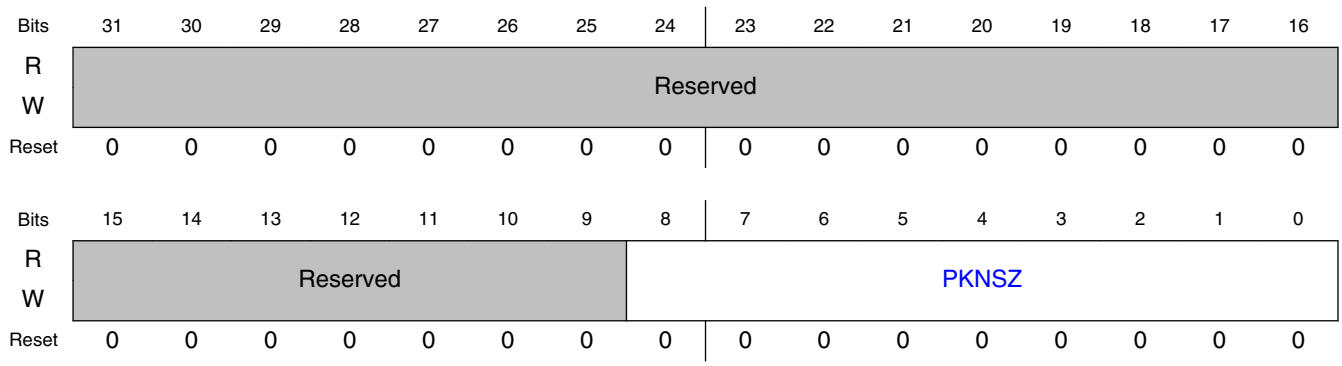
38.12.17.1 Address

Register	Offset
LTC0_PKNSZ	90h

38.12.17.2 Function

The LTC PKHA N Size Register is used to indicate the number of bytes of valid data in the PKHA N Register. That is, it indicates the size, in bytes, of the numeric value contained in the PKHA N Register. Depending on the PKHA operation to be run, this register may need to be loaded prior to starting the PKHA operation. Note that some PKHA operations update this value at the completion of the operation.

38.12.17.3 Diagram



38.12.17.4 Fields

Field	Function
31-9 —	Reserved.
8-0 PKNSZ	PKHA N Size. This is the size of the numeric value, in bytes, contained within the PKHA N Register.

38.12.18 LTC PKHA E Size (LTC0_PKESZ)

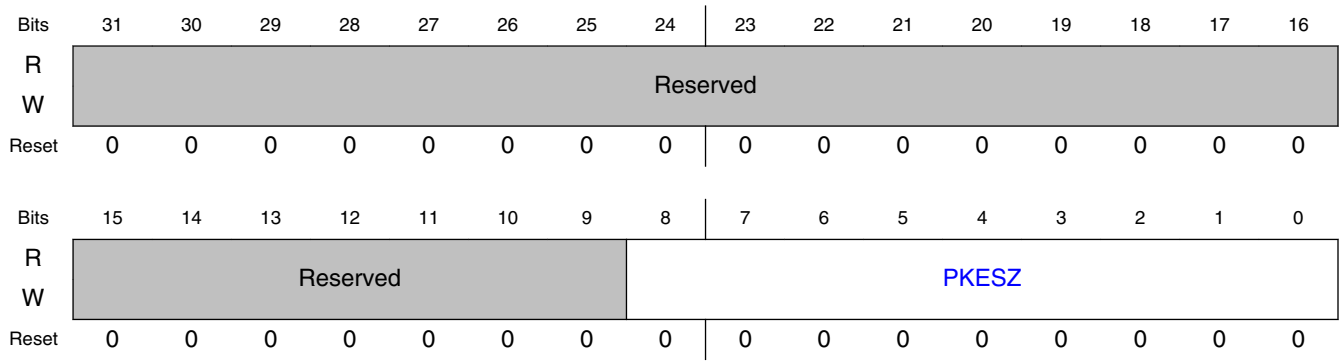
38.12.18.1 Address

Register	Offset
LTC0_PKESZ	98h

38.12.18.2 Function

The LTC PKHA E Size Register is used to indicate the number of bytes of valid data in the PKHA E Register. That is, it indicates the size, in bytes, of the numeric value contained in the PKHA E Register. Depending on the PKHA operation to be run, this register may need to be loaded prior to starting the PKHA operation. Note that some PKHA operations update this value at the completion of the operation.

38.12.18.3 Diagram



38.12.18.4 Fields

Field	Function
31-9 —	Reserved.
8-0 PKESZ	PKHA E Size. This is the size of the numeric value, in bytes, contained within the PKHA E Register.

38.12.19 LTC Context (LTC0_CTX_a)

38.12.19.1 Address

For a = 0 to 15:

Register	Offset
LTC0_CTX_a	100h + (a × 4h)

38.12.19.2 Function

The Context Register holds the context for the internal crypto engine. This register is 512 bits in length. The IP bus write to the Context Register is accessible only as full-word reads or writes to sixteen 32-bit registers. The MSB is located at offset 0100h with respect to the register page.

The bit assignments of this register are dependent on the algorithm, and in some cases the mode of that algorithm. See the appropriate section for the Context Register format used for that algorithm:

- AES ECB: Section AES ECB mode use of the Context Register
- AES CBC: Section AES CBC mode use of the Context Register
- AES CTR: Section AES CTR mode use of the Context Register
- AES CCM: Section AES CCM and CCM* mode use of the Context Register
- DES: Section DESA Context Register
- Triple DES: Section DESA Context Register

38.12.19.3 Diagram

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

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

38.12.19.4 Fields

Field	Function
31-0 CTX	CTX

38.12.20 LTC Keys (LTC0_KEY_a)

38.12.20.1 Address

Register	Offset
LTC0_KEY_0	200h
LTC0_KEY_1	204h
LTC0_KEY_2	208h
LTC0_KEY_3	20Ch
LTC0_KEY_4	210h
LTC0_KEY_5	214h
LTC0_KEY_6	218h
LTC0_KEY_7	21Ch

38.12.20.2 Function

The Key Register normally holds the left-aligned key for the internal crypto engine. The MSB is in offset 200h. The Key Register is 256 bits in length. The IP bus write to the Context Register is accessible only as full-word reads or writes to four 32-bit registers.

Before the value in the Key Register can be used in a cryptographic operation, the size of the key must be written into the Key Size Register. Once the Key Size Register has been written, the Key Register cannot be written again until the Key Size Register has been cleared.

38.12.20.3 Diagram

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

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

38.12.20.4 Fields

Field	Function
31-0 KEY	KEY

38.12.21 LTC Version ID (LTC0_VID1)

38.12.21.1 Address

Register	Offset
LTC0_VID1	4F0h

38.12.21.2 Function

This register contains the ID for LTC and major and minor revision numbers.

38.12.21.3 Diagram

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

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

38.12.21.4 Fields

Field	Function
31-16	ID(0x0038).

Table continues on the next page...

Field	Function
IP_ID	
15-8 MAJ_REV	Major revision number.
7-0 MIN_REV	Minor revision number.

38.12.22 LTC Version ID 2 (LTC0_VID2)

38.12.22.1 Address

Register	Offset
LTC0_VID2	4F4h

38.12.22.2 Function

This register contains the architectural era and eco revision numbers.

38.12.22.3 Diagram

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

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

38.12.22.4 Fields

Field	Function
31-16 —	Reserved
15-8 ARCH_ERA	Architectural ERA.
7-0 ECO_REV	ECO revision number.

38.12.23 LTC CHA Version ID (LTC0_CHAVID)

38.12.23.1 Address

Register	Offset
LTC0_CHAVID	4F8h

38.12.23.2 Function

This register contains the Version ID and Revision Number for the CHAs contained within LTC.

38.12.23.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MDHAVID				MDHAREV				PKHAVID				PKHAREV			
W																
Reset	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	1

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

38.12.23.4 Fields

Field	Function
31-28 MDHAVID	MDHA Hashing Version ID. 0000 - Low-power MDHA, with SHA-1, SHA-256, SHA-224, MD5, SMAC 0001 - Low-power MDHA, with SHA-1, SHA-256, SHA-224, SHA-512, SHA-384, SHA-512/224, SHA-512/256, MD5, HMAC 0010 - Double Speed MDHA, with SHA-1, SHA-256, SHA-224, SHA-512, SHA-384, SHA-512/224, SHA-512/256, MD5, HMAC, SMAC 0011 - Quad Speed MDHA, with SHA-1, SHA-256, SHA-224, SHA-512, SHA-384, SHA-512/224, SHA-512/256, MD5, HMAC, SMAC 0100 - Ultra Low Performance MDHA, with SHA-1, SHA-256, SHA-224 0101 - Ultra Low Performance MDHA, with SHA-1, SHA-256, SHA-224, SHA-512, SHA-384, SHA-512/224, SHA-512/256
27-24 MDHAREV	MDHA Revision Number
23-20 PKHAVID	PK Version ID 0001 - 32-bit PKHA-SD 0010 - 64-bit PKHA-SD 0011 - 128-bit PKHA-SD 0100 - 16-bit PKHA-SD
19-16 PKHAREV	PK Revision Number
15-12 DESVID	DES Version ID(0x0). 0000 - High-performance DESA 0001 - Low-performance DESA
11-8 DESREV	DES Revision Number
7-4 AESVID	AES Version ID
3-0 AESREV	AES Revision Number

38.12.24 LTC FIFO Status (LTC0_FIFOSTA)

38.12.24.1 Address

Register	Offset
LTC0_FIFOSTA	7C0h

38.12.24.2 Function

The LTC FIFO Status shows the current levels of the Input and Output FIFO.

38.12.24.3 Diagram

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

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

38.12.24.4 Fields

Field	Function
31 OFF	Output FIFO Full. The Output FIFO is full and should not be written to.
30-23 —	Reserved
22-16 OFL	Output FIFO Level. These bits indicate the current number of entries in the Output FIFO.
15 IFF	Input FIFO Full. The Input FIFO is full and should not be written to.
14-7 —	Reserved
6-0 IFL	Input FIFO Level. These bits indicate the current number of entries in the Input FIFO.

38.12.25 LTC Input Data FIFO (LTC0_IFIFO)

38.12.25.1 Address

Register	Offset
LTC0_IFIFO	7E0h

38.12.25.2 Function

Data to be processed by the various crypto engines is first pushed into the Input Data FIFO. The Input Data FIFO supports byte enables, allowing one to four bytes to be written to the IFIFO from the IP bus. The IFIFO is four entries deep, and each entry is four bytes. Care must be used to not overflow the Input Data FIFO. Reads from this address will always return 0x0.

38.12.25.3 Diagram

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

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

38.12.25.4 Fields

Field	Function
31-0 IFIFO	IFIFO

38.12.26 LTC Output Data FIFO (LTC0_OFIFO)

38.12.26.1 Address

Register	Offset
LTC0_OFIFO	7F0h

38.12.26.2 Function

Data that is output from the AES is pushed into the Output Data FIFO. The OFIFO is four entries deep, and each entry is four bytes. During normal operation, the AES will never overflow the Output Data FIFO. Writes to this register are ignored.

38.12.26.3 Diagram

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

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

38.12.26.4 Fields

Field	Function
31-0 OFIFO	Output FIFO

38.12.27 LTC PKHA A0 (LTC0_PKA0_a)

38.12.27.1 Address

For a = 0 to 15:

Register	Offset
LTC0_PKA0_a	800h + (a × 4h)

38.12.27.2 Function

The LTC PKHA A0 Register contains a numeric value of up to 64 bytes, right-justified. The LSByte is in offset 800h, in the right-most byte. The LTC PKHA A0 register is 512 bits in length. Note that PKHA A3, A2, A1 and A0 Registers may be combined to contain a value of up to 2048 bits, with A0 containing the least-significant 64 bytes of this value. The value in the PKHA A0 Register (or {A3, A2, A1, A0}, if used as a single value) is limited by the LTC PKHA A Size (LTC0_PKASZ).

Note that the PKHA Registers can be read/written only when the LTC Mode (PublicKey) (LTC0_MDPK) does not contain a value enabling a PKHA operation. If the LTCMR_PK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

38.12.27.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PKHA_A0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PKHA_A0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.12.27.4 Fields

Field	Function
31-0 PKHA_A0	A0 VALUE

38.12.28 LTC PKHA A (LTC0_PKA_a)

38.12.28.1 Address

For a = 0 to 63:

Register	Offset
LTC0_PKA_a	800h + (a × 4h)

38.12.28.2 Function

The LTC PKHA A Register contains a numeric value of up to 256 bytes, right-justified. The LSByte is in offset 800h, in the right-most byte. The LTC PKHA A register is 2048 bits in length. Note that PKHA A Register is a single register that combines A0, A1, A2 and A3. The value in the PKHA A Register is limited by the LTC PKHA A Size (LTC0_PKASZ).

Note that the PKHA Registers can be read/written only when the LTC Mode (PublicKey) (LTC0_MDPK) does not contain a value enabling a PKHA operation. If the LTCMR_PK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

38.12.28.3 Diagram

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

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

38.12.28.4 Fields

Field	Function
31-0 PKHA_A	A VALUE

38.12.29 LTC PKHA A1 (LTC0_PKA1_a)

38.12.29.1 Address

For a = 0 to 15:

Register	Offset
LTC0_PKA1_a	840h + (a × 4h)

38.12.29.2 Function

The LTC PKHA A1 Register contains a numeric value of up to 64 bytes, right-justified. The LSByte is in offset 840h, in the right-most byte. The LTC PKHA A1 register is 512 bits in length. Note that PKHA A3, A2, A1 and A0 Registers may be combined to contain a value of up to 2048 bits, with A0 containing the least-significant 64 bytes of this value. The value in the PKHA A1 Register, if used as a separate value, is limited by the LTC PKHA A Size (LTC0_PKASZ).

Note that the PKHA Registers can be read/written only when the LTC Mode (PublicKey) (LTC0_MDPK) does not contain a value enabling a PKHA operation. If the LTCMR_PK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

38.12.29.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PKHA_A1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PKHA_A1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.12.29.4 Fields

Field	Function
31-0 PKHA_A1	A1 VALUE

38.12.30 LTC PKHA A2 (LTC0_PKA2_a)

38.12.30.1 Address

For a = 0 to 15:

Register	Offset
LTC0_PKA2_a	880h + (a × 4h)

38.12.30.2 Function

The LTC PKHA A2 Register contains a numeric value of up to 64 bytes, right-justified. The LSByte is in offset 880h, in the right-most byte. The LTC PKHA A2 register is 512 bits in length. Note that PKHA A3, A2, A1 and A0 Registers may be combined to contain a value of up to 2048 bits, with A0 containing the least-significant 64 bytes of this value. The value in the PKHA A2 Register, if used as a separate value, is limited by the LTC PKHA A Size (LTC0_PKASZ).

Note that the PKHA Registers can be read/written only when the LTC Mode (PublicKey) (LTC0_MDPK) does not contain a value enabling a PKHA operation. If the LTCMR_PK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

38.12.30.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PKHA_A2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PKHA_A2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.12.30.4 Fields

Field	Function
31-0 PKHA_A2	A2 VALUE

38.12.31 LTC PKHA A3 (LTC0_PKA3_a)

38.12.31.1 Address

For a = 0 to 15:

Register	Offset
LTC0_PKA3_a	8C0h + (a × 4h)

38.12.31.2 Function

The LTC PKHA A3 Register contains a numeric value of up to 64 bytes, right-justified. The LSByte is in offset 8C0h, in the right-most byte. The LTC PKHA A3 register is 512 bits in length. Note that PKHA A3, A2, A1 and A0 Registers may be combined to contain a value of up to 2048 bits, with A0 containing the least-significant 64 bytes of this value. The value in the PKHA A3 Register, if used as a separate value, is limited by the LTC PKHA A Size (LTC0_PKASZ).

Note that the PKHA Registers can be read/written only when the LTC Mode (PublicKey) (LTC0_MDPK) does not contain a value enabling a PKHA operation. If the LTCMR_PK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

38.12.31.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PKHA_A3															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PKHA_A3															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.12.31.4 Fields

Field	Function
31-0	A3 VALUE

Field	Function
PKHA_A3	

38.12.32 LTC PKHA B0 (LTC0_PKB0_a)

38.12.32.1 Address

For a = 0 to 15:

Register	Offset
LTC0_PKB0_a	A00h + (a × 4h)

38.12.32.2 Function

The LTC PKHA B0 Register contains a numeric value of up to 64 bytes, right-justified. The LSByte is in offset A00h, in the right-most byte. The LTC PKHA B0 register is 512 bits in length. Note that PKHA B3, B2, B1 and B0 Registers may be combined to contain a value of up to 2048 bits, with B0 containing the least-significant 64 bytes of this value. The value in the PKHA B0 Register (or in {B3, B2, B1, B0} if used as a single value) is limited by the LTC PKHA B Size (LTC0_PKBSZ).

Note that the PKHA Registers can be read/written only when the LTC Mode (PublicKey) (LTC0_MDPK) does not contain a value enabling a PKHA operation. If the LTC Mode (PublicKey) (LTC0_MDPK) contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

38.12.32.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PKHA_B0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PKHA_B0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.12.32.4 Fields

Field	Function
31-0 PKHA_B0	B0 VALUE

38.12.33 LTC PKHA B (LTC0_PKB_a)

38.12.33.1 Address

For a = 0 to 63:

Register	Offset
LTC0_PKB_a	A00h + (a × 4h)

38.12.33.2 Function

The LTC PKHA B Register contains a numeric value of up to 256 bytes, right-justified. The LSByte is in offset A00h, in the right-most byte. The LTC PKHA B register is 2048 bits in length. Note that PKHA B Register is a single register that combines B0, B1, B2 and B3. The value in the PKHA B Register is limited by the LTC PKHA B Size (LTC0_PKBSZ).

Note that the PKHA Registers can be read/written only when the LTC Mode (PublicKey) (LTC0_MDPK) does not contain a value enabling a PKHA operation. If the LTCMR_PK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

38.12.33.3 Diagram

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

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

38.12.33.4 Fields

Field	Function
31-0 PKHA_B	B VALUE

38.12.34 LTC PKHA B1 (LTC0_PKB1_a)

38.12.34.1 Address

For a = 0 to 15:

Register	Offset
LTC0_PKB1_a	A40h + (a × 4h)

38.12.34.2 Function

The LTC PKHA B1 Register contains a numeric value of up to 64 bytes, right-justified. The LSByte is in offset A40h, in the right-most byte. The LTC PKHA B1 register is 512 bits in length. Note that PKHA B3, B2, B1 and B0 Registers may be combined to contain a value of up to 2048 bits, with B0 containing the least-significant 64 bytes of this value. The value in the PKHA B1 Register, if used as a separate value, is limited by the LTC PKHA B Size (LTC0_PKBSZ).

Note that the PKHA Registers can be read/written only when the LTC Mode (PublicKey) (LTC0_MDPK) does not contain a value enabling a PKHA operation. If the LTCMR_PK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

38.12.34.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PKHA_B1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PKHA_B1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.12.34.4 Fields

Field	Function
31-0 PKHA_B1	B1 VALUE

38.12.35 LTC PKHA B2 (LTC0_PKB2_a)

38.12.35.1 Address

For a = 0 to 15:

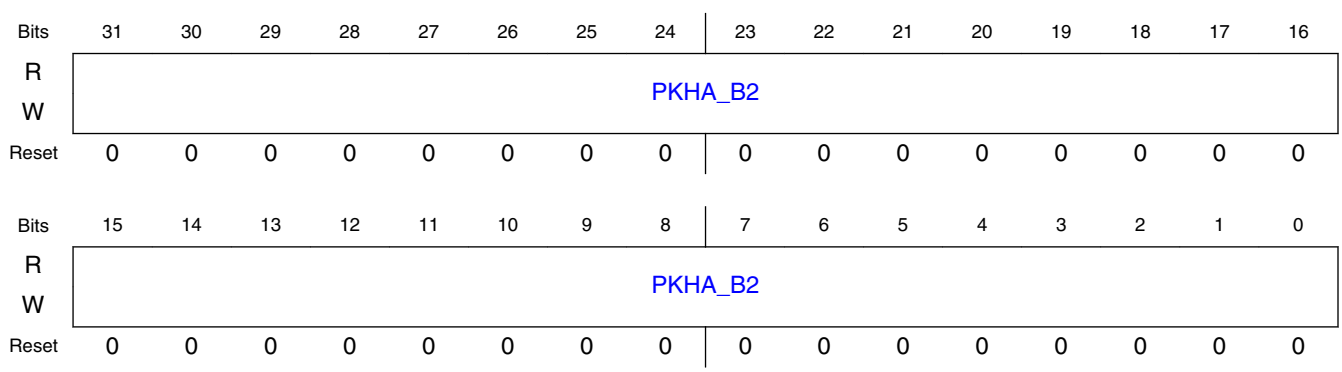
Register	Offset
LTC0_PKB2_a	A80h + (a × 4h)

38.12.35.2 Function

The LTC PKHA B2 Register contains a numeric value of up to 64 bytes, right-justified. The LSByte is in offset A80h, in the right-most byte. The LTC PKHA B2 register is 512 bits in length. Note that PKHA B3, B2, B1 and B0 Registers may be combined to contain a value of up to 2048 bits, with B0 containing the least-significant 64 bytes of this value. The value in the PKHA B2 Register, if used as a separate value, is limited by the LTC PKHA B Size (LTC0_PKBSZ).

Note that the PKHA Registers can be read/written only when the LTC Mode (PublicKey) (LTC0_MDPK) does not contain a value enabling a PKHA operation. If the LTCMR_PK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

38.12.35.3 Diagram



38.12.35.4 Fields

Field	Function
31-0	B2 VALUE

Field	Function
PKHA_B2	

38.12.36 LTC PKHA B3 (LTC0_PKB3_a)

38.12.36.1 Address

For a = 0 to 15:

Register	Offset
LTC0_PKB3_a	AC0h + (a × 4h)

38.12.36.2 Function

The LTC PKHA B3 Register contains a numeric value of up to 64 bytes, right-justified. The LSByte is in offset AC0h, in the right-most byte. The LTC PKHA B3 register is 512 bits in length. Note that PKHA B3, B2, B1 and B0 Registers may be combined to contain a value of up to 2048 bits, with B0 containing the least-significant 64 bytes of this value. The value in the PKHA B3 Register, if used as a separate value, is limited by the LTC PKHA B Size (LTC0_PKBSZ).

Note that the PKHA Registers can be read/written only when the LTC Mode (PublicKey) (LTC0_MDPK) does not contain a value enabling a PKHA operation. If the LTCMR_PK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

38.12.36.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PKHA_B3															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PKHA_B3															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.12.36.4 Fields

Field	Function
31-0 PKHA_B3	B3 VALUE

38.12.37 LTC PKHA N0 (LTC0_PKN0_a)

38.12.37.1 Address

For a = 0 to 15:

Register	Offset
LTC0_PKN0_a	C00h + (a × 4h)

38.12.37.2 Function

The LTC PKHA N0 Register contains a numeric value of up to 64 bytes, right-justified. The LSByte is in offset C00h, in the right-most byte. The LTC PKHA N0 register is 512 bits in length. Note that PKHA N3, N2, N1 and N0 Registers may be combined to contain a value of up to 2048 bits, with N0 containing the least-significant 64 bytes of this value. The value in the PKHA N0 Register (or {N3, N2, N1, N0}, if used as a single value) is limited by the LTC PKHA N Size (LTC0_PKNSZ).

Note that the PKHA Registers can be read/written only when the LTC Mode (PublicKey) (LTC0_MDPK) does not contain a value enabling a PKHA operation. If the LTCMR_PK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

38.12.37.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PKHA_N0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PKHA_N0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.12.37.4 Fields

Field	Function
31-0 PKHA_N0	N0 VALUE

38.12.38 LTC PKHA N (LTC0_PKN_a)

38.12.38.1 Address

For a = 0 to 63:

Register	Offset
LTC0_PKN_a	C00h + (a × 4h)

38.12.38.2 Function

The LTC PKHA N Register contains a numeric value of up to 256 bytes, right-justified. The LSByte is in offset C00h, in the right-most byte. The LTC PKHA N register is 2048 bits in length. Note that PKHA N Register is a single register that combines N0, N1, N2 and N3. The value in the PKHA N Register is limited by the LTC PKHA N Size (LTC0_PKNSZ).

Note that the PKHA Registers can be read/written only when the LTC Mode (PublicKey) (LTC0_MDPK) does not contain a value enabling a PKHA operation. If the LTCMR_PK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

38.12.38.3 Diagram

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

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

38.12.38.4 Fields

Field	Function
31-0 PKHA_N	N VALUE

38.12.39 LTC PKHA N1 (LTC0_PKN1_a)

38.12.39.1 Address

For a = 0 to 15:

Register	Offset
LTC0_PKN1_a	C40h + (a × 4h)

38.12.39.2 Function

The LTC PKHA N1 Register contains a numeric value of up to 64 bytes, right-justified. The LSByte is in offset C40h, in the right-most byte. The LTC PKHA N1 register is 512 bits in length. Note that PKHA N3, N2, N1 and N0 Registers may be combined to contain a value of up to 2048 bits, with N0 containing the least-significant 64 bytes of this value. The value in the PKHA N1 Register, if used as a separate value, is limited by the LTC PKHA N Size (LTC0_PKNSZ).

Note that the PKHA Registers can be read/written only when the LTC Mode (PublicKey) (LTC0_MDPK) does not contain a value enabling a PKHA operation. If the LTCMR_PK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

38.12.39.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PKHA_N1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PKHA_N1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.12.39.4 Fields

Field	Function
31-0 PKHA_N1	N1 VALUE

38.12.40 LTC PKHA N2 (LTC0_PKN2_a)

38.12.40.1 Address

For a = 0 to 15:

Register	Offset
LTC0_PKN2_a	C80h + (a × 4h)

38.12.40.2 Function

The LTC PKHA N2 Register contains a numeric value of up to 64 bytes, right-justified. The LSByte is in offset C80h, in the right-most byte. The LTC PKHA N2 register is 512 bits in length. Note that PKHA N3, N2, N1 and N0 Registers may be combined to contain a value of up to 2048 bits, with N0 containing the least-significant 64 bytes of this value. The value in the PKHA N2 Register, if used as a separate value, is limited by the LTC PKHA N Size (LTC0_PKNSZ).

Note that the PKHA Registers can be read/written only when the LTC Mode (PublicKey) (LTC0_MDPK) does not contain a value enabling a PKHA operation. If the LTCMR_PK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

38.12.40.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PKHA_N2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PKHA_N2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.12.40.4 Fields

Field	Function
31-0	N2 VALUE

Field	Function
PKHA_N2	

38.12.41 LTC PKHA N3 (LTC0_PKN3_a)

38.12.41.1 Address

For a = 0 to 15:

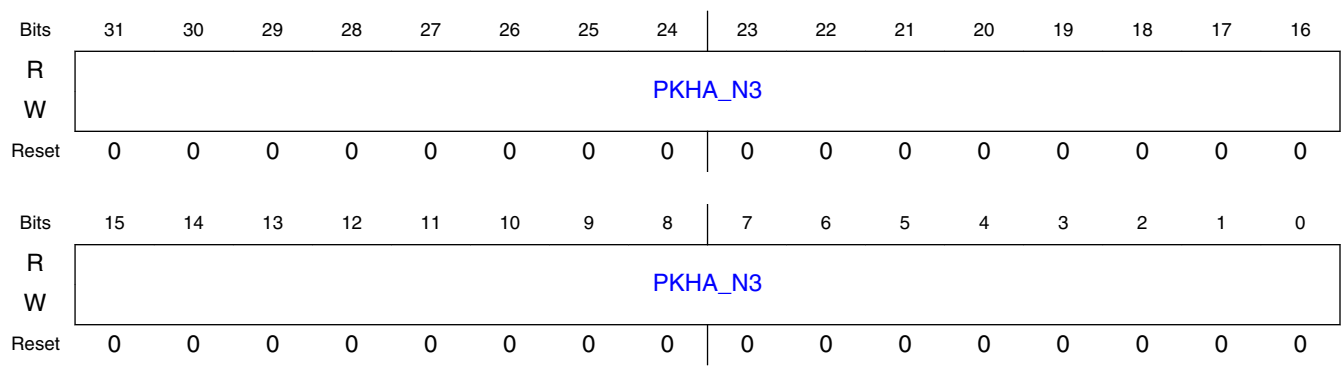
Register	Offset
LTC0_PKN3_a	CC0h + (a × 4h)

38.12.41.2 Function

The LTC PKHA N3 Register contains a numeric value of up to 64 bytes, right-justified. The LSByte is in offset CC0h, in the right-most byte. The LTC PKHA N3 register is 512 bits in length. Note that PKHA N3, N2, N1 and N0 Registers may be combined to contain a value of up to 2048 bits, with N0 containing the least-significant 64 bytes of this value. The value in the PKHA N3 Register, if used as a separate value, is limited by the LTC PKHA N Size (LTC0_PKNSZ).

Note that the PKHA Registers can be read/written only when the LTC Mode (PublicKey) (LTC0_MDPK) does not contain a value enabling a PKHA operation. If the LTCMR_PK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

38.12.41.3 Diagram



38.12.41.4 Fields

Field	Function
31-0 PKHA_N3	N3 VALUE

38.12.42 LTC PKHA E (LTC0_PKE_a)

38.12.42.1 Address

For a = 0 to 63:

Register	Offset
LTC0_PKE_a	E00h + (a × 4h)

38.12.42.2 Function

The LTC PKHA E Register contains a numeric value of up to 256 bytes, right-justified. The LSByte is in offset E00h, in the right-most byte. The LTC PKHA E register is 2048 bits in length. The value in the PKHA E Register is limited by the LTC PKHA E Size (LTC0_PKESZ).

Note that the PKHA E Register can be written only when the LTC Mode (PublicKey) (LTC0_MDPK) does not contain a value enabling a PKHA operation. If the LTCMR_PK contains a PKHA operation mode value, then writes to the PKHA E Register will be ignored.

38.12.42.3 Diagram

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

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

38.12.42.4 Fields

Field	Function
31-0 PKHA_E	E VALUE

Chapter 39
Analog-to-Digital Converter (ADC)

39.1 Chip-specific ADC information

39.1.1 DMA support on ADC

Applications may require continuous sampling of the ADC (4K samples/sec) that may have considerable load on the CPU. The ADC can trigger the DMA (via DMA req) on conversion completion.

39.1.2 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.

NOTE

Some ADC inputs may not appear on all packages. Please refer to KL81 signal multiplexing and pin assignments for more information.

ADC Channel (SC1n[ADCH])	Channel	Input signal(SC1n[DIFF]= 1)	Input signal(SC1n[DIFF]= 0)
00000	DAD0	ADC0_DP0 and ADC0_DM0	ADC0_DP0
00001	DAD1	ADC0_DP1 and ADC0_DM1	ADC0_DP1
00010	DAD2	Reserved	Reserved
00011	DAD3	Reserved	Reserved
00100	AD4a	Reserved	Reserved

Table continues on the next page...

ADC Channel (SC1n[ADCH])	Channel	Input signal(SC1n[DIFF]= 1)	Input signal(SC1n[DIFF]= 0)
00101	AD5a	Reserved	Reserved
00110	AD6a	Reserved	Reserved
00111	AD7a	Reserved	Reserved
00100	AD4b	Reserved	ADC0_SE4b
00101	AD5b	Reserved	ADC0_SE5b
00110	AD6b	Reserved	ADC_SE6b
00111	AD7b	Reserved	ADC0_SE7b
01000	AD8	Reserved	ADC0_SE8
01001	AD9	Reserved	ADC0_SE9
01010	AD10	Reserved	Reserved
01011	AD11	Reserved	Reserved
01100	AD12	Reserved	ADC0_SE12
01101	AD13	Reserved	ADC0_SE13
01110	AD14	Reserved	ADC0_SE14
01111	AD15	Reserved	ADC0_SE15
10000	AD16	Reserved	TAMPER1
10001	AD17	Reserved	ADC0_DM0
10010	AD18	Reserved	ADC0_DM1
10011	AD19	Reserved	Reserved
10100	AD20	Reserved	Reserved
10101	AD21	Reserved	VBAT
10110	AD22	Reserved	VREF_OUT/ADC0_SE22
10111	AD23	Reserved	12-bit DAC0 Output/ ADC0_SE23
11000	AD24	Sense Bus (for test)	Reserved
11001	AD25	Reserved	Reserved
11010	AD26	Temprature Sensor (Diff)	Temprature Sensor (S.E)
11011	AD27	Badgap (Diff)	Bandgap (Diff)
11100	AD28	Reserved	Reserved
11101	AD29	VREFH (Diff)	VREFH (S.E)
11110	AD30	Reserved	VREFL
11111	AD31	Module Disabled	Module Disabled

39.1.3 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.

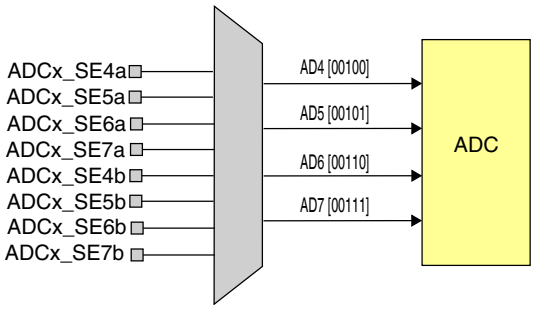


Figure 39-1. ADCx_SEn channels a and b selection

39.1.4 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.

39.1.5 VBAT connection to ADC input channel

The VBAT supply input can be converted as a single ended input to an ADC channel input. See ADC0 Connections/Channel Assignment for more details. When VBAT is greater than the selected voltage reference, the conversion result will show a saturated result (~0xFFFF in 16-bit operation). When measuring the VBAT voltage level the ADC should be configured for a long sample time (ADCx_CFG1[ADLSMP]=1, ADCx_CFG2[ADLSTS]=00).

39.1.6 ADC triggers

The ADC supports both software and hardware triggers. The ADC can conduct conversions in low power modes. 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.

Table 39-1. ADC Alternate trigger options

SIM_SOPT7[ADCxTRGSEL]	Selected Source
00000	External trigger pin input (EXTRG_IN)

Table continues on the next page...

Table 39-1. ADC Alternate trigger options (continued)

SIM_SOPT7[ADCxTRGSEL]	Selected Source
00001	CMP0 output
00010	Reserved
00011	Reserved
00100	PIT trigger 0
00101	PIT trigger 1
00110	PIT trigger 2
00111	PIT trigger 3
01000	TPM0 overflow
01001	TPM1 overflow
01010	TPM2 overflow
01011	LPTMR1 trigger
01100	RTC alarm
01101	RTC Seconds
01110	LPTMR0 trigger
01111	TPM1 channel 0 (A pretrigger) and channel 1 (B pretrigger)

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

39.1.7 ADC conversion clock options

The ADC has multiple input clock sources. Selection is determined by ADCx_CFG1[ADICLK] bitfield. The following table shows the chip-specific clock assignments for this bitfield.

NOTE

The ALTCLK 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.

Table 39-2. ADC Conversion Clock Options

ADCx_CFG1[ADICLK]	ADC defined selection	Chip clock	Note
00	Bus Clock	Bus Clock	
01	ALTCLK2	IRC48MCLK	Note [1]
10	ALTCLK	OSCERCLK	Note [1]
11	Asynchronous clock (ADACK)	N/A - sourced from within ADC block	Note[1]

1. For ADC operation in Compute only, PSTOP1, Stop and VLPS, ADACK and the alternate clock sources are allowed clock sources. Note however that ALTCLK2 is force disabled and therefore not available in VLPS.

39.1.8 Satellite ADC Muxes

ADC hard block only has 7 user analog inputs and one test analog input (selected when ADCH=0x24). The first 4 analog inputs are intended for high precision and differential channel inputs with direct connection between the associated pad and the ADC. The other three ADC inputs are intended to be used for connection to 8-to-1 muxes (instantiated hierarchically external to ADC) and support a chip-specific number of ADC channel input.

Analog inputs that connect to the ADC via these satellite muxes will have some minimal degradation. Control for these muxes are driven from the ADC digital (the lower three bits of ADCH[4:0] as well as the ADCx_CFG2[MUXSEL]).

39.1.9 ADC low-power modes

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

Table 39-3. ADC low-power modes

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

39.2 Introduction

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.

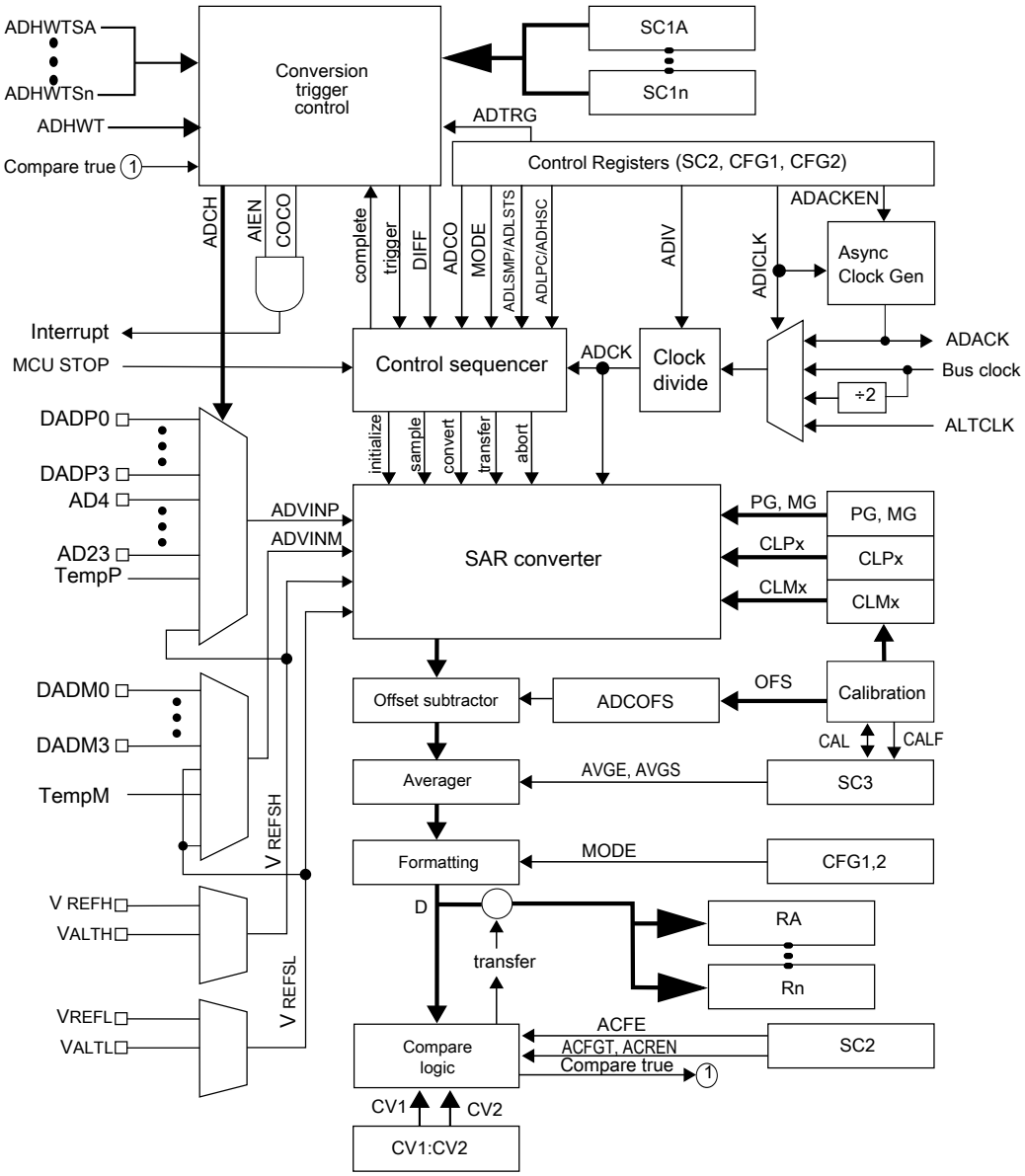
39.2.1 Features

Following are the features of the ADC module.

- 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

39.2.2 Block diagram

The following figure is the ADC module block diagram.


ADHWTSA
ADHWTSn
ADHWT
Compare true
Conversion trigger control
SC1A
SC1n
ADTRG
Control Registers (SC2, CFG1, CFG2)
ADACKEN
Async Clock Gen
ADCH
AIEN
COCO
complete
trigger
DIFF
ADCO
MODE
ADLSMP/ADLSTS
ADLPC/ADHSC
ADIV
ADICLK
Interrupt
MCU STOP
Control sequencer
ADCK
Clock divide
ADACK
Bus clock
÷2
ALTCLK
initialize
sample
convert
transfer
abort
DADP0
DADP3
AD4
AD23
TempP
ADVINP
ADVINM
SAR converter
PG, MG
CLPx
CLMx
DADM0
DADM3
TempM
Offset subtractor
ADCOFS
OFS
Calibration
CAL
CALF
Averager
AVGE, AVGS
SC3
Formatting
MODE
CFG1,2
V REFSH
V REFH
VALTH
V REFSL
VREFL
VALTL
D
transfer
RA
Rn
Compare logic
ACFE
ACFGT, ACREN
Compare true
SC2
CV1
CV2
CV1:CV2

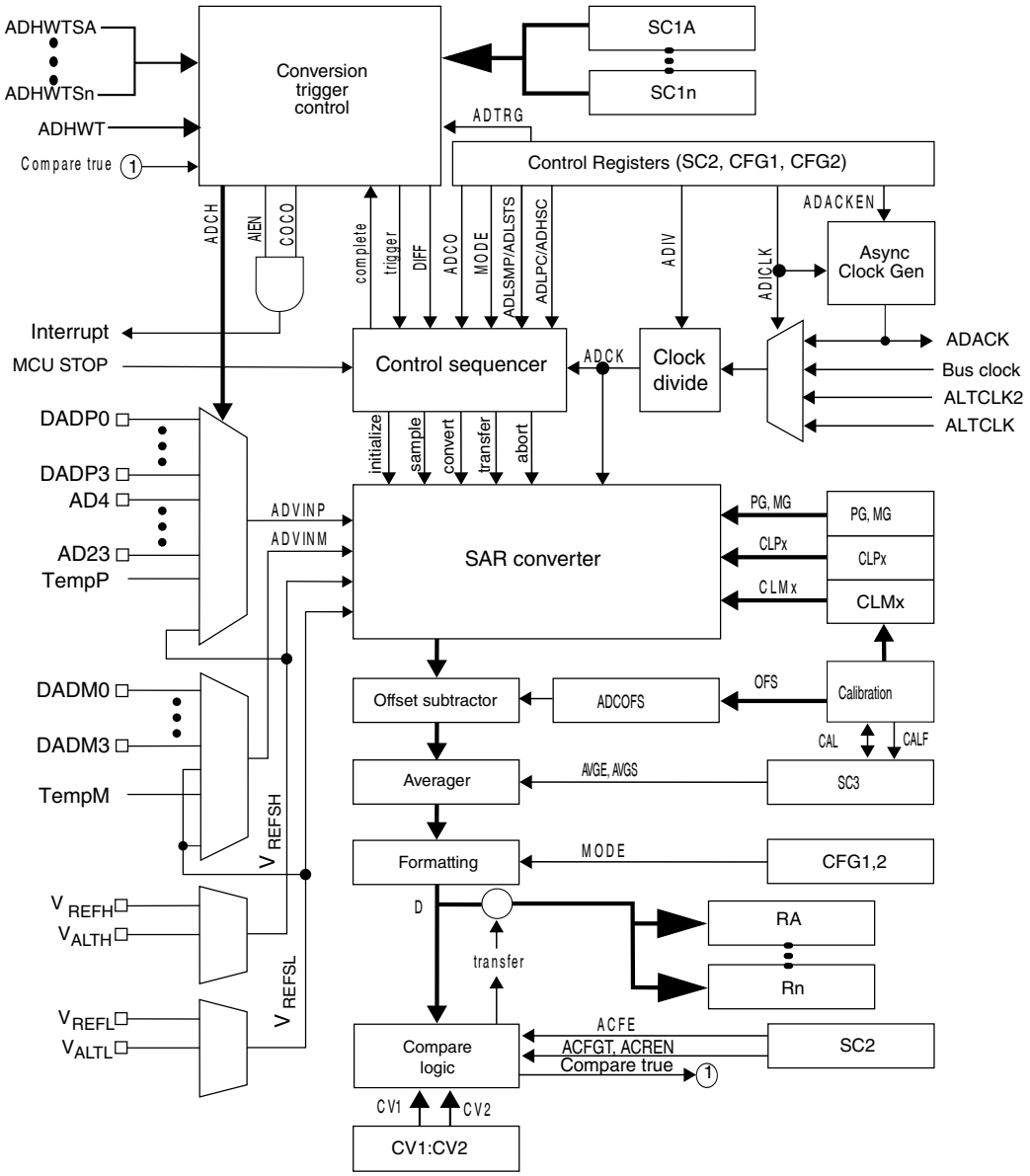


Figure 39-2. ADC block diagram

39.3 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

For the number of channels supported on this device as well as information regarding other chip-specific inputs into the ADC block, see the chip-specific ADC configuration information.

Table 39-4. 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

39.3.1 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.

39.3.2 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}.

39.3.3 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}.

39.3.4 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.

39.3.5 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. For ADC connections specific to this device, see the chip-specific ADC information.

39.4 Memory map and register definitions

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	39.4.1/1125
4003_B004	ADC Status and Control Registers 1 (ADC0_SC1B)	32	R/W	0000_001Fh	39.4.1/1125
4003_B008	ADC Configuration Register 1 (ADC0_CFG1)	32	R/W	0000_0000h	39.4.2/1129
4003_B00C	ADC Configuration Register 2 (ADC0_CFG2)	32	R/W	0000_0000h	39.4.3/1130
4003_B010	ADC Data Result Register (ADC0_RA)	32	R	0000_0000h	39.4.4/1131

Table continues on the next page...

ADC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_B014	ADC Data Result Register (ADC0_RB)	32	R	0000_0000h	39.4.4/1131
4003_B018	Compare Value Registers (ADC0_CV1)	32	R/W	0000_0000h	39.4.5/1133
4003_B01C	Compare Value Registers (ADC0_CV2)	32	R/W	0000_0000h	39.4.5/1133
4003_B020	Status and Control Register 2 (ADC0_SC2)	32	R/W	0000_0000h	39.4.6/1134
4003_B024	Status and Control Register 3 (ADC0_SC3)	32	R/W	0000_0000h	39.4.7/1136
4003_B028	ADC Offset Correction Register (ADC0_OFS)	32	R/W	0000_0004h	39.4.8/1137
4003_B02C	ADC Plus-Side Gain Register (ADC0_PG)	32	R/W	0000_8200h	39.4.9/1138
4003_B030	ADC Minus-Side Gain Register (ADC0_MG)	32	R/W	0000_8200h	39.4.10/ 1138
4003_B034	ADC Plus-Side General Calibration Value Register (ADC0_CLPD)	32	R/W	0000_000Ah	39.4.11/ 1139
4003_B038	ADC Plus-Side General Calibration Value Register (ADC0_CLPS)	32	R/W	0000_0020h	39.4.12/ 1140
4003_B03C	ADC Plus-Side General Calibration Value Register (ADC0_CLP4)	32	R/W	0000_0200h	39.4.13/ 1140
4003_B040	ADC Plus-Side General Calibration Value Register (ADC0_CLP3)	32	R/W	0000_0100h	39.4.14/ 1141
4003_B044	ADC Plus-Side General Calibration Value Register (ADC0_CLP2)	32	R/W	0000_0080h	39.4.15/ 1141
4003_B048	ADC Plus-Side General Calibration Value Register (ADC0_CLP1)	32	R/W	0000_0040h	39.4.16/ 1142
4003_B04C	ADC Plus-Side General Calibration Value Register (ADC0_CLP0)	32	R/W	0000_0020h	39.4.17/ 1142
4003_B054	ADC Minus-Side General Calibration Value Register (ADC0_CLMD)	32	R/W	0000_000Ah	39.4.18/ 1143
4003_B058	ADC Minus-Side General Calibration Value Register (ADC0_CLMS)	32	R/W	0000_0020h	39.4.19/ 1143
4003_B05C	ADC Minus-Side General Calibration Value Register (ADC0_CLM4)	32	R/W	0000_0200h	39.4.20/ 1144
4003_B060	ADC Minus-Side General Calibration Value Register (ADC0_CLM3)	32	R/W	0000_0100h	39.4.21/ 1144
4003_B064	ADC Minus-Side General Calibration Value Register (ADC0_CLM2)	32	R/W	0000_0080h	39.4.22/ 1145
4003_B068	ADC Minus-Side General Calibration Value Register (ADC0_CLM1)	32	R/W	0000_0040h	39.4.23/ 1145
4003_B06C	ADC Minus-Side General Calibration Value Register (ADC0_CLM0)	32	R/W	0000_0020h	39.4.24/ 1146

39.4.1 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 than 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 (module disabled).

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: 4003_B000h base + 0h 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															
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								COCO	AIEN	DIFF	ADCH				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

ADCx_SC1n field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 COCO	Conversion Complete Flag 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. 0 Conversion is not completed. 1 Conversion is completed.
6 AIEN	Interrupt Enable Enables conversion complete interrupts. When COCO becomes set while the respective AIEN is high, an interrupt is asserted. 0 Conversion complete interrupt is disabled. 1 Conversion complete interrupt is enabled.
5 DIFF	Differential Mode Enable 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. 0 Single-ended conversions and input channels are selected. 1 Differential conversions and input channels are selected.
ADCH	Input channel select 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. **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. 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. 00000 When DIFF=0, DADP0 is selected as input; when DIFF=1, DAD0 is selected as input. 00001 When DIFF=0, DADP1 is selected as input; when DIFF=1, DAD1 is selected as input. 00010 When DIFF=0, DADP2 is selected as input; when DIFF=1, DAD2 is selected as input. 00011 When DIFF=0, DADP3 is selected as input; when DIFF=1, DAD3 is selected as input. 00100 When DIFF=0, AD4 is selected as input; when DIFF=1, it is reserved. 00101 When DIFF=0, AD5 is selected as input; when DIFF=1, it is reserved. 00110 When DIFF=0, AD6 is selected as input; when DIFF=1, it is reserved. 00111 When DIFF=0, AD7 is selected as input; when DIFF=1, it is reserved.

Table continues on the next page...

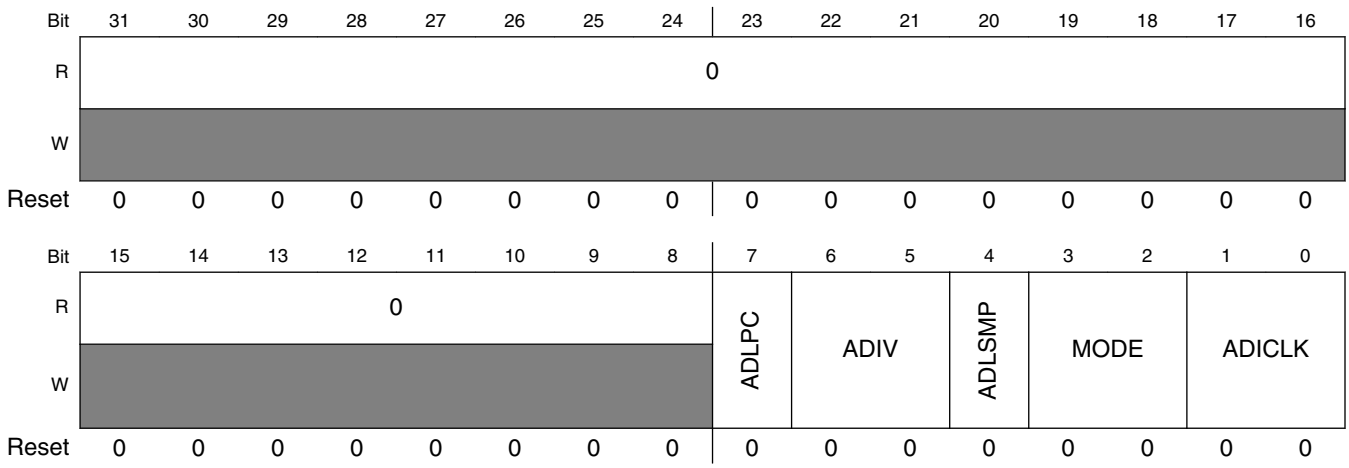
ADC*x*_SC1*n* field descriptions (continued)

Field	Description	
	01000	When DIFF=0, AD8 is selected as input; when DIFF=1, it is reserved.
	01001	When DIFF=0, AD9 is selected as input; when DIFF=1, it is reserved.
	01010	When DIFF=0, AD10 is selected as input; when DIFF=1, it is reserved.
	01011	When DIFF=0, AD11 is selected as input; when DIFF=1, it is reserved.
	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.

39.4.2 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: 4003_B000h base + 8h offset = 4003_B008h



ADCx_CFG1 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 ADLPC	Low-Power Configuration Controls the power configuration of the successive approximation converter. This optimizes power consumption when higher sample rates are not required. 0 Normal power configuration. 1 Low-power configuration. The power is reduced at the expense of maximum clock speed.
6–5 ADIV	Clock Divide Select Selects the divide ratio used by the ADC to generate the internal clock ADCK. 00 The divide ratio is 1 and the clock rate is input clock. 01 The divide ratio is 2 and the clock rate is (input clock)/2. 10 The divide ratio is 4 and the clock rate is (input clock)/4. 11 The divide ratio is 8 and the clock rate is (input clock)/8.
4 ADLSMP	Sample Time Configuration Selects between different sample times based on the conversion mode selected. This field 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.

Table continues on the next page...

ADCx_CFG1 field descriptions (continued)

Field	Description
	0 Short sample time. 1 Long sample time.
3–2 MODE	Conversion mode selection Selects the ADC resolution mode. 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. 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. 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 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
ADICLK	Input Clock Select 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. 00 Bus clock 01 Alternate clock 2 (ALTCLK2) 10 Alternate clock (ALTCLK) 11 Asynchronous clock (ADACK)

39.4.3 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: 4003_B000h base + Ch offset = 4003_B00Ch

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			MUXSEL	ADACKEN	ADHSC	ADLSTS	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_CFG2 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 MUXSEL	ADC Mux Select Changes the ADC mux setting to select between alternate sets of ADC channels. 0 ADxxa channels are selected. 1 ADxxb channels are selected.
3 ADACKEN	Asynchronous Clock Output Enable 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. 0 Asynchronous clock output disabled; Asynchronous clock is enabled only if selected by ADICLK and a conversion is active. 1 Asynchronous clock and clock output is enabled regardless of the state of the ADC.
2 ADHSC	High-Speed Configuration 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. 0 Normal conversion sequence selected. 1 High-speed conversion sequence selected with 2 additional ADCK cycles to total conversion time.
ADLSTS	Long Sample Time Select 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. 00 Default longest sample time; 20 extra ADCK cycles; 24 ADCK cycles total. 01 12 extra ADCK cycles; 16 ADCK cycles total sample time. 10 6 extra ADCK cycles; 10 ADCK cycles total sample time. 11 2 extra ADCK cycles; 6 ADCK cycles total sample time.

39.4.4 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-aligned 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 39-5. Data result register description

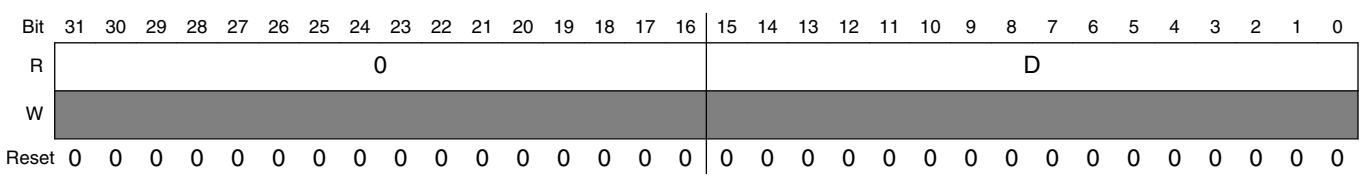
Conversion mode	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	Format
16-bit differential	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	Signed 2's complement
16-bit single-ended	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	Unsigned right justified
13-bit differential	S	S	S	S	D	D	D	D	D	D	D	D	D	D	D	D	Sign-extended 2's complement
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
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: 4003_B000h base + 10h offset + (4d × i), where i=0d to 1d



ADCx_R*n* field descriptions

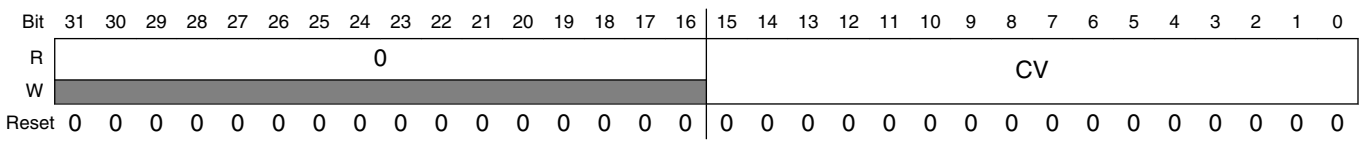
Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
D	Data result

39.4.5 Compare Value Registers (ADC*x*_CV*n*)

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: 4003_B000h base + 18h offset + (4d × i), where i=0d to 1d



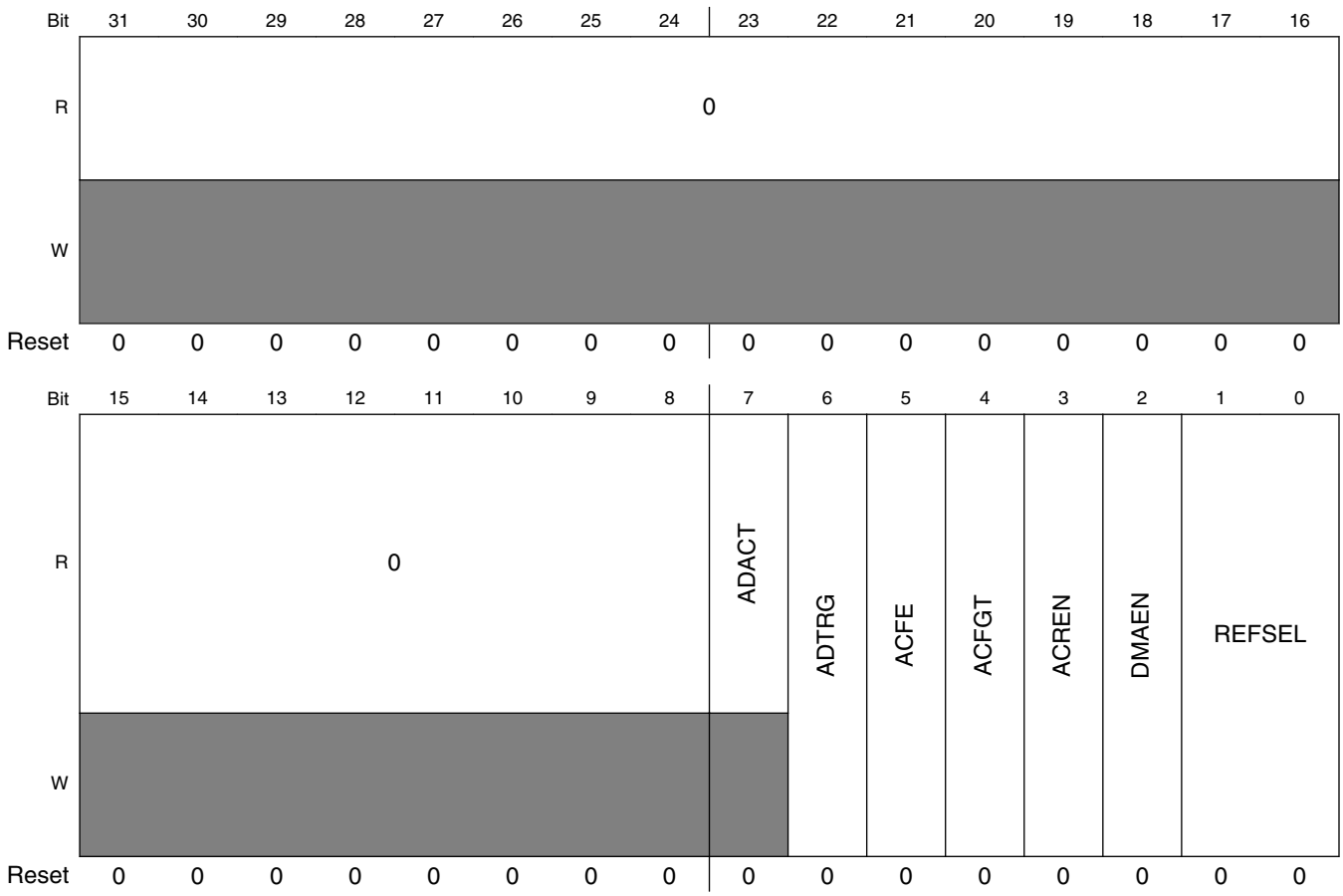
ADC*x*_CV*n* field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CV	Compare Value.

39.4.6 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: 4003_B000h base + 20h offset = 4003_B020h



ADCx_SC2 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 ADACT	Conversion Active 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. 0 Conversion not in progress. 1 Conversion in progress.
6 ADTRG	Conversion Trigger Select Selects the type of trigger used for initiating a conversion. Two types of trigger are selectable:

Table continues on the next page...

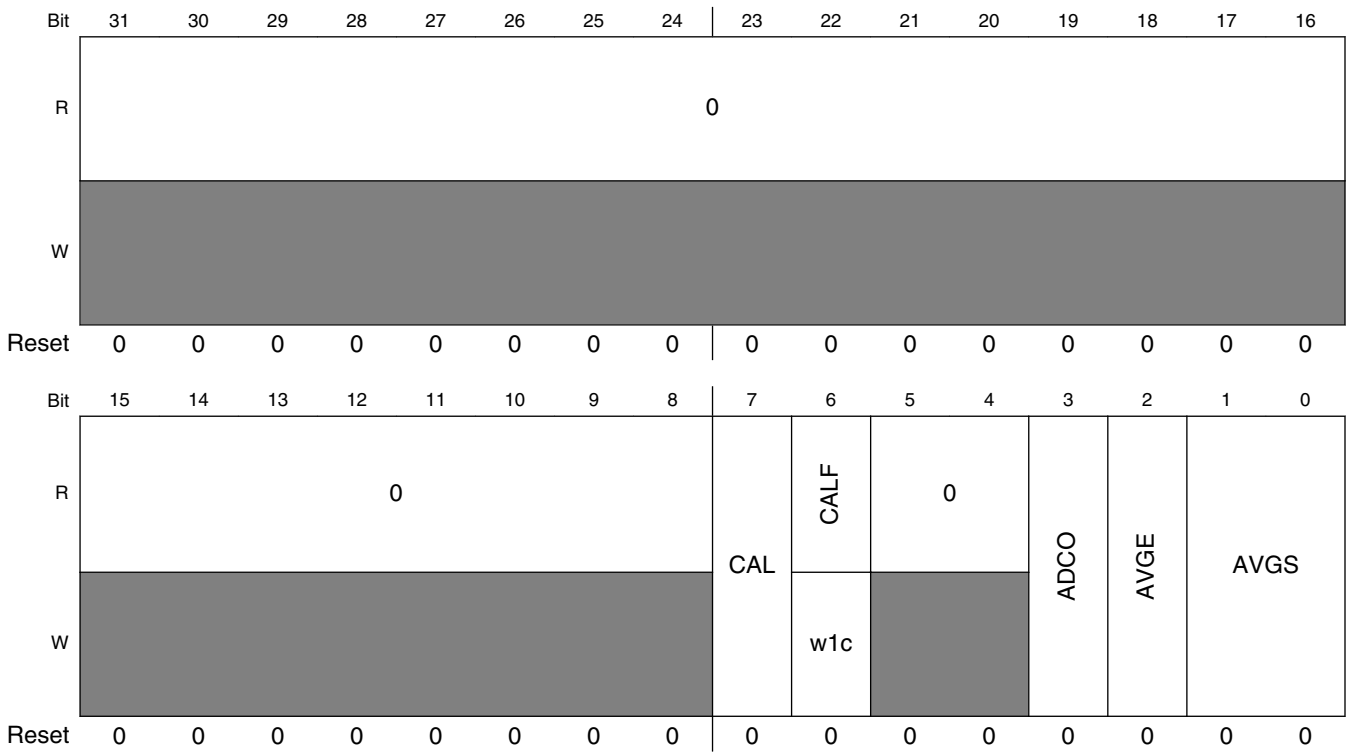
ADCx_SC2 field descriptions (continued)

Field	Description
	• Software trigger: When software trigger is selected, a conversion is initiated following a write to SC1A. • 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. 0 Software trigger selected. 1 Hardware trigger selected.
5 ACFE	Compare Function Enable Enables the compare function. 0 Compare function disabled. 1 Compare function enabled.
4 ACFGT	Compare Function Greater Than Enable 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. 0 Configures less than threshold, outside range not inclusive and inside range not inclusive; functionality based on the values placed in CV1 and CV2. 1 Configures greater than or equal to threshold, outside and inside ranges inclusive; functionality based on the values placed in CV1 and CV2.
3 ACREN	Compare Function Range Enable 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. 0 Range function disabled. Only CV1 is compared. 1 Range function enabled. Both CV1 and CV2 are compared.
2 DMAEN	DMA Enable 0 DMA is disabled. 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.
REFSEL	Voltage Reference Selection Selects the voltage reference source used for conversions. 00 Default voltage reference pin pair, that is, external pins V_{REFH} and V_{REFL} 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 10 Reserved 11 Reserved

39.4.7 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: 4003_B000h base + 24h offset = 4003_B024h



ADCx_SC3 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 CAL	Calibration 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 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 CALF	Calibration Failed Flag 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. 0 Calibration completed normally. 1 Calibration failed. ADC accuracy specifications are not guaranteed.

Table continues on the next page...

ADCx_SC3 field descriptions (continued)

Field	Description
5–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 ADCO	Continuous Conversion Enable Enables continuous conversions. 0 One conversion or one set of conversions if the hardware average function is enabled, that is, AVGE=1, after initiating a conversion. 1 Continuous conversions or sets of conversions if the hardware average function is enabled, that is, AVGE=1, after initiating a conversion.
2 AVGE	Hardware Average Enable Enables the hardware average function of the ADC. 0 Hardware average function disabled. 1 Hardware average function enabled.
AVGS	Hardware Average Select Determines how many ADC conversions will be averaged to create the ADC average result. 00 4 samples averaged. 01 8 samples averaged. 10 16 samples averaged. 11 32 samples averaged.

39.4.8 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.

For more information regarding the calibration procedure, please refer to the Calibration function section.

Address: 4003_B000h base + 28h offset = 4003_B028h

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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
OFS	Offset Error Correction Value

39.4.9 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 PG[15] and PG[14]. 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.

For more information regarding the calibration procedure, please refer to the Calibration function section.

Address: 4003_B000h base + 2Ch offset = 4003_B02Ch

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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
PG	Plus-Side Gain

39.4.10 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 MG[15] and MG[14]. 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.

For more information regarding the calibration procedure, please refer to the Calibration function section.

Address: 4003_B000h base + 30h offset = 4003_B030h

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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
MG	Minus-Side Gain

39.4.11 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.

For more information regarding the calibration procedure, please refer to the Calibration function section.

Address: 4003_B000h base + 34h offset = 4003_B034h

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																										CLPD					
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_CLPD field descriptions

Field	Description
31–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLPD	Calibration Value Calibration Value

39.4.12 ADC Plus-Side General Calibration Value Register (ADCx_CLPS)

For more information, see CLPD register description.

Address: 4003_B000h base + 38h offset = 4003_B038h

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																										CLPS					
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_CLPS field descriptions

Field	Description
31–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLPS	Calibration Value Calibration Value

39.4.13 ADC Plus-Side General Calibration Value Register (ADCx_CLP4)

For more information, see CLPD register description.

Address: 4003_B000h base + 3Ch offset = 4003_B03Ch

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																										CLP4					
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

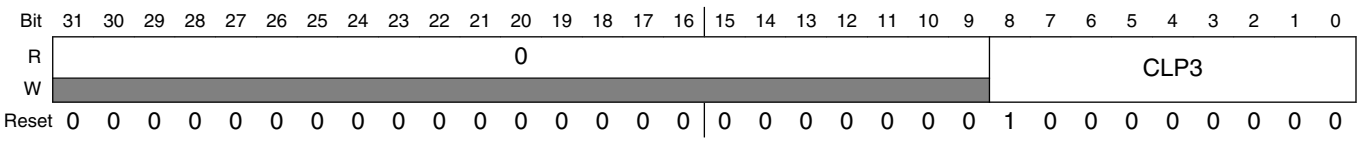
ADCx_CLP4 field descriptions

Field	Description
31–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLP4	Calibration Value Calibration Value

39.4.14 ADC Plus-Side General Calibration Value Register (ADCx_CLP3)

For more information, see CLPD register description.

Address: 4003_B000h base + 40h offset = 4003_B040h



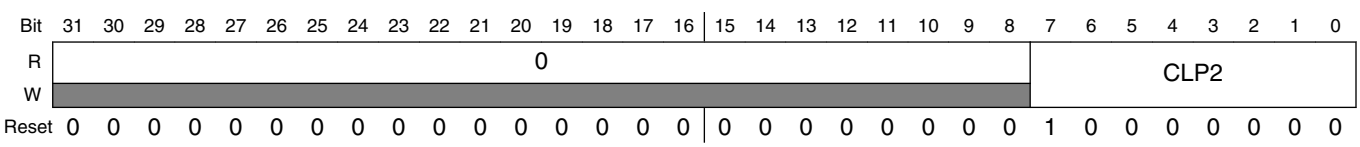
ADCx_CLP3 field descriptions

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLP3	Calibration Value Calibration Value

39.4.15 ADC Plus-Side General Calibration Value Register (ADCx_CLP2)

For more information, see CLPD register description.

Address: 4003_B000h base + 44h offset = 4003_B044h



ADCx_CLP2 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLP2	Calibration Value Calibration Value

39.4.16 ADC Plus-Side General Calibration Value Register (ADCx_CLP1)

For more information, see CLPD register description.

Address: 4003_B000h base + 48h offset = 4003_B048h

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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLP1	Calibration Value Calibration Value

39.4.17 ADC Plus-Side General Calibration Value Register (ADCx_CLP0)

For more information, see CLPD register description.

Address: 4003_B000h base + 4Ch offset = 4003_B04Ch

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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLP0	Calibration Value Calibration Value

39.4.18 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.

For more information regarding the calibration procedure, please refer to the Calibration function section.

Address: 4003_B000h base + 54h offset = 4003_B054h

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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLMD	Calibration Value Calibration Value

39.4.19 ADC Minus-Side General Calibration Value Register (ADCx_CLMS)

For more information, see CLMD register description.

Address: 4003_B000h base + 58h offset = 4003_B058h

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

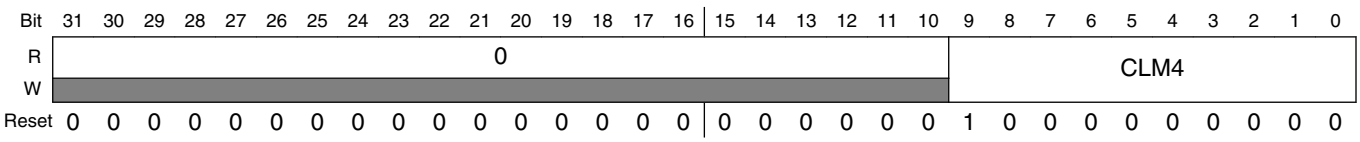
ADCx_CLMS field descriptions

Field	Description
31–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLMS	Calibration Value Calibration Value

39.4.20 ADC Minus-Side General Calibration Value Register (ADCx_CLM4)

For more information, see CLMD register description.

Address: 4003_B000h base + 5Ch offset = 4003_B05Ch



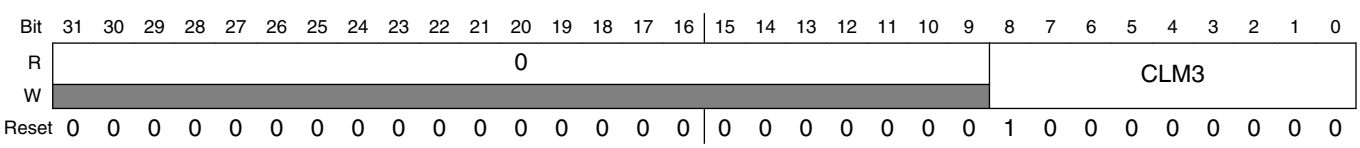
ADCx_CLM4 field descriptions

Field	Description
31–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLM4	Calibration Value Calibration Value

39.4.21 ADC Minus-Side General Calibration Value Register (ADCx_CLM3)

For more information, see CLMD register description.

Address: 4003_B000h base + 60h offset = 4003_B060h



ADCx_CLM3 field descriptions

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

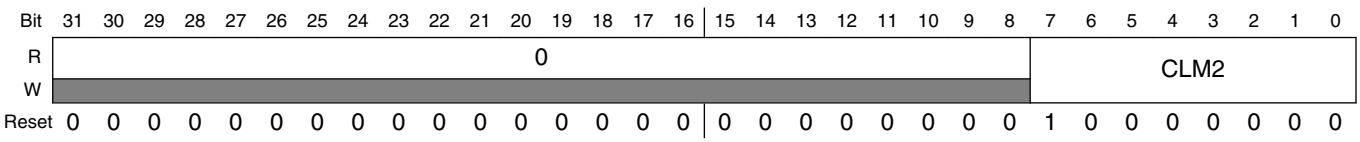
ADCx_CLM3 field descriptions (continued)

Field	Description
CLM3	Calibration Value Calibration Value

39.4.22 ADC Minus-Side General Calibration Value Register (ADCx_CLM2)

For more information, see CLMD register description.

Address: 4003_B000h base + 64h offset = 4003_B064h



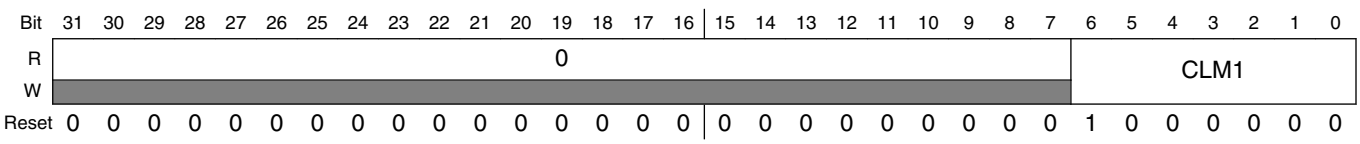
ADCx_CLM2 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLM2	Calibration Value Calibration Value

39.4.23 ADC Minus-Side General Calibration Value Register (ADCx_CLM1)

For more information, see CLMD register description.

Address: 4003_B000h base + 68h offset = 4003_B068h



ADCx_CLM1 field descriptions

Field	Description
31–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLM1	Calibration Value Calibration Value

39.4.24 ADC Minus-Side General Calibration Value Register (ADCx_CLM0)

For more information, see CLMD register description.

Address: 4003_B000h base + 6Ch offset = 4003_B06Ch

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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLM0	Calibration Value Calibration Value

39.5 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.

39.5.1 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.
- ALTCLK2: As defined for this MCU. See the chip configuration information. Conversions are possible using ALTCLK2 as the input clock source while the MCU is in Normal Stop mode.
- ALTCLK: As defined for this MCU. See the chip configuration information. Conversions are possible using ALTCLK as the input clock source while the MCU is in Normal Stop mode.
- 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.

39.5.2 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.

39.5.3 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-specific ADC information 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, 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.

39.5.4 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

39.5.4.1 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 SC3[ADCO] = 1.

If continuous conversions are enabled, a new conversion is automatically initiated after the completion of the current conversion. In software triggered operation, that is, when SC2[ADTRG] = 0, continuous conversions begin after SC1A is written and continue until aborted. In hardware triggered operation, that is, when SC2[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.

39.5.4.2 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.

39.5.4.3 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 or Alternate Clock Sources 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.

39.5.4.4 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}.

39.5.4.5 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.

$$\text{ConversionTime} = \text{SFCAdder} + \text{AverageNum} \times (\text{BCT} + \text{LSTAdder} + \text{HSCAdder})$$

Equation 1. Conversion time equation

Table 39-6. 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 39-7. 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 39-8. 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

Table continues on the next page...

Table 39-8. Base conversion time (BCT) (continued)

Mode	Base conversion time (BCT)
16b single-ended	25 ADCK cycles
16b differential	34 ADCK cycles

Table 39-9. 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 39-10. 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.

39.5.4.6 Conversion time examples

The following examples use the Equation 1 on page 1153, and the information provided in Table 39-6 through Table 39-10.

39.5.4.6.1 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 Equation 1 on page 1153, and the information provided in Table 39-6 through Table 39-10. The table below lists the variables of Equation 1 on page 1153.

Table 39-11. Typical conversion time

Variable	Time
SFCAdder	5 ADCK cycles + 5 bus clock cycles
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.

39.5.4.6.2 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 Equation 1 on page 1153, and the information provided in Table 39-6 through Table 39-10. The following table lists the variables of the Equation 1 on page 1153.

Table 39-12. 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.

39.5.4.6.3 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 Equation 1 on page 1153, and the information provided in Table 39-6 through Table 39-10. The table below lists the variables of Equation 1 on page 1153.

Table 39-13. 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.

39.5.4.7 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] is set.

39.5.5 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 39-14. 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.

39.5.6 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.

Further information on the calibration procedure can be found in the Calibration section of AN3949: ADC16 Calibration Procedure and Programmable Delay Block Synchronization.

39.5.7 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.

39.5.8 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.

$$Temp = 25 - \left(\left(V_{TEMP} - V_{TEMP25} \right) \div m \right)$$

Equation 2. 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).

39.5.9 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, ADACK, and Alternate Clock sources 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.

39.5.10 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.

39.5.10.1 Normal Stop mode with Bus Clock selected

If the Bus Clock is 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.

39.5.10.2 Normal Stop mode with ADACK or Alternate clock sources enabled

If ADACK or an Alternate clock source is selected as the conversion clock, the ADC continues operation during Normal Stop mode. See the chip-specific ADC information for configuration information for this device.

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.

39.5.11 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.

39.6 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 39-9, Table 39-10, and Table 39-11.

Note

Hexadecimal values are designated by a preceding 0x, binary values designated by a preceding %, and decimal values have no preceding character.

39.6.1 ADC module initialization example

39.6.1.1 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.

39.6.1.2 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.
```

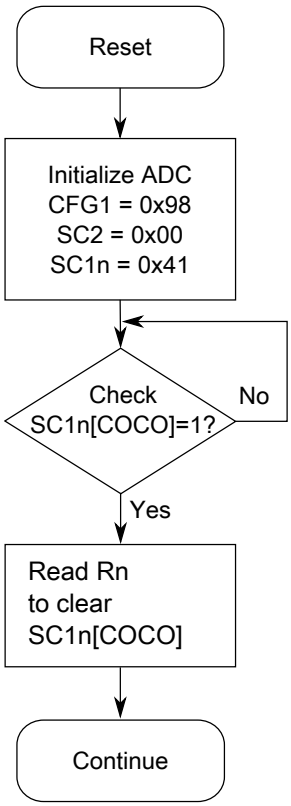


Figure 39-3. Initialization flowchart example

39.7 Application information

The ADC has been designed to be integrated into a microcontroller for use in embedded control applications requiring an ADC.

For guidance on selecting optimum external component values and converter parameters see AN4373: Cookbook for SAR ADC Measurements.

39.7.1 External pins and routing

39.7.1.1 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.

39.7.1.2 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.

39.7.1.3 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.

39.7.2 Sources of error

39.7.2.1 Sampling error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy.

$$RAS + RADIN = SC / (FMAX * NUMTAU * CADIN)$$

Figure 39-4. 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.

39.7.2.2 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.

39.7.2.3 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 or an Alternate clock 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.

39.7.2.4 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:

$$1LSB=\left(V_{REFH}\right)/2^{N}$$

Equation 3. 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.

39.7.2.5 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.

39.7.2.6 Code jitter, non-monotonicity, and missing codes

Analog-to-digital converters are susceptible to three special forms of error:

- Code jitter: Code jitter occurs when 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 occurs 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 40
Comparator (CMP)

40.1 Chip-specific Comparator information

40.1.1 CMP instantiation information

Table 40-1. CMP Instantiation Information

CMP configurable feature	KL80_128
Number of 6-bit DACs	1
Analog Mux size	8 input
Number of CMP	1
Ext Inputs on HSCMP0	5

NOTE

CMP0_out inter-connected to the RX pin of LPUART0 and LPUART1 on chip. See SIM for details.

The device includes one high-speed comparator and two 8-input multiplexers for both theinverting and non-inverting inputs of the comparator. Each CMP input channel connectsto both muxes. Two of the channels are connected to internal sources, leaving resourcesto support up to 6 input pins. See the channel assignment table for a summary of CMPinput connections for this device.

The CMP also includes one 6-bit DAC with a 64-tap resistor ladder network, whichprovides a selectable voltage reference for applications where voltage reference is neededfor internal connection to the CMP.The CMP can be optionally on in all modes except VLLS0.

The CMP has several module-to-module interconnects in order to facilitate ADCtriggering, TPM triggering, and infrared interfaces. For complete details on the CMPmodule interconnects, see the Module-to-Module section.

The CMP supports sample and window compare function. LPTPM2 CH0 and CH1 output control the CMP Sample/Window timing. The assert of LPTPM CH0 will set WINDOW/SAMPLE = 1 while the assert of LPTPM CH1 will set WINDOW/SAMPLE = 0.

Due to the pin number limitation, the CMP pass through mode is not supported by this device, so the CMPx_MUXCR[PSTM] must be left as 0.

40.1.2 CMP input connections

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

Table 40-2. CMP input connections

CMP Inputs	CMP0
IN0	CMP0_IN0
IN1	CMP0_IN1
IN2	CMP0_IN2
IN3	CMP0_IN3
IN5	VREF Output/CMP0_IN5
IN6	Bandgap
IN7	6b DAC0 Reference

40.1.3 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

40.1.4 CMP trigger mode

The CMP and 6-bit DAC sub-block supports trigger mode operation when the CMPx_CR1[TRIGM] is set. When trigger mode is enabled, the trigger event will initiate a compare sequence that must first enable the CMP and DAC prior to performing a CMP operation and capturing the output. In this device, control for this two staged sequencing is provided from the LPTMR0. The LPTMR0 provides a single trigger output to all implemented comparators. Through configuration of the CMPx_CR1[TRIGM] bits the trigger can be used to trigger a single comparator or multiple comparators concurrently.

The LPTMR0 triggering output is always enabled when the LPTMR0 is enabled. The first signal is supplied to enable the CMP and DAC and is asserted at the same time as the TCF flag is set. The delay to the second signal that triggers the CMP to capture the result of the compare operation is dependent on the LPTMR0 configuration. In Time Counter mode with prescaler enabled, the delay is 1/2 Prescaler output period. In Time Counter mode with prescaler bypassed, the delay is 1/2 Prescaler clock period.

The delay between the first signal from LPTMR0 and the second signal from LPTMR0 must be greater than the Analog comparator initialization delay as defined in the device datasheet.

40.2 Introduction

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.

40.2.1 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 except VLLS0
- The window and filter functions are not available in the following modes:
    - Stop
    - VLPS
    - LLS
    - VLLSx

40.2.2 6-bit DAC key features

The 6-bit DAC has the following 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

40.2.3 ANMUX key features

The ANMUX has the following features:

- Two 8-to-1 channel mux
- Operational over the entire supply range

40.2.4 CMP, DAC and ANMUX diagram

The following figure shows the block diagram for the High-Speed Comparator, DAC, and ANMUX modules.

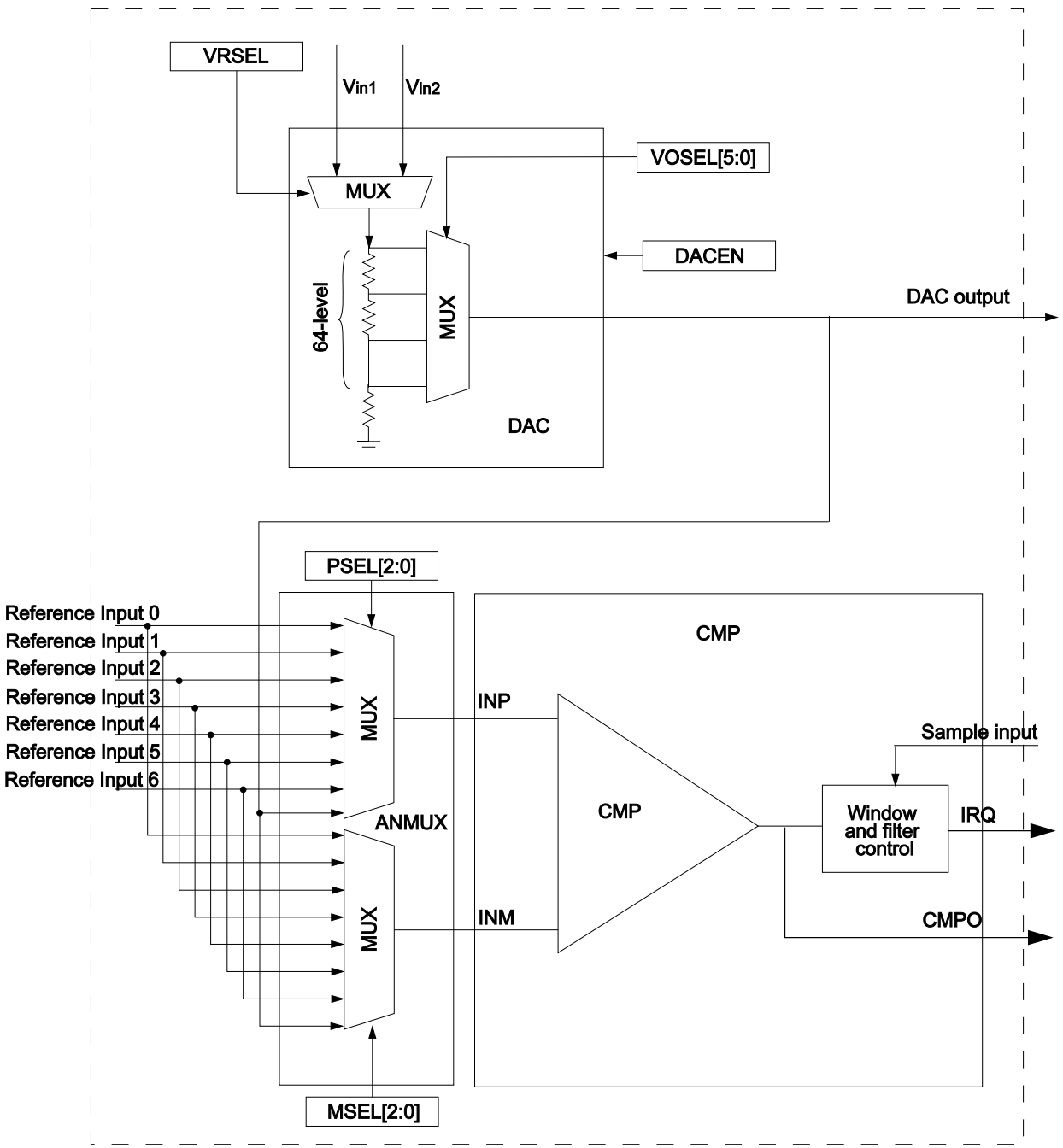


Figure 40-1. CMP, DAC and ANMUX block diagram

40.2.5 CMP block diagram

The following figure shows the block diagram for the CMP module.

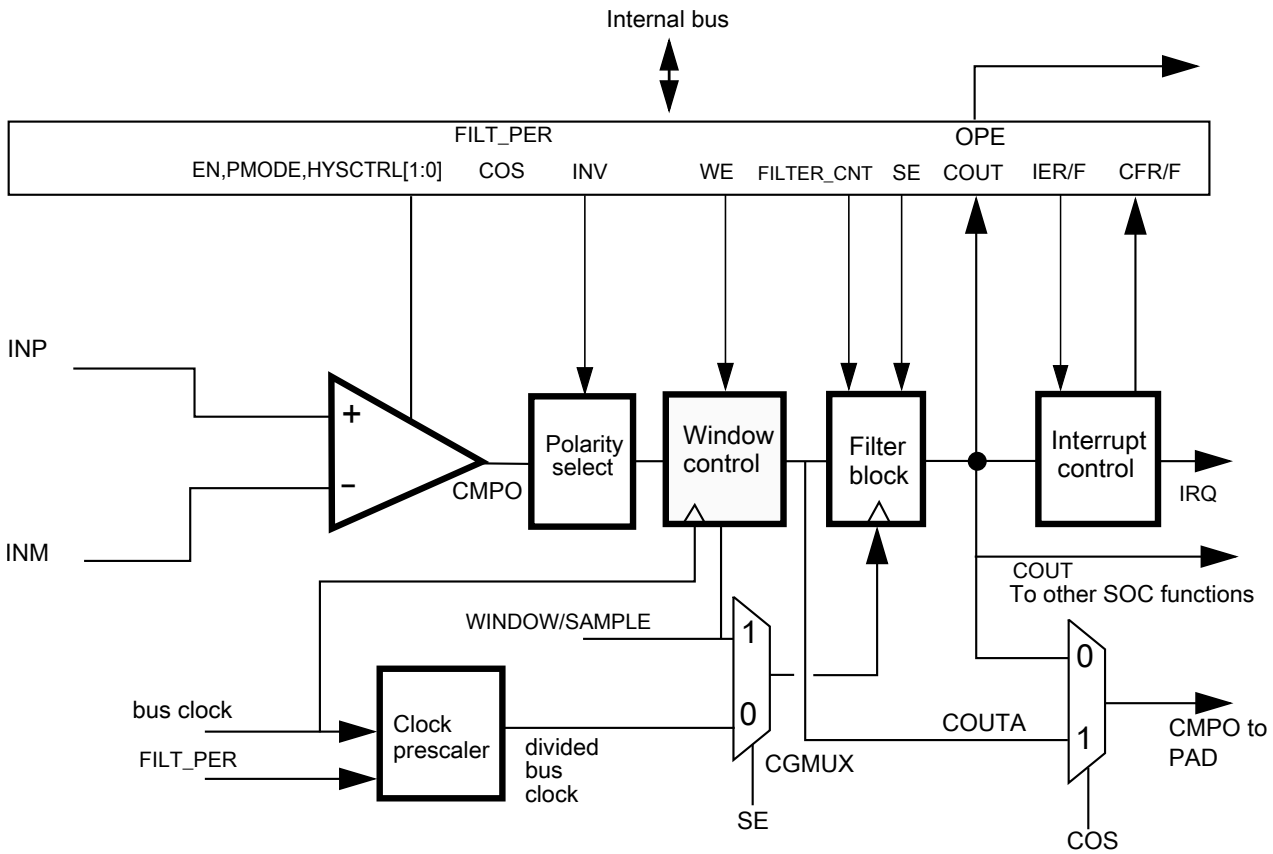


Figure 40-2. Comparator module block diagram

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.

40.3 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	40.3.1/1179
4007_3001	CMP Control Register 1 (CMP0_CR1)	8	R/W	00h	40.3.2/1180
4007_3002	CMP Filter Period Register (CMP0_FPR)	8	R/W	00h	40.3.3/1182
4007_3003	CMP Status and Control Register (CMP0_SCR)	8	R/W	00h	40.3.4/1182
4007_3004	DAC Control Register (CMP0_DACCR)	8	R/W	00h	40.3.5/1183
4007_3005	MUX Control Register (CMP0_MUXCR)	8	R/W	00h	40.3.6/1184

40.3.1 CMP Control Register 0 (CMPx_CR0)

Address: 4007_3000h base + 0h offset = 4007_3000h

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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–4 FILTER_CNT	Filter Sample Count 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. 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.

Table continues on the next page...

CMPx_CR0 field descriptions (continued)

Field	Description
	001 One sample must agree. The comparator output is simply sampled. 010 2 consecutive samples must agree. 011 3 consecutive samples must agree. 100 4 consecutive samples must agree. 101 5 consecutive samples must agree. 110 6 consecutive samples must agree. 111 7 consecutive samples must agree.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
HYSTCTR	Comparator hard block hysteresis control 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. 00 Level 0 01 Level 1 10 Level 2 11 Level 3

40.3.2 CMP Control Register 1 (CMPx_CR1)

Address: 4007_3000h base + 1h offset = 4007_3001h

Bit	7	6	5	4	3	2	1	0
Read Write	SE	WE	TRIGM	PMODE	INV	COS	OPE	EN
Reset	0	0	0	0	0	0	0	0

CMPx_CR1 field descriptions

Field	Description
7 SE	Sample Enable 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. 0 Sampling mode is not selected. 1 Sampling mode is selected.
6 WE	Windowing Enable 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 Windowing mode is not selected. 1 Windowing mode is selected.
5 TRIGM	Trigger Mode Enable CMP and DAC are configured to CMP Trigger mode when CMP_CR1[TRIGM] is set to 1. In addition, the CMP should be enabled. If the DAC is to be used as a reference to the CMP, it should also be enabled. CMP Trigger mode depends on an external timer resource to periodically enable the CMP and 6-bit DAC in order to generate a triggered compare. Upon setting TRIGM, the CMP and DAC are placed in a standby state until an external timer resource trigger is received. See the chip configuration for details about the external timer resource. 0 Trigger mode is disabled. 1 Trigger mode is enabled.
4 PMODE	Power Mode Select See the electrical specifications table in the device Data Sheet for details. 0 Low-Speed (LS) Comparison mode selected. In this mode, CMP has slower output propagation delay and lower current consumption. 1 High-Speed (HS) Comparison mode selected. In this mode, CMP has faster output propagation delay and higher current consumption.
3 INV	Comparator INVERT 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. 0 Does not invert the comparator output. 1 Inverts the comparator output.
2 COS	Comparator Output Select 0 Set the filtered comparator output (CMPO) to equal COUT. 1 Set the unfiltered comparator output (CMPO) to equal COUTA.
1 OPE	Comparator Output Pin Enable 0 CMPO is not available on the associated CMPO output pin. If the comparator does not own the pin, this field has no effect. 1 CMPO is available on the associated CMPO output pin. 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 EN	Comparator Module Enable 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. 0 Analog Comparator is disabled. 1 Analog Comparator is enabled.

40.3.3 CMP Filter Period Register (CMPx_FPR)

Address: 4007_3000h base + 2h offset = 4007_3002h

Bit	7	6	5	4	3	2	1	0
Read Write	FILT_PER							
Reset	0	0	0	0	0	0	0	0

CMPx_FPR field descriptions

Field	Description
FILT_PER	Filter Sample Period 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. This field has no effect when CR1[SE]=1. In that case, the external SAMPLE signal is used to determine the sampling period.

40.3.4 CMP Status and Control Register (CMPx_SCR)

Address: 4007_3000h base + 3h offset = 4007_3003h

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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 DMAEN	DMA Enable Control 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. 0 DMA is disabled. 1 DMA is enabled.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 IER	Comparator Interrupt Enable Rising 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. 1 Interrupt is enabled.
3 IEF	Comparator Interrupt Enable Falling Enables the CFF interrupt from the CMP. When this field is set, an interrupt will be asserted when CFF is set. 0 Interrupt is disabled. 1 Interrupt is enabled.
2 CFR	Analog Comparator Flag Rising 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 . 0 Rising-edge on COUT has not been detected. 1 Rising-edge on COUT has occurred.
1 CFF	Analog Comparator Flag Falling 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 . 0 Falling-edge on COUT has not been detected. 1 Falling-edge on COUT has occurred.
0 COUT	Analog Comparator Output 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.

40.3.5 DAC Control Register (CMPx_DACCR)

Address: 4007_3000h base + 4h offset = 4007_3004h

Bit	7	6	5	4	3	2	1	0
Read Write	DACEN	VRSEL	VOSEL					
Reset	0	0	0	0	0	0	0	0

CMPx_DACCR field descriptions

Field	Description
7 DACEN	DAC Enable Enables the DAC. When the DAC is disabled, it is powered down to conserve power. 0 DAC is disabled. 1 DAC is enabled.
6 VRSEL	Supply Voltage Reference Source Select

Table continues on the next page...

CMPx_DACCR field descriptions (continued)

Field	Description
	0 V_{in1} is selected as resistor ladder network supply reference. 1 V_{in2} is selected as resistor ladder network supply reference.
VOSEL	DAC Output Voltage Select Selects an output voltage from one of 64 distinct levels. `DACO = (V in /64) * (VOSEL[5:0] + 1)` , so the DACO range is from V_{in} /64 to V_{in} .

40.3.6 MUX Control Register (CMPx_MUXCR)

Address: 4007_3000h base + 5h offset = 4007_3005h

Bit	7	6	5	4	3	2	1	0
Read Write	Reserved	0	PSEL			MSEL		
Reset	0	0	0	0	0	0	0	0

CMPx_MUXCR field descriptions

Field	Description
7 Reserved	Bit can be programmed to zero only . This field is reserved.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–3 PSEL	Plus Input Mux Control Determines which input is selected for the plus input of the comparator. For INx inputs, see CMP, DAC, and ANMUX block diagrams. **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. 000 IN0 001 IN1 010 IN2 011 IN3 100 IN4 101 IN5 110 IN6 111 IN7
MSEL	Minus Input Mux Control Determines which input is selected for the minus input of the comparator. For INx inputs, see CMP, DAC, and ANMUX block diagrams. **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. 000 IN0

Table continues on the next page...

CMPx_MUXCR field descriptions (continued)

Field	Description
	001 IN1 010 IN2 011 IN3 100 IN4 101 IN5 110 IN6 111 IN7

40.4 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].

40.4.1 CMP functional modes

There are the following 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 40-3. 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** See the Disabled mode (# 1).
2A	1	0	0	0x00	X	**Continuous Mode** See the Continuous mode (#s 2A & 2B).
2B	1	0	0	X	0x00	
3A	1	0	1	0x01	X	**Sampled, Non-Filtered mode** See the Sampled, Non-Filtered mode (#s 3A & 3B).
3B	1	0	0	0x01	> 0x00	
4A	1	0	1	> 0x01	X	**Sampled, Filtered mode** See the Sampled, Filtered mode (#s 4A & 4B).
4B	1	0	0	> 0x01	> 0x00	
5A	1	1	0	0x00	X	**Windowed mode** Comparator output is sampled on every rising bus clock edge when SAMPLE=1 to generate COUTA. See the Windowed mode (#s 5A & 5B).
5B	1	1	0	X	0x00	
6	1	1	0	0x01	0x01–0xFF	**Windowed/Resampled mode** 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. See the Windowed/Resampled mode (# 6).
7	1	1	0	> 0x01	0x01–0xFF	**Windowed/Filtered mode** 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. 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.

40.4.1.1 Disabled mode (# 1)

In Disabled mode, the analog comparator is non-functional and consumes no power. CMPO is 0 in this mode.

40.4.1.2 Continuous mode (#s 2A & 2B)

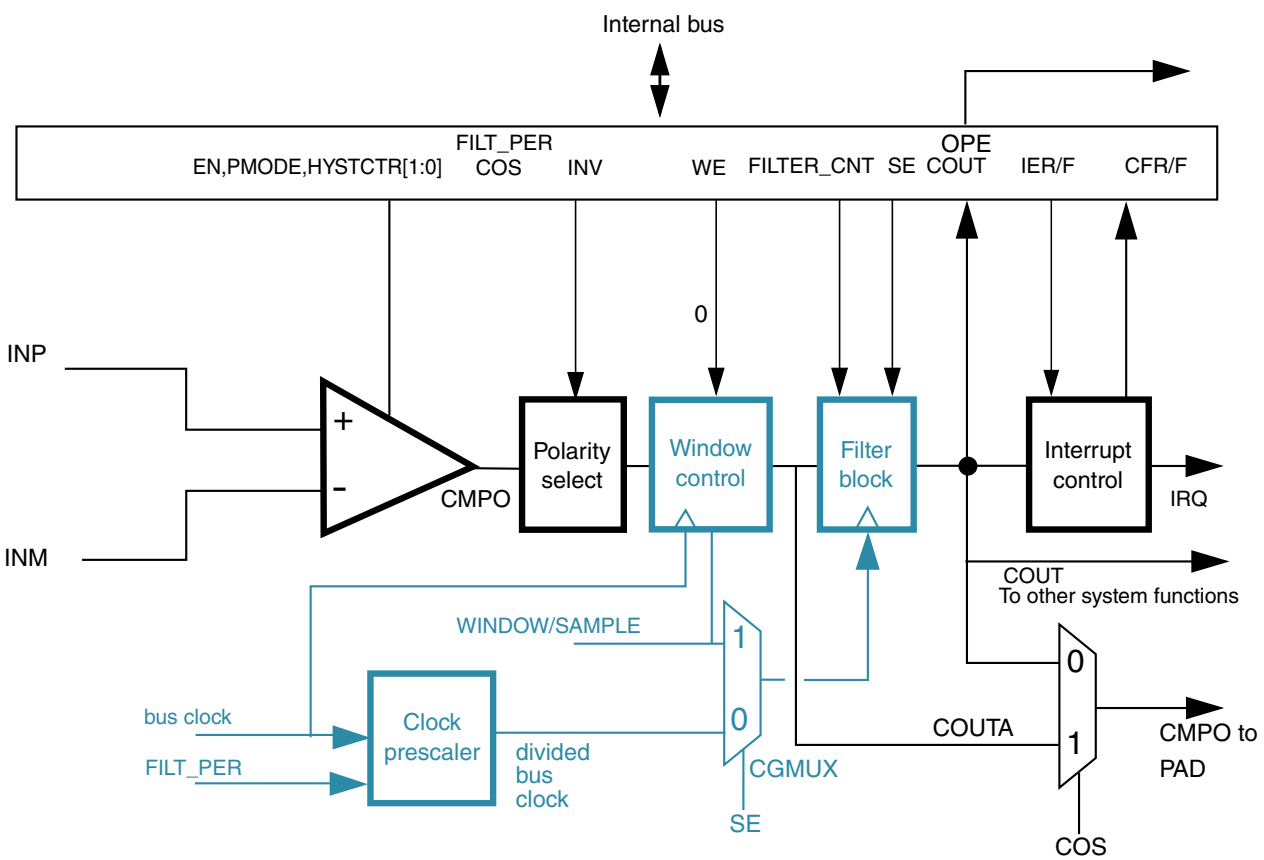


Figure 40-3. Comparator operation in Continuous mode

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.

40.4.1.3 Sampled, Non-Filtered mode (#s 3A & 3B)

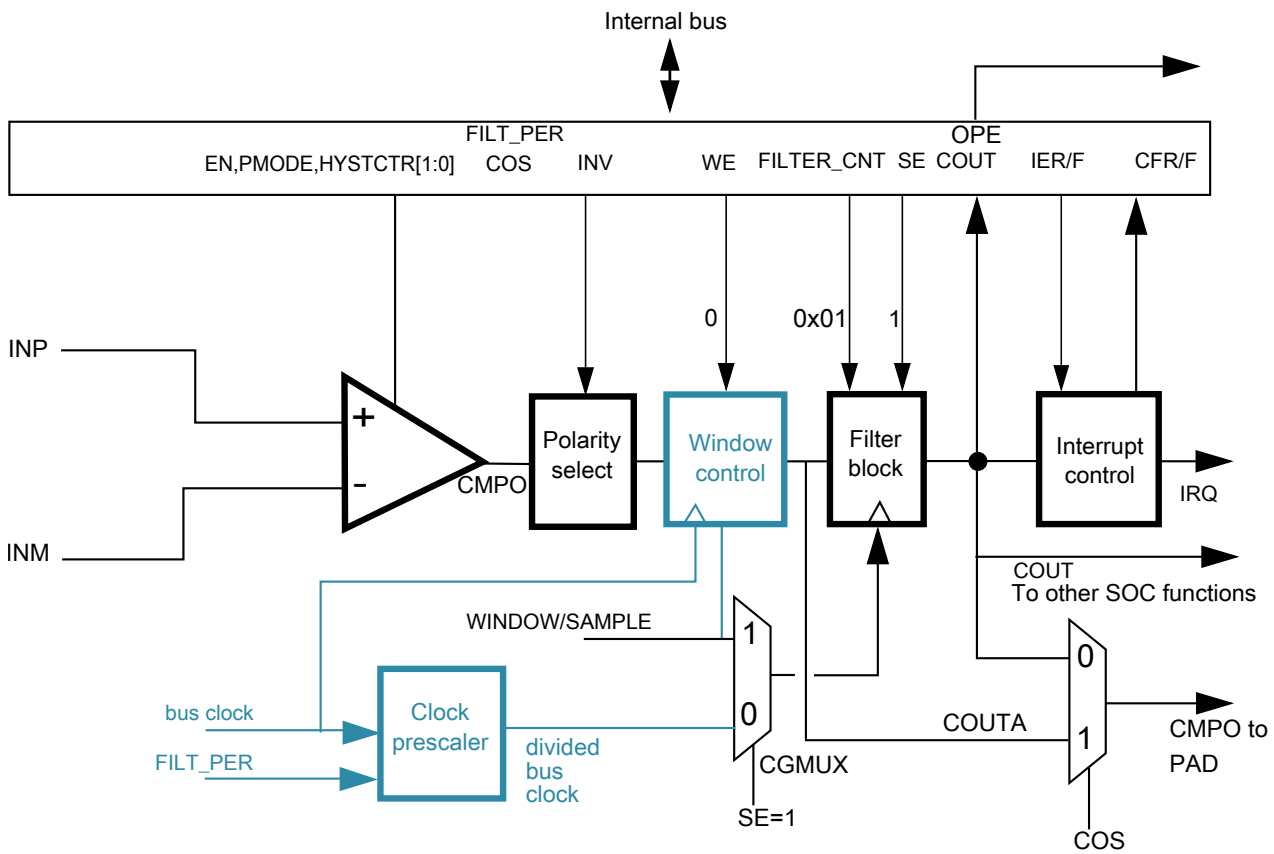


Figure 40-4. Sampled, Non-Filtered (# 3A): sampling point externally driven

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).

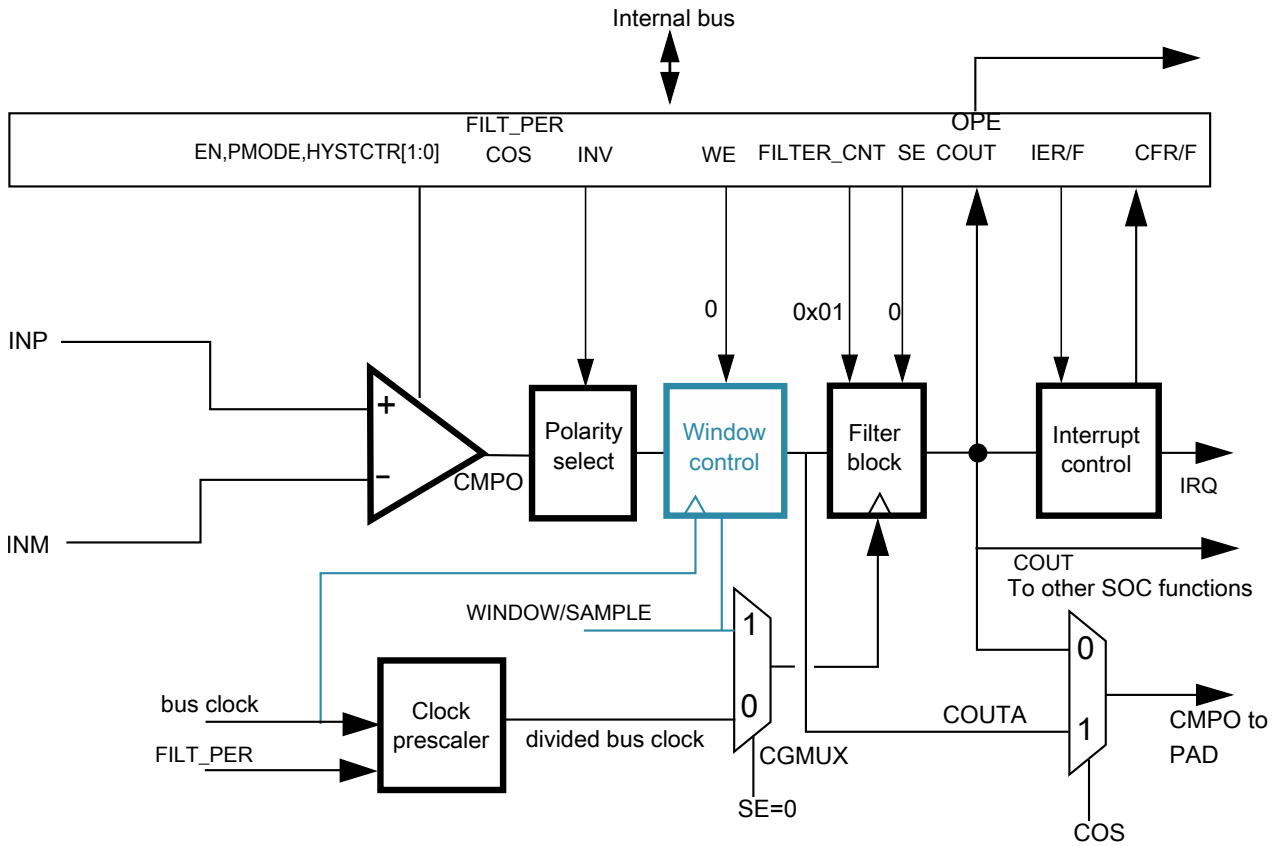


Figure 40-5. Sampled, Non-Filtered (# 3B): sampling interval internally derived

40.4.1.4 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.

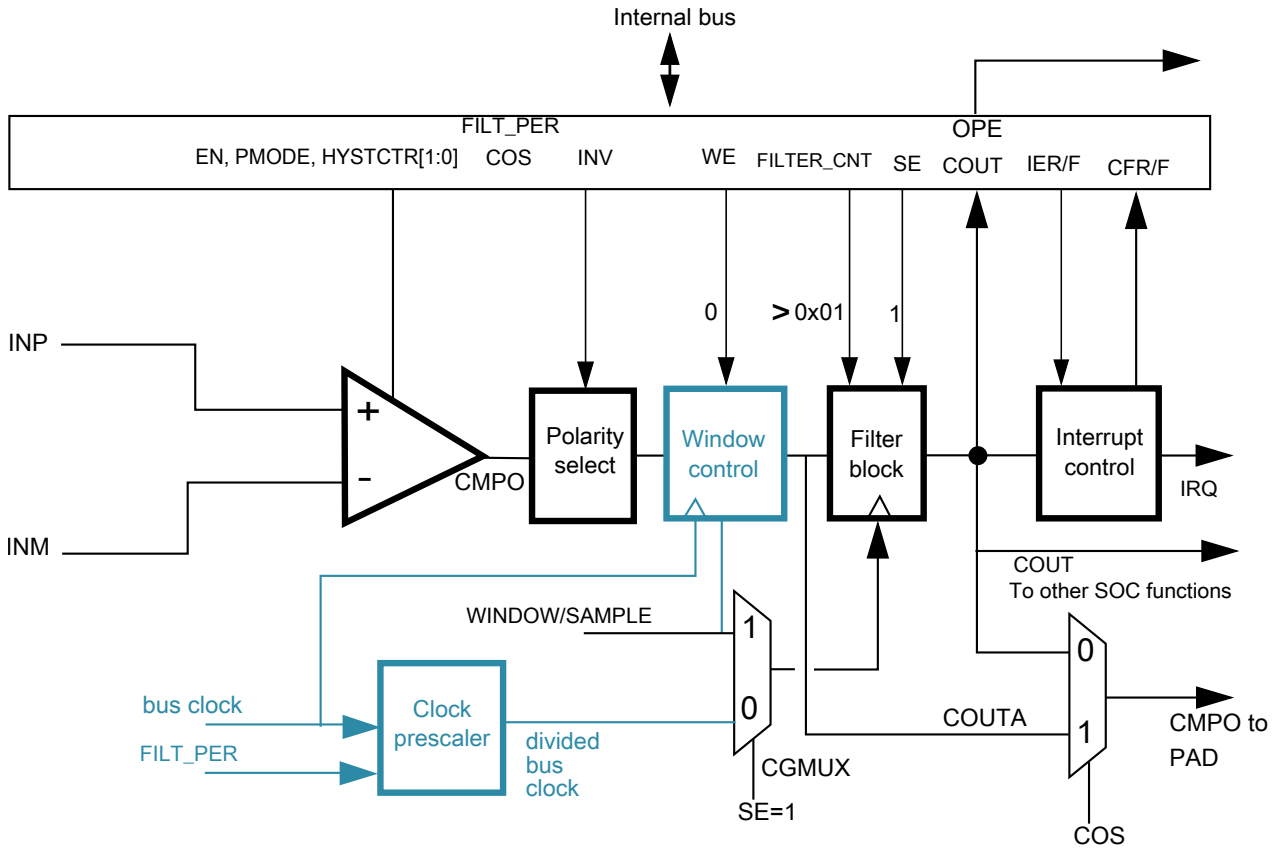


Figure 40-6. Sampled, Filtered (# 4A): sampling point externally driven

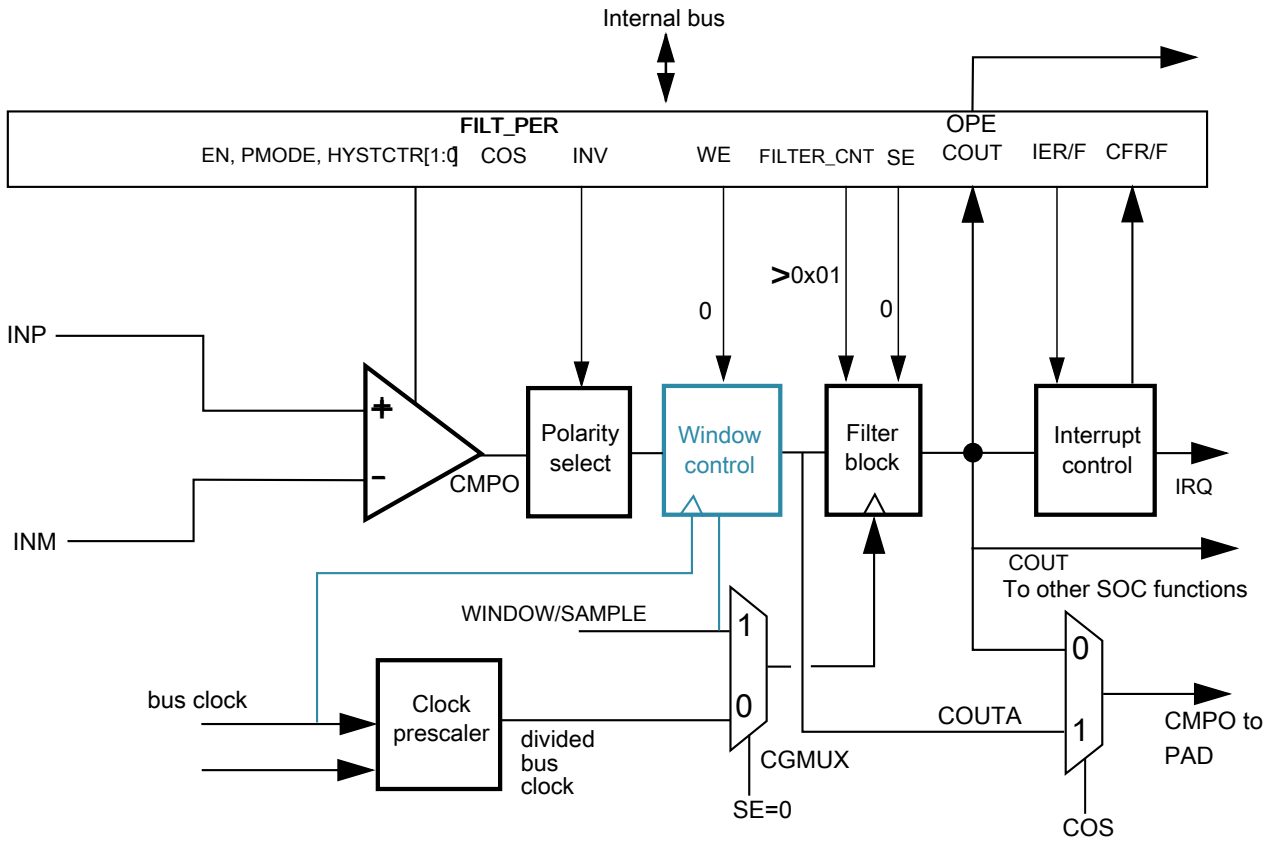


Figure 40-7. Sampled, Filtered (# 4B): sampling point internally derived

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.

40.4.1.5 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.

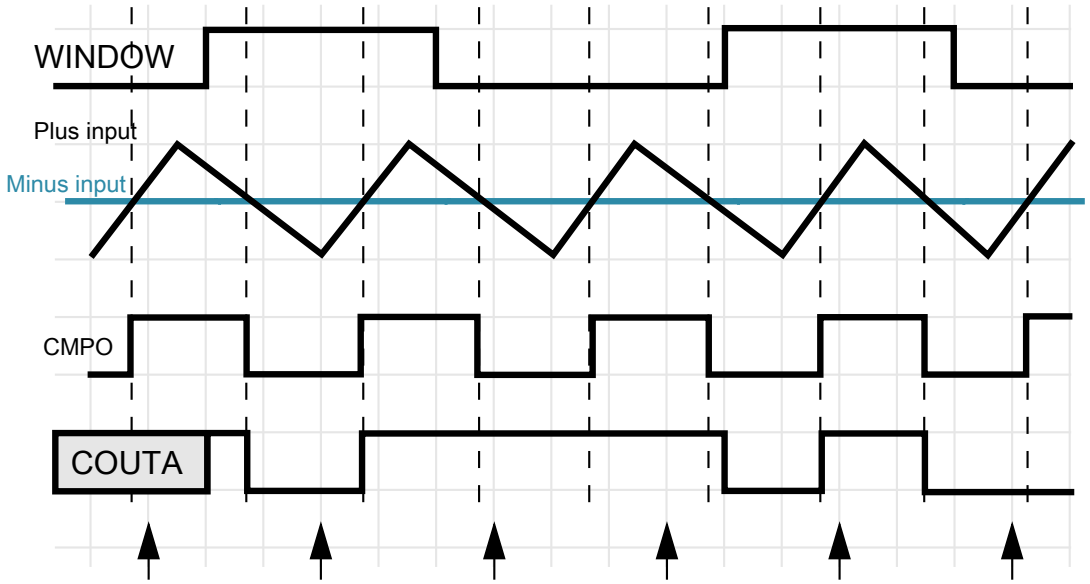


Figure 40-8. Windowed mode operation

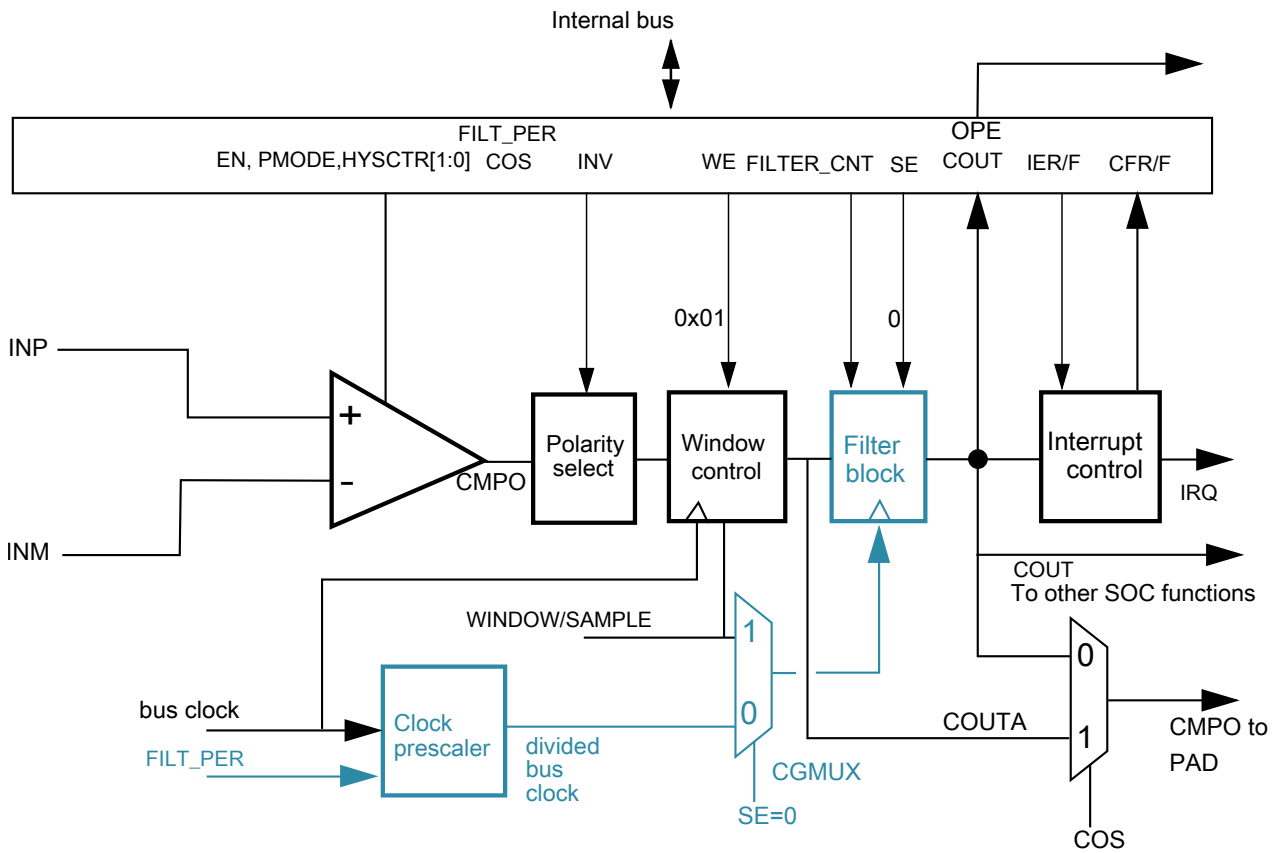


Figure 40-9. Windowed mode

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.

40.4.1.6 Windowed/Resampled mode (# 6)

The following figure uses the same input stimulus shown in Figure 40-8, 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.

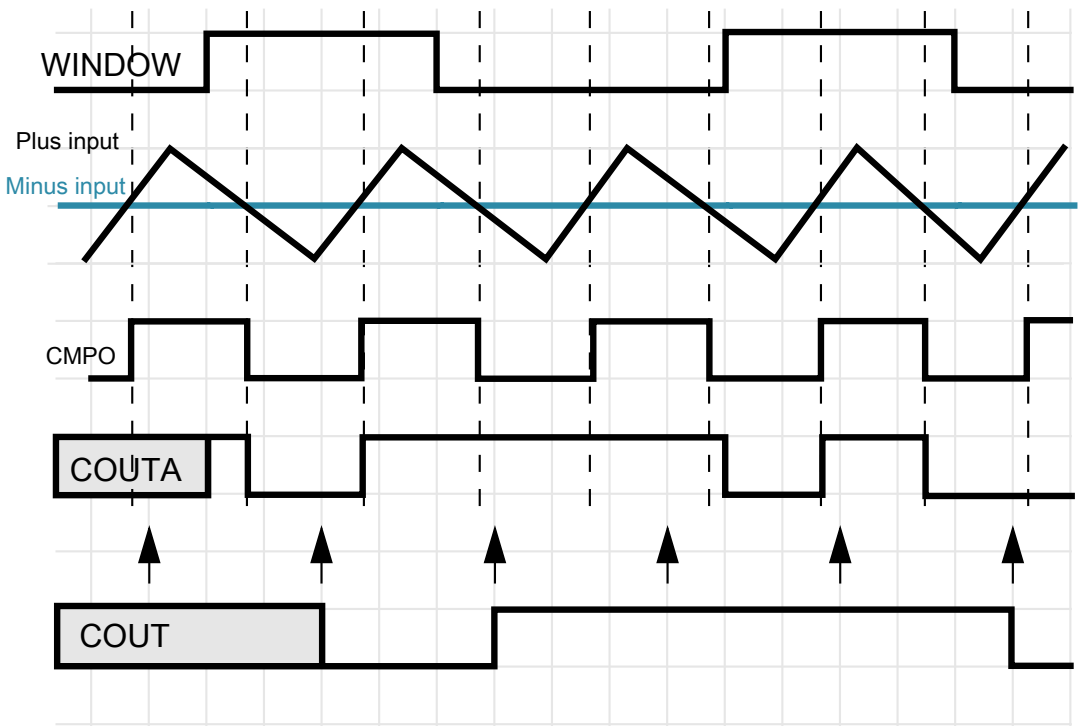


Figure 40-10. Windowed/resampled mode operation

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.

40.4.1.7 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.

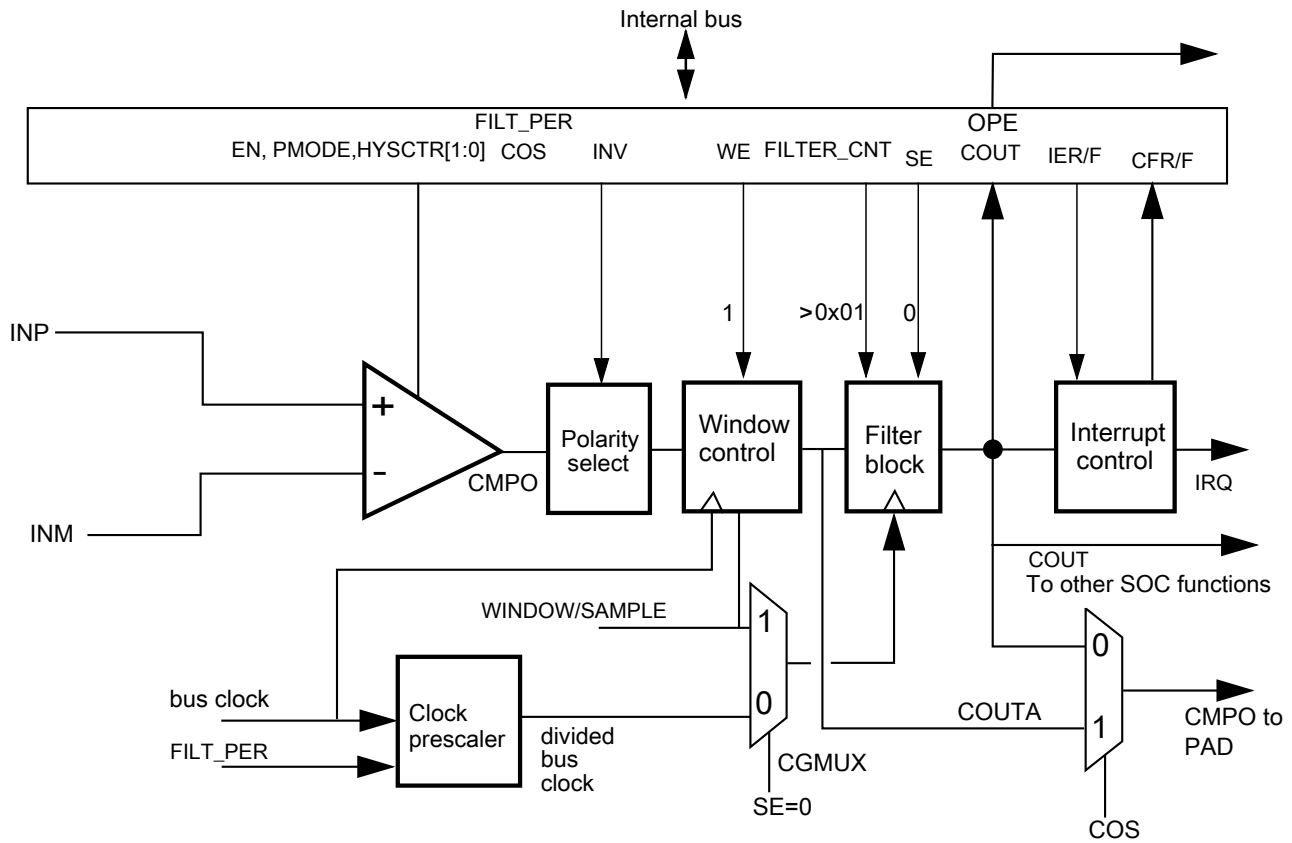


Figure 40-11. Windowed/Filtered mode

40.4.2 Power modes

40.4.2.1 Wait mode operation

During Wait and VLPW modes, the CMP, if enabled, continues to operate normally and a CMP interrupt can wake the MCU.

40.4.2.2 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.

40.4.2.3 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].

40.4.3 Startup and operation

A typical startup sequence is listed here.

- 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.

40.4.4 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.

40.4.4.1 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.

40.4.4.2 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 40-4. 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}
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}

Table continues on the next page...

Table 40-4. Comparator sample/filter maximum latencies (continued)

Mode #	CR1[EN]	CR1[WE]	CR1[SE]	CR0[FILTER_CNT]	FPR[FILT_PER]	Operation	Maximum latency[1]
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}) + $2T_{per}$
7	1	1	0	> 0x01	0x01 - 0xFF	Windowed / Filtered mode	T_{PD} + (CR0[FILTER_CNT] * FPR[FILT_PER] x T_{per}) + $2T_{per}$

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.

40.5 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

40.6 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.

40.7 CMP Asynchronous 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.

40.8 Digital-to-analog converter

The figure found here 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.

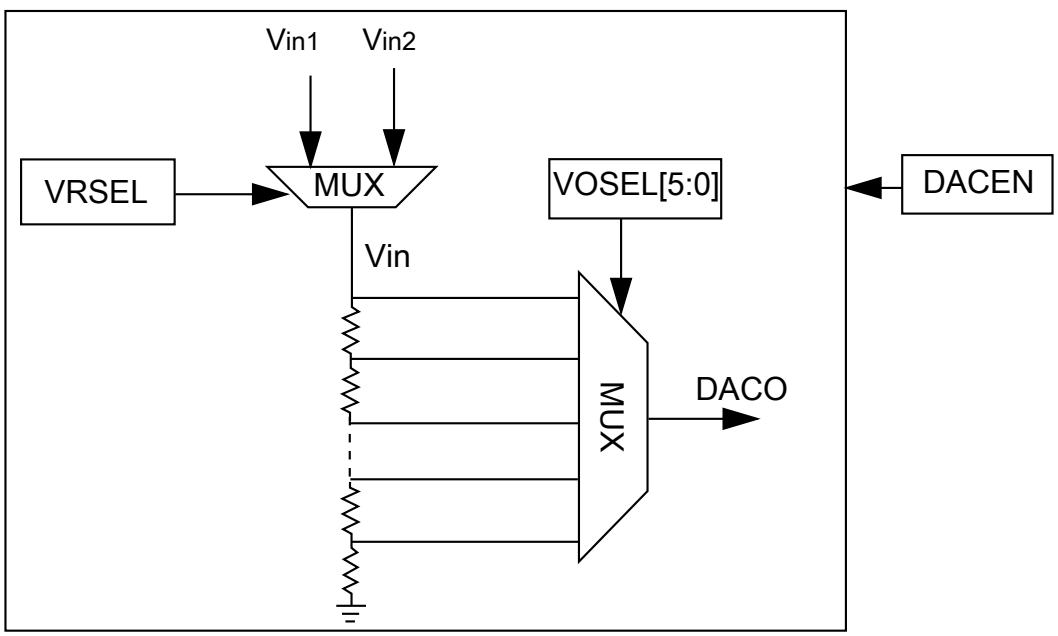


Figure 40-12. 6-bit DAC block diagram

40.9 DAC functional description

This section provides DAC functional description information.

40.9.1 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

40.10 DAC resets

This module has a single reset input, corresponding to the chip-wide peripheral reset.

40.11 DAC clocks

This module has a single clock input, the bus clock.

40.12 DAC interrupts

This module has no interrupts.

Chapter 41
Voltage Reference (VREFV1)

41.1 Chip-specific VREF information

41.1.1 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

On certain package of this device, the VREF module cannot be used due to the lack of VREF_OUT pin.

41.2 Introduction

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.

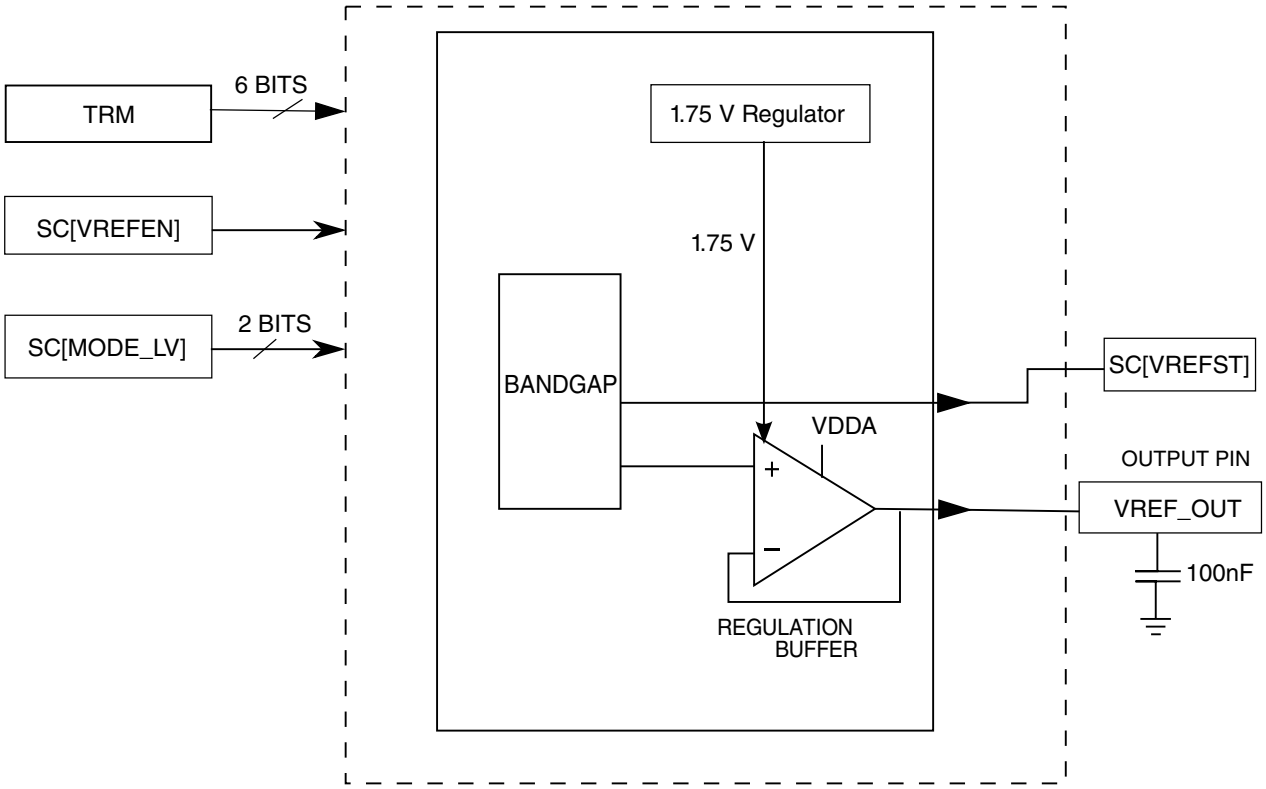


Figure 41-1. Voltage reference block diagram

41.2.1 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 details for a description of these options. The reference voltage signal 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.

41.2.2 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

41.2.3 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 and/or the chop oscillator in the very low power modes, the system reference voltage (also referred to as the bandgap voltage reference) must be enabled in these modes. Refer to the chip configuration details 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.

41.2.4 VREF Signal Descriptions

The following table shows the Voltage Reference signals properties.

Table 41-1. 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.

41.3 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	41.3.1/1206
4007_4001	VREF Status and Control Register (VREF_SC)	8	R/W	00h	41.3.2/1207

41.3.1 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	Reserved	CHOPEN	TRIM					
Write								
Reset	x*	0	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

VREF_TRM field descriptions

Field	Description
7 Reserved	This field is reserved. Upon reset this value is loaded with a factory trim value.
6 CHOPEN	Chop oscillator enable. When set, internal chopping operation is enabled and the internal analog offset will be minimized. 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. If the internal voltage regulator is being used (REGEN bit is set to 1), the chop oscillator must also be enabled. If the chop oscillator is to be used in very low power modes, the system (bandgap) voltage reference must also be enabled. See the chip-specific VREF information (also known as "chip configuration" details) for a description of how this can be achieved. 0 Chop oscillator is disabled. 1 Chop oscillator is enabled.
TRIM	Trim bits These bits change the resulting VREF by approximately ± 0.5 mV for each step. **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.

Table continues on the next page...

VREF_TRM field descriptions (continued)

Field	Description
	000000 Min .... 111111 Max

41.3.2 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 VREFEN	Internal Voltage Reference enable This bit is used to enable the bandgap reference within the Voltage Reference module. **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. 0 The module is disabled. 1 The module is enabled.
6 REGEN	Regulator enable 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 details for a description on how this can be achieved. This bit should be written to 1 to achieve the performance stated in the data sheet. **NOTE:** See section "Internal voltage regulator" for details on the required sequence to enable the internal regulator. 0 Internal 1.75 V regulator is disabled. 1 Internal 1.75 V regulator is enabled.
5 ICOMPEN	Second order curvature compensation enable This bit should be written to 1 to achieve the performance stated in the data sheet.

Table continues on the next page...

VREF_SC field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 VREFST	Internal Voltage Reference stable This bit indicates that the bandgap reference within the Voltage Reference module has completed its startup and stabilization. **NOTE:** This bit is valid only when the chop oscillator is not being used. 0 The module is disabled or not stable. 1 The module is stable.
MODE_LV	Buffer Mode selection These bits select the buffer modes for the Voltage Reference module. 00 Bandgap on only, for stabilization and startup 01 High power buffer mode enabled 10 Low-power buffer mode enabled 11 Reserved

41.4 Functional Description

The Voltage Reference is a bandgap buffer system. Unity gain amplifiers are used.

The VREF_OUT signal can be used by both internal and external peripherals in low and high power buffer mode. A 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 41-2. 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

41.4.1 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.

41.4.2 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.

41.4.2.1 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 when the chop oscillator is not enabled.

If the chop oscillator is being used, the internal bandgap reference voltage settles within the chop oscillator start up time, Tchop_osc_stup.

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.

41.4.2.2 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 when the chop oscillator is not enabled. If the chop oscillator is being used, you must wait the time specified by Tchop_osc_stup (chop oscillator start up time) to ensure the VREF output has stabilized.

In this mode, a 100 nF capacitor is required to connect between the VREF_OUT pin and VSSA.

41.4.2.3 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 when the chop oscillator is not enabled. If the chop oscillator is being used, you must wait the time specified by Tchop_osc_stup (chop oscillator start up time) to ensure the VREF output has stabilized.

In this mode, a 100 nF capacitor is required to connect between the VREF_OUT pin and VSSA.

41.4.2.4 SC[MODE_LV] = 11

Reserved

41.4.3 Internal voltage regulator

The VREF module contains an internal voltage regulator that can be enabled to provide additional supply noise rejection. It is recommended that when possible, this regulator be enabled to provide the optimum VREF performance.

If the internal voltage regulator is being used, the chop oscillator must also be enabled. A specific sequence must be followed when enabling the internal regulator as follows:

1. Enable the chop oscillator (VREF_TRM[CHOPEN] = 1)
2. Configure the VREF_SC register to the desired settings with the internal regulator disabled, VREF_SC[REGEN] = 0
3. Wait > 300ns

4. Enable the internal regulator by setting VREF_SC[REGEN] to 1

41.5 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 when the chop oscillator is not enabled. When the chop oscillator is enabled, the settling time of the internal bandgap reference is defined by Tchop_osc_stup (chop oscillator start up time). You must wait this time (Tchop_osc_stup) after the internal bandgap has been enabled to ensure the VREF internal reference voltage has stabilized.

When the Voltage Reference is already enabled and stabilized, changing 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 must be written to 1 to achieve the performance stated in the device data sheet.

NOTE

See section "Internal voltage regulator" for details on the required sequence to enable the internal regulator.

Chapter 42
12-bit Digital-to-Analog Converter (DAC)

42.1 Chip-specific DAC information

42.1.1 12-bit DAC Overview

This device contains 1 instance of a 12-bit digital-to-analog converter (DAC) with programmable reference generator output. The DAC includes a FIFO for DMA support. This FIFO depth is 16.

The DAC hardware trigger is controlled by PIT Channel 0 output.

42.1.2 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.

42.1.3 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.

42.2 Introduction

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.

42.3 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

42.4 Block diagram

The block diagram of the DAC module is as follows:

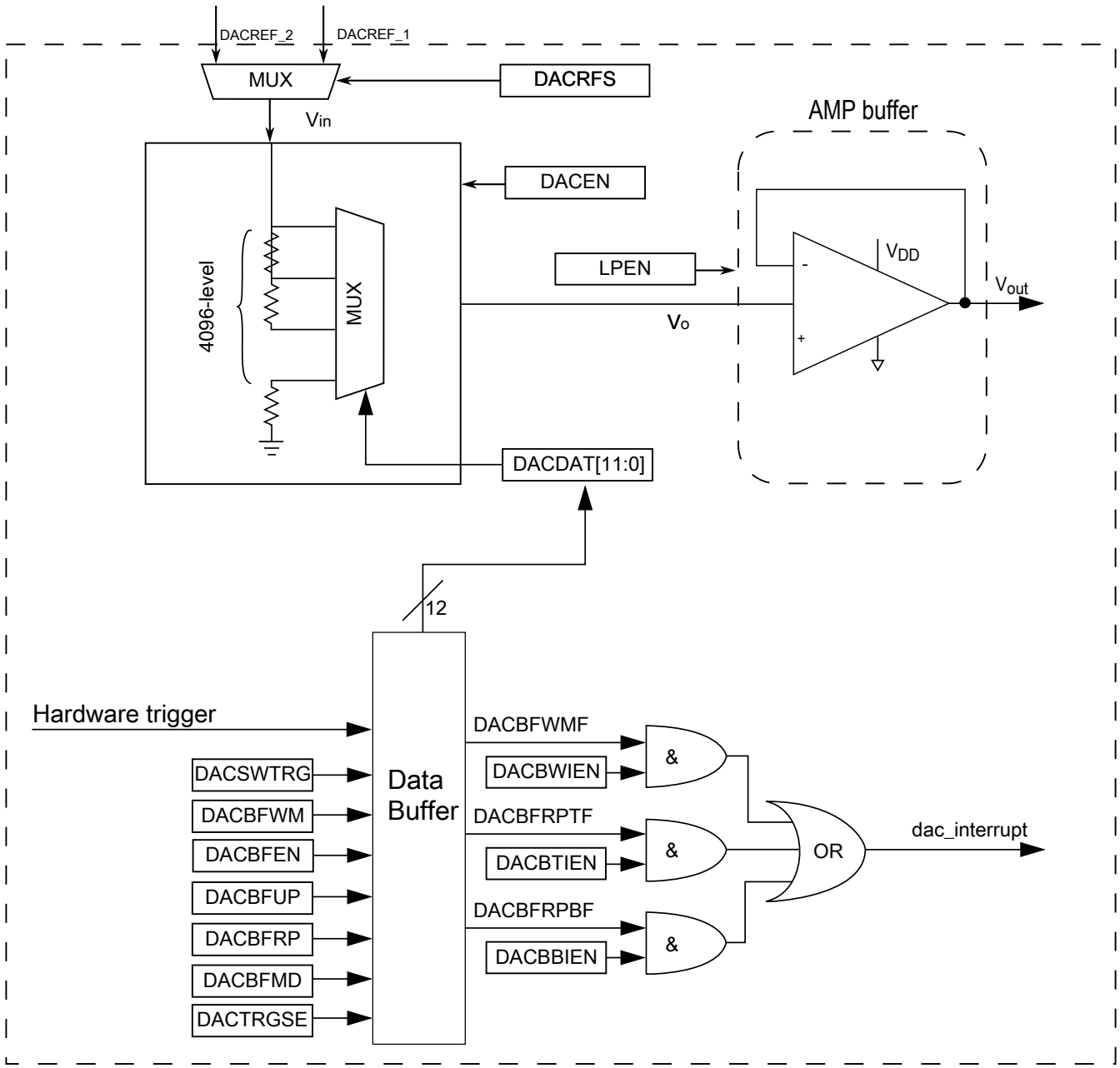


Figure 42-1. DAC block diagram

42.5 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
4003_F000	DAC Data Low Register (DAC0_DAT0L)	8	R/W	00h	42.5.1/1217
4003_F001	DAC Data High Register (DAC0_DAT0H)	8	R/W	00h	42.5.2/1217
4003_F002	DAC Data Low Register (DAC0_DAT1L)	8	R/W	00h	42.5.1/1217
4003_F003	DAC Data High Register (DAC0_DAT1H)	8	R/W	00h	42.5.2/1217
4003_F004	DAC Data Low Register (DAC0_DAT2L)	8	R/W	00h	42.5.1/1217
4003_F005	DAC Data High Register (DAC0_DAT2H)	8	R/W	00h	42.5.2/1217
4003_F006	DAC Data Low Register (DAC0_DAT3L)	8	R/W	00h	42.5.1/1217
4003_F007	DAC Data High Register (DAC0_DAT3H)	8	R/W	00h	42.5.2/1217
4003_F008	DAC Data Low Register (DAC0_DAT4L)	8	R/W	00h	42.5.1/1217
4003_F009	DAC Data High Register (DAC0_DAT4H)	8	R/W	00h	42.5.2/1217
4003_F00A	DAC Data Low Register (DAC0_DAT5L)	8	R/W	00h	42.5.1/1217
4003_F00B	DAC Data High Register (DAC0_DAT5H)	8	R/W	00h	42.5.2/1217
4003_F00C	DAC Data Low Register (DAC0_DAT6L)	8	R/W	00h	42.5.1/1217
4003_F00D	DAC Data High Register (DAC0_DAT6H)	8	R/W	00h	42.5.2/1217
4003_F00E	DAC Data Low Register (DAC0_DAT7L)	8	R/W	00h	42.5.1/1217
4003_F00F	DAC Data High Register (DAC0_DAT7H)	8	R/W	00h	42.5.2/1217
4003_F010	DAC Data Low Register (DAC0_DAT8L)	8	R/W	00h	42.5.1/1217
4003_F011	DAC Data High Register (DAC0_DAT8H)	8	R/W	00h	42.5.2/1217
4003_F012	DAC Data Low Register (DAC0_DAT9L)	8	R/W	00h	42.5.1/1217
4003_F013	DAC Data High Register (DAC0_DAT9H)	8	R/W	00h	42.5.2/1217
4003_F014	DAC Data Low Register (DAC0_DAT10L)	8	R/W	00h	42.5.1/1217
4003_F015	DAC Data High Register (DAC0_DAT10H)	8	R/W	00h	42.5.2/1217
4003_F016	DAC Data Low Register (DAC0_DAT11L)	8	R/W	00h	42.5.1/1217
4003_F017	DAC Data High Register (DAC0_DAT11H)	8	R/W	00h	42.5.2/1217
4003_F018	DAC Data Low Register (DAC0_DAT12L)	8	R/W	00h	42.5.1/1217
4003_F019	DAC Data High Register (DAC0_DAT12H)	8	R/W	00h	42.5.2/1217
4003_F01A	DAC Data Low Register (DAC0_DAT13L)	8	R/W	00h	42.5.1/1217
4003_F01B	DAC Data High Register (DAC0_DAT13H)	8	R/W	00h	42.5.2/1217
4003_F01C	DAC Data Low Register (DAC0_DAT14L)	8	R/W	00h	42.5.1/1217
4003_F01D	DAC Data High Register (DAC0_DAT14H)	8	R/W	00h	42.5.2/1217
4003_F01E	DAC Data Low Register (DAC0_DAT15L)	8	R/W	00h	42.5.1/1217
4003_F01F	DAC Data High Register (DAC0_DAT15H)	8	R/W	00h	42.5.2/1217
4003_F020	DAC Status Register (DAC0_SR)	8	R/W	02h	42.5.3/1218
4003_F021	DAC Control Register (DAC0_C0)	8	R/W	00h	42.5.4/1219
4003_F022	DAC Control Register 1 (DAC0_C1)	8	R/W	00h	42.5.5/1220
4003_F023	DAC Control Register 2 (DAC0_C2)	8	R/W	0Fh	42.5.6/1221

42.5.1 DAC Data Low Register (DAC*x*_DAT*n*L)

Address: 4003_F000h base + 0h offset + (2d × i), where i=0d to 15d

Bit	7	6	5	4	3	2	1	0
Read Write	DATA0							
Reset	0	0	0	0	0	0	0	0

DAC*x*_DAT*n*L field descriptions

Field	Description
DATA0	DATA0 When the DAC buffer is not enabled, DATA[11:0] controls the output voltage based on the following formula: $V_{out} = V_{in} * (1 + DACDAT0[11:0])/4096$ When the DAC buffer is enabled, DATA is mapped to the 16-word buffer.

42.5.2 DAC Data High Register (DAC*x*_DAT*n*H)

Address: 4003_F000h base + 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

DAC*x*_DAT*n*H field descriptions

Field	Description
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
DATA1	DATA1 When the DAC Buffer is not enabled, DATA[11:0] controls the output voltage based on the following formula. $V_{out} = V_{in} * (1 + DACDAT0[11:0])/4096$ When the DAC buffer is enabled, DATA[11:0] is mapped to the 16-word buffer.

42.5.3 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.

Address: 4003_F000h base + 20h offset = 4003_F020h

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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 DACBFWMF	DAC Buffer Watermark Flag This bit is set if the remaining FIFO data is less than the watermark setting. It is cleared automatically by writing data into FIFO by DMA or CPU. Write to this bit is ignored in FIFO mode. 0 The DAC buffer read pointer has not reached the watermark level. 1 The DAC buffer read pointer has reached the watermark level.
1 DACBFRPTF	DAC Buffer Read Pointer Top Position Flag In FIFO mode, it is FIFO nearly empty flag. It is set when only one data remains in FIFO. Any DAC trigger does not increase the Read Pointer if this bit is set to avoid any possible glitch or abrupt change at DAC output. It is cleared automatically if FIFO is not empty. 0 The DAC buffer read pointer is not zero. 1 The DAC buffer read pointer is zero.
0 DACBFRPBF	DAC Buffer Read Pointer Bottom Position Flag In FIFO mode, it is FIFO FULL status bit. It means FIFO read pointer equals Write Pointer because of Write Pointer increase. If this bit is set, any write to FIFO from either DMA or CPU is ignored by DAC. It is cleared if there is any DAC trigger making the DAC read pointer increase. Write to this bit is ignored in FIFO mode. 0 The DAC buffer read pointer is not equal to C2[DACBFUP]. 1 The DAC buffer read pointer is equal to C2[DACBFUP].

42.5.4 DAC Control Register (DACx_C0)

Address: 4003_F000h base + 21h offset = 4003_F021h

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 DACEN	DAC Enable Starts the Programmable Reference Generator operation. 0 The DAC system is disabled. 1 The DAC system is enabled.
6 DACRFS	DAC Reference Select 0 The DAC selects DACREF_1 as the reference voltage. 1 The DAC selects DACREF_2 as the reference voltage.
5 DACTRGSEL	DAC Trigger Select 0 The DAC hardware trigger is selected. 1 The DAC software trigger is selected.
4 DACSWTRG	DAC Software Trigger 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. 0 The DAC soft trigger is not valid. 1 The DAC soft trigger is valid.
3 LPEN	DAC Low Power Control **NOTE:** See the 12-bit DAC electrical characteristics of the device data sheet for details on the impact of the modes below. 0 High-Power mode 1 Low-Power mode
2 DACBWIEN	DAC Buffer Watermark Interrupt Enable 0 The DAC buffer watermark interrupt is disabled. 1 The DAC buffer watermark interrupt is enabled.
1 DACBTIEN	DAC Buffer Read Pointer Top Flag Interrupt Enable 0 The DAC buffer read pointer top flag interrupt is disabled. 1 The DAC buffer read pointer top flag interrupt is enabled.
0 DACBBIEN	DAC Buffer Read Pointer Bottom Flag Interrupt Enable

Table continues on the next page...

DACx_C0 field descriptions (continued)

Field	Description
	0 The DAC buffer read pointer bottom flag interrupt is disabled. 1 The DAC buffer read pointer bottom flag interrupt is enabled.

42.5.5 DAC Control Register 1 (DACx_C1)

Address: 4003_F000h base + 22h offset = 4003_F022h

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 DMAEN	DMA Enable Select 0 DMA is disabled. 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–3 DACBFWM	DAC Buffer Watermark Select In normal mode it controls when SR[DACBFWMF] is 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. In FIFO mode, it is FIFO watermark select field. 00 In normal mode, 1 word . In FIFO mode, 2 or less than 2 data remaining in FIFO will set watermark status bit. 01 In normal mode, 2 words . In FIFO mode, Max/4 or less than Max/4 data remaining in FIFO will set watermark status bit. 10 In normal mode, 3 words . In FIFO mode, Max/2 or less than Max/2 data remaining in FIFO will set watermark status bit. 11 In normal mode, 4 words . In FIFO mode, Max-2 or less than Max-2 data remaining in FIFO will set watermark status bit.
2–1 DACBFMD	DAC Buffer Work Mode Select 00 Normal mode 01 Swing mode 10 One-Time Scan mode 11 FIFO mode
0 DACBFEN	DAC Buffer Enable 0 Buffer read pointer is disabled. The converted data is always the first word of the buffer. 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.

42.5.6 DAC Control Register 2 (DAC*x*_C2)

Address: 4003_F000h base + 23h offset = 4003_F023h

Bit	7	6	5	4	3	2	1	0
Read Write	DACBFRP				DACBFUP			
Reset	0	0	0	0	1	1	1	1

DAC*x*_C2 field descriptions

Field	Description
7–4 DACBFRP	DAC Buffer Read Pointer In normal mode it keeps the current value of the buffer read pointer. FIFO mode, it is the FIFO read pointer. It is writable in FIFO mode. User can configure it to same address to reset FIFO as empty.
DACBFUP	DAC Buffer Upper Limit In normal mode it selects the upper limit of the DAC buffer. The buffer read pointer cannot exceed it. In FIFO mode it is the FIFO write pointer. User cannot set Buffer Up limit in FIFO mode. In Normal mode its reset value is MAX. When IP is configured to FIFO mode, this register becomes Write_Pointer, and its value is initially set to equal READ_POINTER automatically, and the FIFO status is empty. It is writable and user can configure it to the same address to reset FIFO as empty.

42.6 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 chip-specific DAC information to determine the source options 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 $V_{in}/4096$.

42.6.1 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 the 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 a hardware or software trigger event occurs.

The data buffer can be configured to operate in Normal mode, Swing mode, One-Time Scan mode or FIFO 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].

42.6.1.1 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].

42.6.1.2 Modes of DAC data buffer operation

The following table describes the different modes of data buffer operation for the DAC module.

Table 42-1. 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. **NOTE:** If the software set the read pointer to the upper limit, the read pointer will not advance in this mode.
FIFO Mode	In FIFO mode, the buffer is organized as a FIFO. For a valid write to any DACDATx, the data is put into the FIFO, and the write pointer is automatically incremented. The module is connected internally to a 32bit interface. For any 16bit or 8bit FIFO access, address bit[1] needs to be 0; otherwise, the write is ignored. For any 32bit FIFO access, the Write_Pointer needs to be an EVEN number; otherwise, the write is ignored.

Table 42-1. Modes of DAC data buffer operation

Modes	Description
	NOTE: A successful 32bit FIFO write will increase the write pointer by 2. Any write will cause the FIFO over-flow will be ignored, the cases includes: 1.FIFO is full, the write will be ignored. 2.FIFO is nearly full (FIFO_SIZE-1), 32bit write will be ignored. **NOTE:** For 8bit write, address bit[0] determine which byte lane will be written to the FIFO according to little endian alignment. Only both byte lanes are written will the write pointer increase. User need to make sure 8bit access happened in pair and both upper & lower bytes are written. There is no requirement on which byte write first. In FIFO mode, there is no change to read access of DACDATx (from normal mode), read to DACDATx will return the DATA addressed by the access address to the data buffer, and both write pointer and read pointer in FIFO mode will NOT be changed by read access. FIFO write can be happened when DAC is not enabled for 1st data conversion enable. But FIFO mode need to work at buffer Enabled at DACC1[DACBFEN]. In FIFO mode, the DATA BUF will be organized as FIFO.

42.6.2 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.

42.6.3 Resets

During reset, the DAC is configured in the default mode and is disabled.

42.6.4 Low-Power mode operation

The following table shows the wait mode and the stop mode operation of the DAC module.

Table 42-2. 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. 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 43
Timer/PWM Module (TPM)

43.1 Chip-specific TPM information

43.1.1 TPM Instantiation Information

This device contains 3 low power TPM module(s) (TPM). All TPM modules in the device are configured with basic TPM functionality and all can be functional in Stop/VLPS mode. The clock source is either external or internal in Stop/VLPS mode.

The following table shows how these modules are configured.

Table 43-1. TPM configuration

TPM instance	Number of channels	Features/usage
TPM0	6	Basic TPM,functional in Stop/VLPS mode
TPM1	2	Basic TPM,functional in Stop/VLPS mode
TPM2	2	Basic TPM,functional in Stop/VLPS mode

43.1.2 Clock Options

The TPM blocks are clocked from a single TPM clock that can be selected from several clock sources. Refer to the clock distribution for details.

Each TPM also supports an external clock mode (TPM_SC[CMOD]=1x) in which the counter increments after a synchronized (to the selected TPM clock source) rising edge detect of an external clock input. The available external clock (either TPM_CLKIN0 or TPM_CLKIN1) is selected by SIM_SOPT9[TPMxCLKSEL] control register. To guarantee valid operation the selected external clock must be less than half the frequency of the selected TPM clock source.

43.1.3 Trigger Options

Each TPM has a selectable trigger input source controlled by the TPMx_CONF[TRGSEL] field to use for starting the counter and/or reloading the counter. The options available are shown in the following table.

Table 43-2. TPM trigger options

TPMx_CONF[TRGSEL]	Selected source
0000	External trigger pin input (EXTRG_IN)
0001	CMP0 output
0010	CMP1 output
0011	Reserved
0100	PIT0 trigger 0
0101	PIT0 trigger 1
0110	PIT0 trigger 2
0111	PIT0 trigger 3
1000	TPM0 overflow
1001	TPM1 overflow
1010	TPM2 overflow
1011	Reserved
1100	RTC alarm
1101	RTC seconds
1110	LPTMR0
1111	Reserved

43.1.4 Global Timebase

Each TPM has a global timebase feature controlled by the TPMx_CONF[GTBEEN] bit. TPM1 is configured as the global time when this option is enabled.

43.1.5 TPM Interrupts

The TPM has multiple sources of interrupt. However, these sources are OR'd together to generate a single interrupt request to the interrupt controller. When an TPM interrupt occurs, read the TPM status registers to determine the exact interrupt source.

43.2 Introduction

The TPM (Timer/PWM Module) is a 2- to 8-channel timer which supports input capture, output compare, and the generation of PWM signals to control electric motor and power management applications.

The counter, compare and capture registers are clocked by an asynchronous clock that can remain enabled in low power modes. An example of using the TPM with the asynchronous DMA is described in AN4631:Using the Asynchronous DMA features of the Kinetis L Series .

43.2.1 TPM Philosophy

The TPM is built upon a very simple timer (HCS08 Timer PWM Module – TPM) used for many years on NXP's 8-bit microcontrollers. The TPM extends the functionality to support operation in low power modes by clocking the counter, compare and capture registers from an asynchronous clock that can remain functional in low power modes.

43.2.2 Features

The TPM features include:

- TPM clock mode is selectable
  - Can increment on every edge of the asynchronous counter clock
  - Can increment on rising edge of an external clock input synchronized to the asynchronous counter clock
- Prescaler divide-by 1, 2, 4, 8, 16, 32, 64, or 128
- TPM includes a 16-bit counter
  - It can be a free-running counter or modulo counter
  - The counting can be up or up-down
- Includes 6 channels that can be configured for input capture, output compare, edge-aligned PWM mode, or center-aligned PWM mode
  - In input capture mode the capture can occur on rising edges, falling edges or both edges

    - In output compare mode the output signal can be set, cleared, pulsed, or toggled on match
    - All channels can be configured for edge-aligned PWM mode or center-aligned PWM mode
- Support the generation of an interrupt and/or DMA request per channel
- Support the generation of an interrupt and/or DMA request when the counter overflows
- Support selectable trigger input to optionally reset or cause the counter to start incrementing.
    - The counter can also optionally stop incrementing on counter overflow
- Support the generation of hardware triggers when the counter overflows and per channel

43.2.3 Modes of operation

During debug mode, the TPM can can be configured to temporarily pause all counting until the core returns to normal user operating mode or to operate normally. When the counter is paused, trigger inputs and input capture events are ignored.

During doze mode, the TPM can be configured to operate normally or to pause all counting for the duration of doze mode. When the counter is paused, trigger inputs and input capture events are ignored.

During stop mode, the TPM counter clock can remain functional and the TPM can generate an asynchronous interrupt to exit the MCU from stop mode.

43.2.4 Block diagram

The TPM uses one input/output (I/O) pin per channel, CHn (TPM channel (n)) where n is the channel number.

The following figure shows the TPM structure. The central component of the TPM is the 16-bit counter with programmable final value and its counting can be up or up-down.

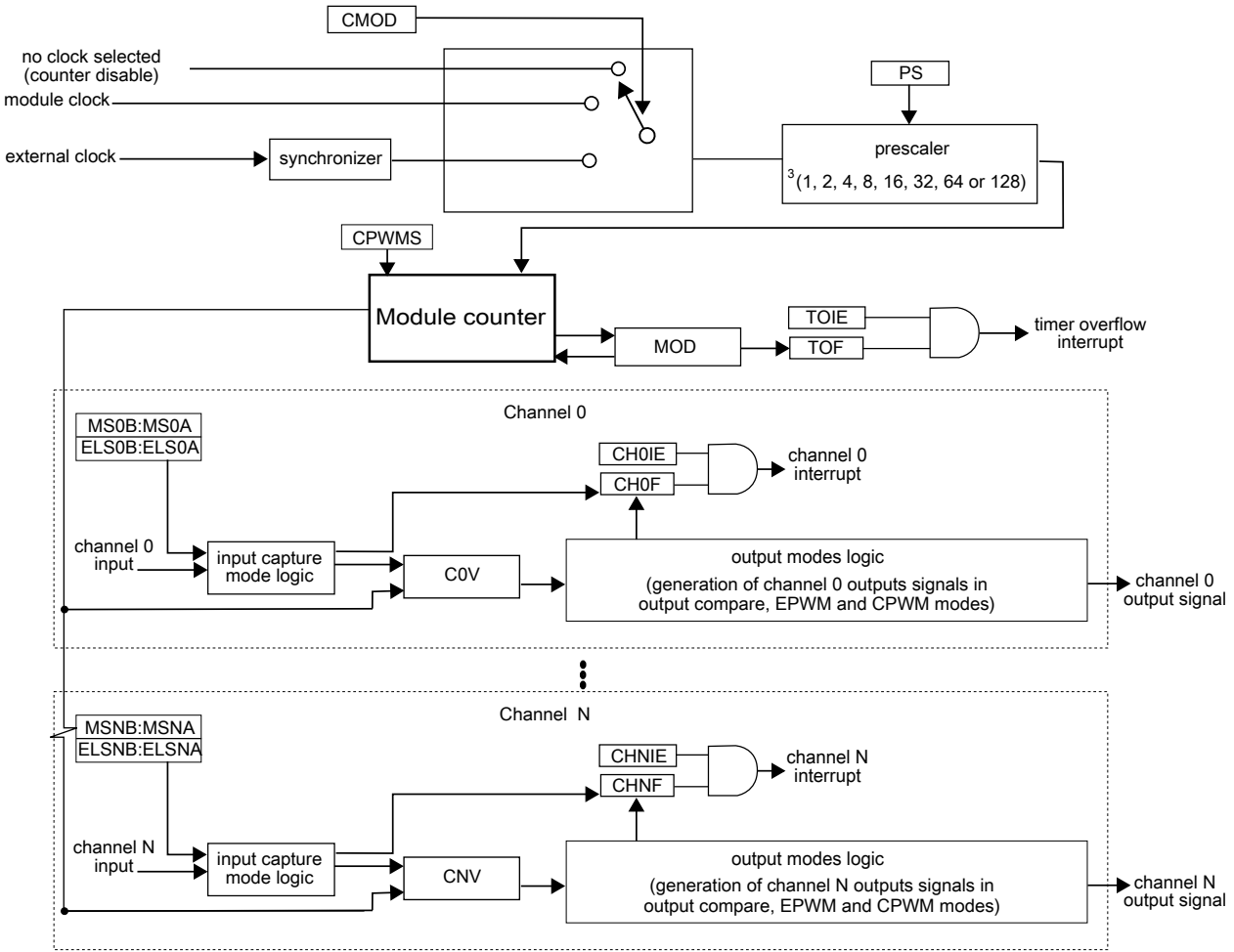


Figure 43-1. TPM block diagram

43.3 TPM Signal Descriptions

Table 43-3 shows the user-accessible signals for the TPM.

Table 43-3. TPM signal descriptions

Signal	Description	I/O
TPM_EXTCLK	External clock. TPM external clock can be selected to increment the TPM counter on every rising edge synchronized to the counter clock.	I
TPM_CHn	TPM channel (n = 5 to 0). A TPM channel pin is configured as output when configured in an output compare or PWM mode and the TPM counter is enabled, otherwise the TPM channel pin is an input.	I/O

43.3.1 TPM_EXTCLK — TPM External Clock

The rising edge of the external input signal is used to increment the TPM counter if selected by CMOD[1:0] bits in the SC register. This input signal must be less than half of the TPM counter clock frequency. The TPM counter prescaler selection and settings are also used when an external input is selected.

43.3.2 TPM_CHn — TPM Channel (n) I/O Pin

Each TPM 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.

43.4 Memory Map and Register Definition

This section provides a detailed description of all TPM registers.

Attempting to access a reserved register location in the TPM memory map will generate a bus error.

TPM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_8000	Status and Control (TPM0_SC)	32	R/W	0000_0000h	43.4.1/1232
4003_8004	Counter (TPM0_CNT)	32	R/W	0000_0000h	43.4.2/1234
4003_8008	Modulo (TPM0_MOD)	32	R/W	0000_FFFFh	43.4.3/1234
4003_800C	Channel (n) Status and Control (TPM0_C0SC)	32	R/W	0000_0000h	43.4.4/1235
4003_8010	Channel (n) Value (TPM0_C0V)	32	R/W	0000_0000h	43.4.5/1237
4003_8014	Channel (n) Status and Control (TPM0_C1SC)	32	R/W	0000_0000h	43.4.4/1235
4003_8018	Channel (n) Value (TPM0_C1V)	32	R/W	0000_0000h	43.4.5/1237
4003_801C	Channel (n) Status and Control (TPM0_C2SC)	32	R/W	0000_0000h	43.4.4/1235
4003_8020	Channel (n) Value (TPM0_C2V)	32	R/W	0000_0000h	43.4.5/1237
4003_8024	Channel (n) Status and Control (TPM0_C3SC)	32	R/W	0000_0000h	43.4.4/1235
4003_8028	Channel (n) Value (TPM0_C3V)	32	R/W	0000_0000h	43.4.5/1237
4003_802C	Channel (n) Status and Control (TPM0_C4SC)	32	R/W	0000_0000h	43.4.4/1235
4003_8030	Channel (n) Value (TPM0_C4V)	32	R/W	0000_0000h	43.4.5/1237
4003_8034	Channel (n) Status and Control (TPM0_C5SC)	32	R/W	0000_0000h	43.4.4/1235
4003_8038	Channel (n) Value (TPM0_C5V)	32	R/W	0000_0000h	43.4.5/1237
4003_8050	Capture and Compare Status (TPM0_STATUS)	32	R/W	0000_0000h	43.4.6/1238

Table continues on the next page...

TPM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_8064	Combine Channel Register (TPM0_COMBINE)	32	R/W	0000_0000h	43.4.7/1240
4003_8070	Channel Polarity (TPM0_POL)	32	R/W	0000_0000h	43.4.8/1241
4003_8078	Filter Control (TPM0_FILTER)	32	R/W	0000_0000h	43.4.9/1242
4003_8084	Configuration (TPM0_CONF)	32	R/W	0000_0000h	43.4.10/ 1243
4003_9000	Status and Control (TPM1_SC)	32	R/W	0000_0000h	43.4.1/1232
4003_9004	Counter (TPM1_CNT)	32	R/W	0000_0000h	43.4.2/1234
4003_9008	Modulo (TPM1_MOD)	32	R/W	0000_FFFFh	43.4.3/1234
4003_900C	Channel (n) Status and Control (TPM1_C0SC)	32	R/W	0000_0000h	43.4.4/1235
4003_9010	Channel (n) Value (TPM1_C0V)	32	R/W	0000_0000h	43.4.5/1237
4003_9014	Channel (n) Status and Control (TPM1_C1SC)	32	R/W	0000_0000h	43.4.4/1235
4003_9018	Channel (n) Value (TPM1_C1V)	32	R/W	0000_0000h	43.4.5/1237
4003_901C	Channel (n) Status and Control (TPM1_C2SC)	32	R/W	0000_0000h	43.4.4/1235
4003_9020	Channel (n) Value (TPM1_C2V)	32	R/W	0000_0000h	43.4.5/1237
4003_9024	Channel (n) Status and Control (TPM1_C3SC)	32	R/W	0000_0000h	43.4.4/1235
4003_9028	Channel (n) Value (TPM1_C3V)	32	R/W	0000_0000h	43.4.5/1237
4003_902C	Channel (n) Status and Control (TPM1_C4SC)	32	R/W	0000_0000h	43.4.4/1235
4003_9030	Channel (n) Value (TPM1_C4V)	32	R/W	0000_0000h	43.4.5/1237
4003_9034	Channel (n) Status and Control (TPM1_C5SC)	32	R/W	0000_0000h	43.4.4/1235
4003_9038	Channel (n) Value (TPM1_C5V)	32	R/W	0000_0000h	43.4.5/1237
4003_9050	Capture and Compare Status (TPM1_STATUS)	32	R/W	0000_0000h	43.4.6/1238
4003_9064	Combine Channel Register (TPM1_COMBINE)	32	R/W	0000_0000h	43.4.7/1240
4003_9070	Channel Polarity (TPM1_POL)	32	R/W	0000_0000h	43.4.8/1241
4003_9078	Filter Control (TPM1_FILTER)	32	R/W	0000_0000h	43.4.9/1242
4003_9084	Configuration (TPM1_CONF)	32	R/W	0000_0000h	43.4.10/ 1243
4003_A000	Status and Control (TPM2_SC)	32	R/W	0000_0000h	43.4.1/1232
4003_A004	Counter (TPM2_CNT)	32	R/W	0000_0000h	43.4.2/1234
4003_A008	Modulo (TPM2_MOD)	32	R/W	0000_FFFFh	43.4.3/1234
4003_A00C	Channel (n) Status and Control (TPM2_C0SC)	32	R/W	0000_0000h	43.4.4/1235
4003_A010	Channel (n) Value (TPM2_C0V)	32	R/W	0000_0000h	43.4.5/1237
4003_A014	Channel (n) Status and Control (TPM2_C1SC)	32	R/W	0000_0000h	43.4.4/1235
4003_A018	Channel (n) Value (TPM2_C1V)	32	R/W	0000_0000h	43.4.5/1237
4003_A01C	Channel (n) Status and Control (TPM2_C2SC)	32	R/W	0000_0000h	43.4.4/1235
4003_A020	Channel (n) Value (TPM2_C2V)	32	R/W	0000_0000h	43.4.5/1237
4003_A024	Channel (n) Status and Control (TPM2_C3SC)	32	R/W	0000_0000h	43.4.4/1235
4003_A028	Channel (n) Value (TPM2_C3V)	32	R/W	0000_0000h	43.4.5/1237
4003_A02C	Channel (n) Status and Control (TPM2_C4SC)	32	R/W	0000_0000h	43.4.4/1235

Table continues on the next page...

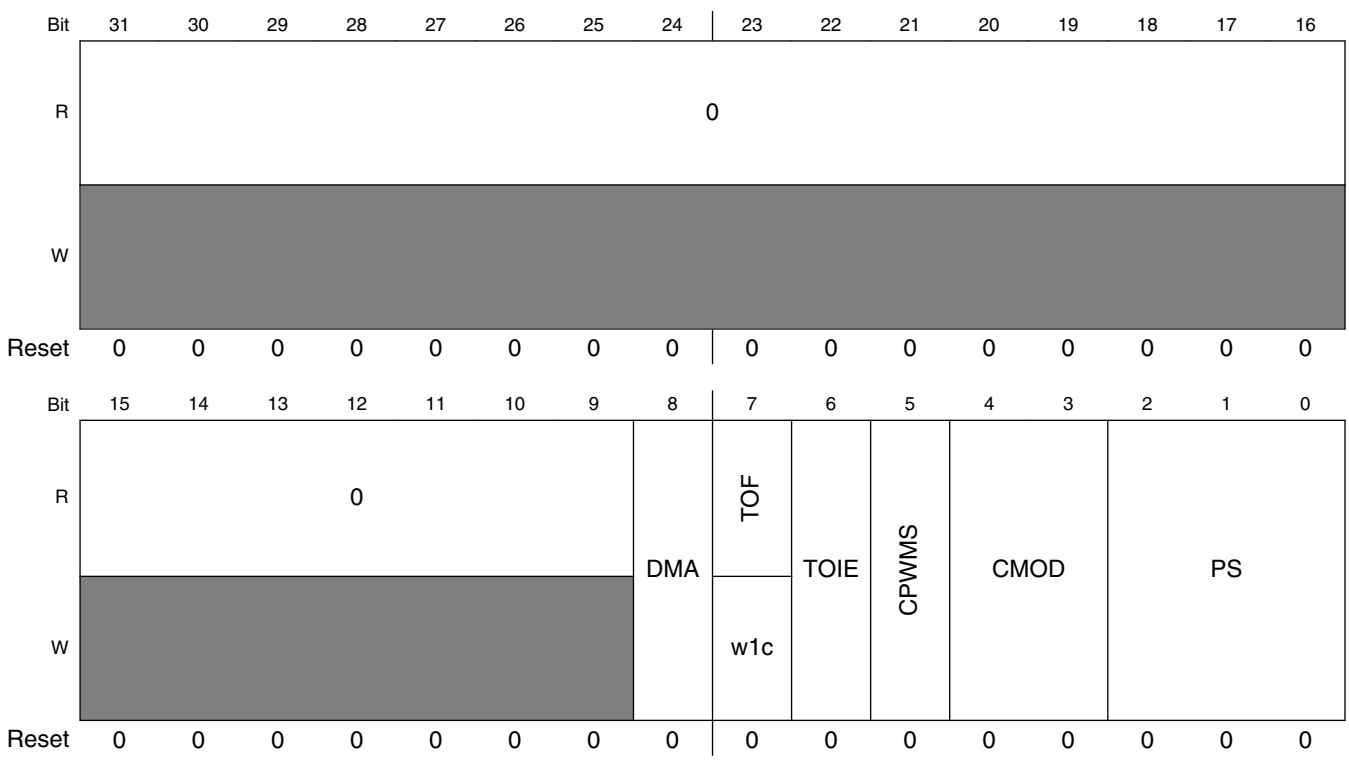
TPM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_A030	Channel (n) Value (TPM2_C4V)	32	R/W	0000_0000h	43.4.5/1237
4003_A034	Channel (n) Status and Control (TPM2_C5SC)	32	R/W	0000_0000h	43.4.4/1235
4003_A038	Channel (n) Value (TPM2_C5V)	32	R/W	0000_0000h	43.4.5/1237
4003_A050	Capture and Compare Status (TPM2_STATUS)	32	R/W	0000_0000h	43.4.6/1238
4003_A064	Combine Channel Register (TPM2_COMBINE)	32	R/W	0000_0000h	43.4.7/1240
4003_A070	Channel Polarity (TPM2_POL)	32	R/W	0000_0000h	43.4.8/1241
4003_A078	Filter Control (TPM2_FILTER)	32	R/W	0000_0000h	43.4.9/1242
4003_A084	Configuration (TPM2_CONF)	32	R/W	0000_0000h	43.4.10/ 1243

43.4.1 Status and Control (TPMx_SC)

SC contains the overflow status flag and control bits used to configure the interrupt enable, module configuration and prescaler factor. These controls relate to all channels within this module.

Address: Base address + 0h offset



TPMx_SC field descriptions

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 DMA	DMA Enable Enables DMA transfers for the overflow flag. 0 Disables DMA transfers. 1 Enables DMA transfers.
7 TOF	Timer Overflow Flag Set by hardware when the TPM counter equals the value in the MOD register and increments. Writing a 1 to TOF clears it. Writing a 0 to TOF has no effect. If another TPM overflow occurs between the flag setting and the flag clearing, the write operation has no effect; therefore, TOF remains set indicating another overflow has occurred. In this case a TOF interrupt request is not lost due to a delay in clearing the previous TOF. 0 TPM counter has not overflowed. 1 TPM counter has overflowed.
6 TOIE	Timer Overflow Interrupt Enable Enables TPM overflow interrupts. 0 Disable TOF interrupts. Use software polling or DMA request. 1 Enable TOF interrupts. An interrupt is generated when TOF equals one.
5 CPWMS	Center-Aligned PWM Select Selects CPWM mode. This mode configures the TPM to operate in up-down counting mode. This field is write protected. It can be written only when the counter is disabled. 0 TPM counter operates in up counting mode. 1 TPM counter operates in up-down counting mode.
4–3 CMOD	Clock Mode Selection Selects the TPM counter clock modes. When disabling the counter, this field remain set until acknolwedged in the TPM clock domain. 00 TPM counter is disabled 01 TPM counter increments on every TPM counter clock 10 TPM counter increments on rising edge of TPM_EXTCLK synchronized to the TPM counter clock 11 Reserved.
PS	Prescale Factor Selection Selects one of 8 division factors for the clock mode selected by CMOD. This field is write protected. It can be written only when the counter is disabled. 000 Divide by 1 001 Divide by 2 010 Divide by 4 011 Divide by 8 100 Divide by 16 101 Divide by 32

Table continues on the next page...

TPMx_SC field descriptions (continued)

Field	Description
	110 Divide by 64 111 Divide by 128

43.4.2 Counter (TPMx_CNT)

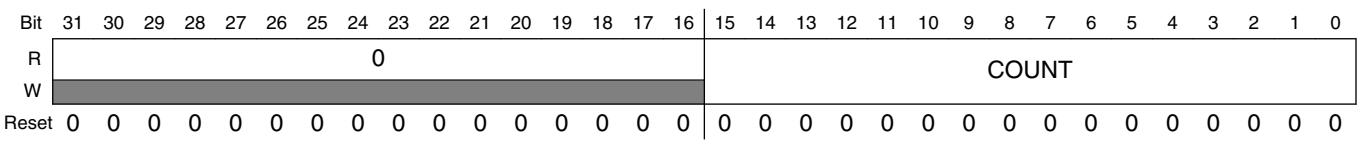
The CNT register contains the TPM counter value.

Reset clears the CNT register. Writing any value to COUNT also clears the counter.

When debug is active, the TPM counter does not increment unless configured otherwise.

Reading the CNT register adds two wait states to the register access due to synchronization delays.

Address: Base address + 4h offset



TPMx_CNT field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Counter value

43.4.3 Modulo (TPMx_MOD)

The Modulo register contains the modulo value for the TPM counter. When the TPM counter reaches the modulo value and increments, the overflow flag (TOF) is set and the next value of TPM 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 MOD Register Update . Additional writes to the MOD write buffer are ignored until the register has been updated.

It is recommended to initialize the TPM counter (write 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	0																MOD															
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

TPMx_MOD field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
MOD	Modulo value This field must be written with single 16-bit or 32-bit access.

43.4.4 Channel (n) Status and Control (TPMx_C*n*SC)

CnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function. When switching from one channel mode to a different channel mode, the channel must first be disabled and this must be acknowledged in the TPM counter clock domain.

Table 43-4. Mode, Edge, and Level Selection

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	00	00	None	Channel disabled
X	01	00	Software compare	Pin not used for TPM
0	00	01	Input capture	Capture on Rising Edge Only
		10		Capture on Falling Edge Only
		11		Capture on Rising or Falling Edge
	01	01	Output compare	Toggle Output on match
		10		Clear Output on match
		11		Set Output on match
	10	10	Edge-aligned PWM	High-true pulses (clear Output on match, set Output on reload)
		X1		Low-true pulses (set Output on match, clear Output on reload)
	11	10	Output compare	Pulse Output low on match
		01		Pulse Output high on match

Table continues on the next page...

Table 43-4. Mode, Edge, and Level Selection (continued)

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
1	10	10	Center-aligned PWM	High-true pulses (clear Output on match-up, set Output on match-down)
		01		Low-true pulses (set Output on match-up, clear Output on match-down)

Address: Base address + Ch offset + (8d × i), where i=0d to 5d

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								CHF	CHIE	MSB	MSA	ELSB	ELSA	0	DMA
W									w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TPMx_CnSC field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 CHF	Channel Flag Set by hardware when an event occurs on the channel. CHF is cleared by writing a 1 to the CHF bit. Writing a 0 to CHF has no effect. If another event occurs between the CHF sets and the write operation, the write operation has no effect; therefore, CHF remains set indicating another event has occurred. In this case a CHF interrupt request is not lost due to the delay in clearing the previous CHF. 0 No channel event has occurred. 1 A channel event has occurred.
6 CHIE	Channel Interrupt Enable Enables channel interrupts. 0 Disable channel interrupts. 1 Enable channel interrupts.
5 MSB	Channel Mode Select Used for further selections in the channel logic. Its functionality is dependent on the channel mode. When a channel is disabled, this field will not change state until acknowledged in the TPM counter clock domain.
4 MSA	Channel Mode Select

Table continues on the next page...

TPMx_C*n*SC field descriptions (continued)

Field	Description
	Used for further selections in the channel logic. Its functionality is dependent on the channel mode. When a channel is disabled, this field will not change state until acknowledged in the TPM counter clock domain.
3 ELSB	Edge or Level Select The functionality of ELSB and ELSA depends on the channel mode. When a channel is disabled, this field will not change state until acknowledged in the TPM counter clock domain.
2 ELSA	Edge or Level Select The functionality of ELSB and ELSA depends on the channel mode. When a channel is disabled, this field will not change state until acknowledged in the TPM counter clock domain.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 DMA	DMA Enable Enables DMA transfers for the channel. 0 Disable DMA transfers. 1 Enable DMA transfers.

43.4.5 Channel (n) Value (TPMx_C*n*V)

These registers contain the captured TPM counter value for the input modes or the match value for the output modes.

In input capture mode, any write to a CnV register is ignored.

In compare 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 CnV Register Update . Additional writes to the CnV write buffer are ignored until the register has been updated.

Address: Base address + 10h offset + (8d × i), where i=0d to 5d

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

TPMx_C*n*V field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
VAL	Channel Value Captured TPM counter value of the input modes or the match value for the output modes. This field must be written with single 16-bit or 32-bit access.

43.4.6 Capture and Compare Status (TPMx_STATUS)

The STATUS register contains a copy of the status flag, CnSC[CHnF] for each TPM channel, as well as SC[TOF], 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 writing all ones to STATUS.

Hardware sets the individual channel flags when an event occurs on the channel. Writing a 1 to CHF clears it. Writing a 0 to CHF has no effect.

If another event occurs between the flag setting and the write operation, the write operation has no effect; therefore, CHF remains set indicating another event has occurred. In this case a CHF interrupt request is not lost due to the clearing sequence for a previous CHF.

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							TOF	0		CH5F	CH4F	CH3F	CH2F	CH1F	CH0F
W								w1c			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TPMx_STATUS field descriptions

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 TOF	Timer Overflow Flag See register description

Table continues on the next page...

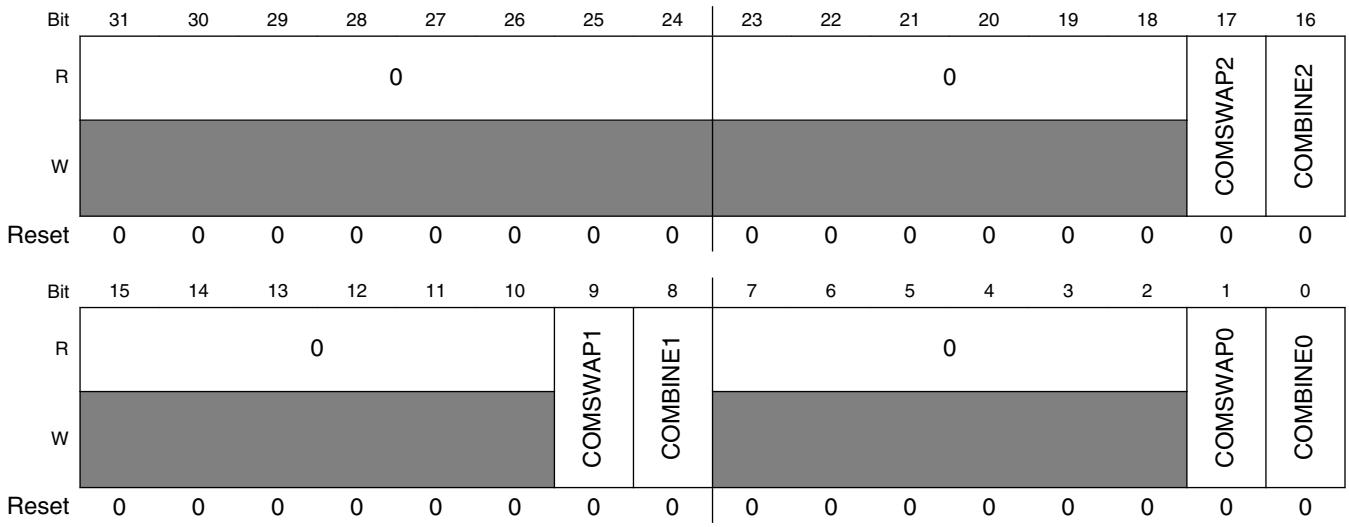
TPMx_STATUS field descriptions (continued)

Field	Description
	0 TPM counter has not overflowed. 1 TPM counter has overflowed.
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 CH5F	Channel 5 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
4 CH4F	Channel 4 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
3 CH3F	Channel 3 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
2 CH2F	Channel 2 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
1 CH1F	Channel 1 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
0 CH0F	Channel 0 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.

43.4.7 Combine Channel Register (TPMx_COMBINE)

This register contains the control bits used to configure the combine channel modes for each pair of channels (n) and (n+1), where n is all the even numbered channels.

Address: Base address + 64h offset



TPMx_COMBINE field descriptions

Field	Description
31–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 COMSWAP2	Combine Channels 4 and 5 Swap When set in combine mode, the even channel is used for the input capture and 1st compare, the odd channel is used for the 2nd compare. 0 Even channel is used for input capture and 1st compare. 1 Odd channel is used for input capture and 1st compare.
16 COMBINE2	Combine Channels 4 and 5 Enables the combine feature for channels 2 and 3. In input capture mode, the combined channels use the even channel input. In software compare modes, the even channel match asserts the output trigger and the odd channel match negates the output trigger. In PWM modes, the even channel match is used for the 1st compare and odd channel match for the 2nd compare. 0 Channels 4 and 5 are independent. 1 Channels 4 and 5 are combined.
15–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 COMSWAP1	Combine Channels 2 and 3 Swap

Table continues on the next page...

TPMx_COMBINE field descriptions (continued)

Field	Description
	When set in combine mode, the odd channel is used for the input capture and 1st compare, the even channel is used for the 2nd compare. 0 Even channel is used for input capture and 1st compare. 1 Odd channel is used for input capture and 1st compare.
8 COMBINE1	Combine Channels 2 and 3 Enables the combine feature for channels 2 and 3. In input capture mode, the combined channels use the even channel input. In software compare modes, the even channel match asserts the output trigger and the odd channel match negates the output trigger. In PWM modes, the even channel match is used for the 1st compare and odd channel match for the 2nd compare. 0 Channels 2 and 3 are independent. 1 Channels 2 and 3 are combined.
7–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 COMSWAP0	Combine Channel 0 and 1 Swap When set in combine mode, the even channel is used for the input capture and 1st compare, the odd channel is used for the 2nd compare. 0 Even channel is used for input capture and 1st compare. 1 Odd channel is used for input capture and 1st compare.
0 COMBINE0	Combine Channels 0 and 1 Enables the combine feature for channels 0 and 1. In input capture mode, the combined channels use the even channel input. In software compare modes, the even channel match asserts the output trigger and the odd channel match negates the output trigger. In PWM modes, the even channel match is used for the 1st compare and odd channel match for the 2nd compare. 0 Channels 0 and 1 are independent. 1 Channels 0 and 1 are combined.

43.4.8 Channel Polarity (TPMx_POL)

This register defines the input and output polarity of each of the channels.

Address: Base address + 70h 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										POL5	POL4	POL3	POL2	POL1	POL0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TPMx_POL field descriptions

Field	Description
31–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 POL5	Channel 5 Polarity 0 The channel polarity is active high. 1 The channel polarity is active low.
4 POL4	Channel 4 Polarity 0 The channel polarity is active high 1 The channel polarity is active low.
3 POL3	Channel 3 Polarity 0 The channel polarity is active high. 1 The channel polarity is active low.
2 POL2	Channel 2 Polarity 0 The channel polarity is active high. 1 The channel polarity is active low.
1 POL1	Channel 1 Polarity 0 The channel polarity is active high. 1 The channel polarity is active low.
0 POL0	Channel 0 Polarity 0 The channel polarity is active high. 1 The channel polarity is active low.

43.4.9 Filter Control (TPMx_FILTER)

This register selects the filter value of the channel inputs, and an additional output delay value for the channel outputs. In PWM combine modes, the filter can effectively implements deadtime insertion.

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	0								CH5FVAL				CH4FVAL				CH3FVAL				CH2FVAL				CH1FVAL				CH0FVAL			
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

TPMx_FILTER field descriptions

Field	Description
31–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

TPMx_FILTER field descriptions (continued)

Field	Description
23–20 CH5FVAL	Channel 5 Filter Value Selects the filter value for the channel input and the delay value for the channel output. The filter/delay is disabled when the value is zero, otherwise the filter/delay is configured as (CH5FVAL * 4) clock cycles.
19–16 CH4FVAL	Channel 4 Filter Value Selects the filter value for the channel input and the delay value for the channel output. The filter/delay is disabled when the value is zero, otherwise the filter/delay is configured as (CH4FVAL * 4) clock cycles.
15–12 CH3FVAL	Channel 3 Filter Value Selects the filter value for the channel input and the delay value for the channel output. The filter/delay is disabled when the value is zero, otherwise the filter/delay is configured as (CH3FVAL * 4) clock cycles.
11–8 CH2FVAL	Channel 2 Filter Value Selects the filter value for the channel input and the delay value for the channel output. The filter/delay is disabled when the value is zero, otherwise the filter/delay is configured as (CH2FVAL * 4) clock cycles.
7–4 CH1FVAL	Channel 1 Filter Value Selects the filter value for the channel input and the delay value for the channel output. The filter/delay is disabled when the value is zero, otherwise the filter/delay is configured as (CH1FVAL * 4) clock cycles.
CH0FVAL	Channel 0 Filter Value Selects the filter value for the channel input and the delay value for the channel output. The filter/delay is disabled when the value is zero, otherwise the filter/delay is configured as (CH0FVAL * 4) clock cycles.

43.4.10 Configuration (TPMx_CONF)

This register selects the behavior in debug and wait modes and the use of an external global time base.

Address: Base address + 84h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				TRGSEL				TRGSRC	TRGPOL	0		CPOT	CROT	CSOO	CSOT
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						GTBEEN	GTBSYNC	DBGMODE		DOZEEN	0				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TPMx_CONF field descriptions

Field	Description
31–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–24 TRGSEL	Trigger Select Selects the input trigger to use for starting, reloading and/or pausing the counter. The source of the trigger (external or internal to the TPM) is configured by the TRGSRC field. This field should only be changed when the TPM counter is disabled. Refer to the chip configuration section for available external trigger options. The available internal trigger sources are listed below. 0001 Channel 0 pin input capture 0010 Channel 1 pin input capture 0011 Channel 0 or Channel 1 pin input capture 0100 Channel 2 pin input capture 0101 Channel 0 or Channel 2 pin input capture 0110 Channel 1 or Channel 2 pin input capture 0111 Channel 0 or Channel 1 or Channel 2 pin input capture 1000 Channel 3 pin input capture 1001 Channel 0 or Channel 3 pin input capture 1010 Channel 1 or Channel 3 pin input capture 1011 Channel 0 or Channel 1 or Channel 3 pin input capture 1100 Channel 2 or Channel 3 pin input capture 1101 Channel 0 or Channel 2 or Channel 3 pin input capture 1110 Channel 1 or Channel 2 or Channel 3 pin input capture 1111 Channel 0 or Channel 1 or Channel 2 or Channel 3 pin input capture
23 TRGSRC	Trigger Source Selects between internal (channel pin input capture) or external trigger sources. When selecting an internal trigger, the channel selected should be configured for input capture. Only a rising edge input capture can be used to initially start the counter using the CSOT configuration; either rising edge or falling edge input capture can be used to reload the counter using the CROT configuration; and the state of the channel input pin is used to pause the counter using the CPOT configuration. The channel polarity register can be used to invert the polarity of the channel input pins. This field should only be changed when the TPM counter is disabled. 0 Trigger source selected by TRGSEL is external. 1 Trigger source selected by TRGSEL is internal (channel pin input capture).
22 TRGPOL	Trigger Polarity Selects the polarity of the external trigger source. This field should only be changed when the TPM counter is disabled. 0 Trigger is active high. 1 Trigger is active low.
21–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 CPOT	Counter Pause On Trigger When enabled, the counter will pause incrementing while the trigger remains asserted (level sensitive). This field should only be changed when the TPM counter is disabled.

Table continues on the next page...

TPMx_CONF field descriptions (continued)

Field	Description
18 CROT	Counter Reload On Trigger When set, the TPM counter will reload with 0 (and initialize PWM outputs to their default value) when a rising edge is detected on the selected trigger input. The trigger input is ignored if the TPM counter is paused during debug mode or doze mode. This field should only be changed when the TPM counter is disabled. 0 Counter is not reloaded due to a rising edge on the selected input trigger 1 Counter is reloaded when a rising edge is detected on the selected input trigger
17 CSOO	Counter Stop On Overflow When set, the TPM counter will stop incrementing once the counter equals the MOD value and incremented (this also sets the TOF). Reloading the counter with 0 due to writing to the counter register or due to a trigger input does not cause the counter to stop incrementing. Once the counter has stopped incrementing, the counter will not start incrementing unless it is disabled and then enabled again, or a rising edge on the selected trigger input is detected when CSOT set. This field should only be changed when the TPM counter is disabled. 0 TPM counter continues incrementing or decrementing after overflow 1 TPM counter stops incrementing or decrementing after overflow.
16 CSOT	Counter Start on Trigger When set, the TPM counter will not start incrementing after it is enabled until a rising edge on the selected trigger input is detected. If the TPM counter is stopped due to an overflow, a rising edge on the selected trigger input will also cause the TPM counter to start incrementing again. The trigger input is ignored if the TPM counter is paused during debug mode or doze mode. This field should only be changed when the TPM counter is disabled. 0 TPM counter starts to increment immediately, once it is enabled. 1 TPM counter only starts to increment when it a rising edge on the selected input trigger is detected, after it has been enabled or after it has stopped due to overflow.
15–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 GTBEEN	Global time base enable Configures the TPM to use an externally generated global time base counter. When an externally generated timebase is used, the internal TPM counter is not used by the channels but can be used to generate a periodic interruptor DMA request using the Modulo register and timer overflow flag. 0 All channels use the internally generated TPM counter as their timebase 1 All channels use an externally generated global timebase as their timebase
8 GTBSYNC	Global Time Base Synchronization When enabled, the TPM counter is synchronized to the global time base. It uses the global timebase enable, trigger and overflow to ensure the TPM counter starts incrementing at the same time as the global timebase, stops incrementing at the same time as the global timebase and is reset at the same time as the global timebase. This field should only be changed when the TPM counter is disabled. 0 Global timebase synchronization disabled. 1 Global timebase synchronization enabled.
7–6 DBGMODE	Debug Mode

Table continues on the next page...

TPMx_CONF field descriptions (continued)

Field	Description
	Configures the TPM behavior in debug mode. All other configurations are reserved. 00 TPM counter is paused and does not increment during debug mode. Trigger inputs and input capture events are also ignored. 11 TPM counter continues in debug mode.
5 DOZEEN	Doze Enable Configures the TPM behavior in wait mode. 0 Internal TPM counter continues in Doze mode. 1 Internal TPM counter is paused and does not increment during Doze mode. Trigger inputs and input capture events are also ignored.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

43.5 Functional description

The following sections describe the TPM features.

43.5.1 Clock domains

The TPM module supports two clock domains.

The bus clock domain is used by the register interface and for synchronizing interrupts and DMA requests.

The TPM counter clock domain is used to clock the counter and prescaler along with the output compare and input capture logic. The TPM counter clock is considered asynchronous to the bus clock, can be a higher or lower frequency than the bus clock and can remain operational in Stop mode. Multiple TPM instances are all clocked by the same TPM counter clock in support of the external timebase feature.

43.5.1.1 Counter Clock Mode

The CMOD[1:0] bits in the SC register either disable the TPM counter or select one of two possible clock modes for the TPM counter. After any reset, CMOD[1:0] = 0:0 so the TPM counter is disabled.

The CMOD[1:0] bits may be read or written at any time. Disabling the TPM counter by writing zero to the CMOD[1:0] bits does not affect the TPM counter value or other registers, but must be acknowledged by the TPM counter clock domain before they read as zero.

The external clock input passes through a synchronizer clocked by the TPM counter clock to assure that counter transitions are properly aligned to counter clock transitions. Therefore, to meet Nyquist criteria considering also jitter, the frequency of the external clock source must be less than half of the counter clock frequency.

43.5.2 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 TPM counter.

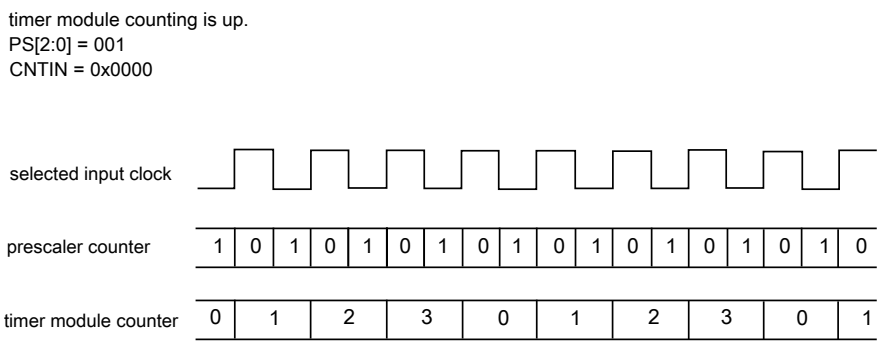


Figure 43-2. Example of the Prescaler Counter

43.5.3 Counter

The TPM has a 16-bit counter that is used by the channels either for input or output modes.

The counter updates from the selected clock divided by the prescaler.

The TPM counter has these modes of operation:

- up counting (see Up counting)
- up-down counting (see Up-down counting)

43.5.3.1 Up counting

Up counting is selected when SC[CPWMS] = 0.

The value of zero is loaded into the TPM counter, and the counter increments until the value of MOD is reached, at which point the counter is reloaded with zero.

The TPM period when using up counting is (MOD + 0x0001) × period of the TPM counter clock.

The TOF bit is set when the TPM counter changes from MOD to zero.

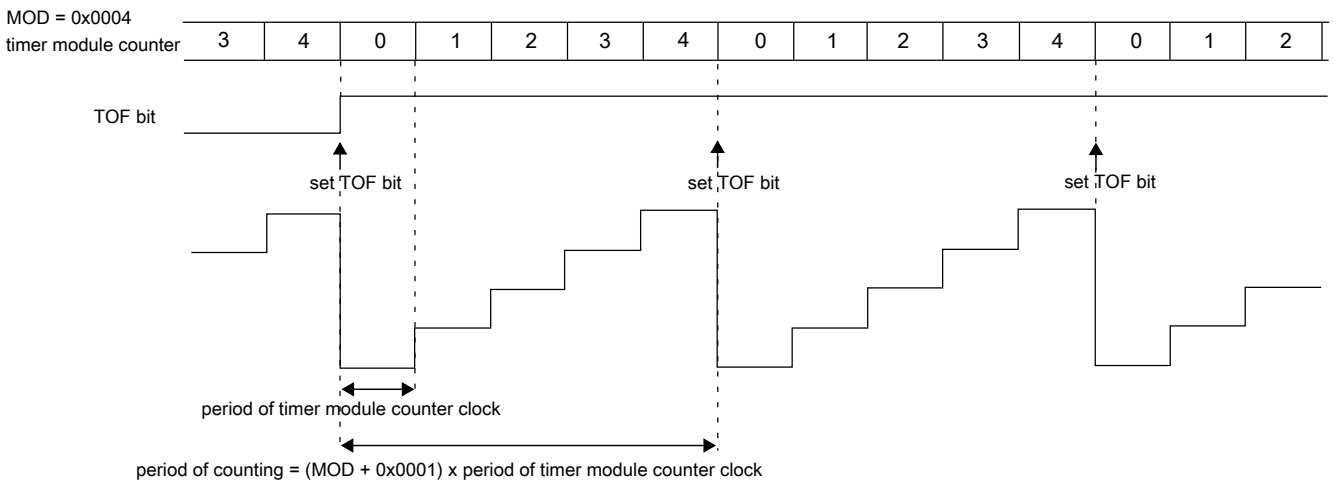


Figure 43-3. Example of TPM Up Counting

Note

- MOD = 0000 is a redundant condition. In this case, the TPM counter is always equal to MOD and the TOF bit is set in each rising edge of the TPM counter clock.

43.5.3.2 Up-down counting

Up-down counting is selected when SC[CPWMS] = 1. When configured for up-down counting, configuring CONF[MOD] to less than 2 is not supported.

The value of 0 is loaded into the TPM counter, and the counter increments until the value of MOD is reached, at which point the counter is decremented until it returns to zero and the up-down counting restarts.

The TPM period when using up-down counting is 2 × MOD × period of the TPM counter clock.

The TOF bit is set when the TPM counter changes from MOD to (MOD – 1).

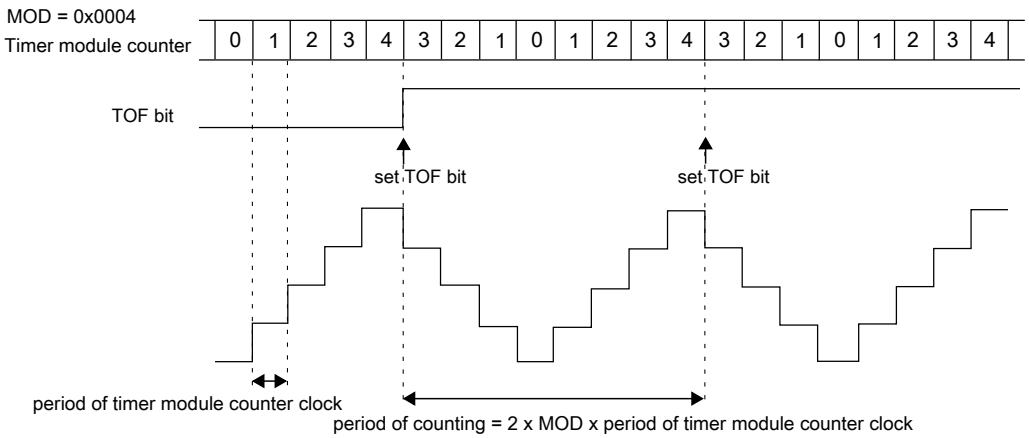


Figure 43-4. Example of up-down counting

43.5.3.3 Counter Reset

Any write to CNT resets the TPM counter and the channel outputs to their initial values (except for channels in output compare mode).

43.5.3.4 Global time base (GTB)

The global time base (GTB) is a TPM function that allows multiple TPM modules to share the same timebase. When the global time base is enabled (CONF[GTBEEN] = 1), the local TPM channels use the counter value, counter enable and overflow indication from the TPM generating the global time base. If the local TPM counter is not generating the global time base, then it can be used as an independent counter or pulse accumulator.

The local TPM counter can also be configured to synchronize to the global time base, by configuring (GTBSYNC = 1). When synchronized to the global time base, the local counter will use the counter enable and counter overflow indication from the TPM generating the global time base. This enables multiple TPM to be configured with the same phase, but with different periods (although the global time base must be configured with the longest period).

43.5.3.5 Counter trigger

The TPM counter can be configured to start, stop or reset in response to a hardware trigger input. The trigger input is synchronized to the asynchronous counter clock, so there is a 3 counter clock delay between the trigger assertion and the counter responding.

- When (CSOT = 1), the counter will not start incrementing until a rising edge is detected on the trigger input.
- When (CSOO= 1), the counter will stop incrementing whenever the TOF flag is set. The counter does not increment again unless it is disabled, or if CSOT = 1 and a rising edge is detected on the trigger input.
- When (CROT= 1), the counter will reset to zero as if an overflow occurred whenever a rising edge is detected on the trigger input.
- When (CPOT = 1), the counter will pause incrementing whenever the trigger input is asserted. The counter will continue incrementing when the trigger input negates.

The polarity of the external input trigger can be configured by the TRGPOL register bit.

When an internal trigger source is selected, the trigger input is selected from one or more channel input capture events. The input capture filters are used with the internal trigger sources and the POLn bits can be used to invert the polarity of the input channels. Note that following restrictions apply with input capture channel sources.

- When (CSOT = 1), the counter will only start incrementing on a rising edge on the channel input, provided ELSnA = 1.
- When (CROT= 1), the counter will reset to zero on either edge of the channel input, as configured by ELSnB:ELSnA.
- When (CPOT = 1), the counter will pause incrementing whenever the channel input is asserted.

43.5.4 Input Capture Mode

The input capture mode is selected when (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 TPM 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 TPM_CHn 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 counter clock divided by 4, which is required to meet Nyquist criteria for signal sampling.

Writes to the CnV register are ignored in input capture mode.

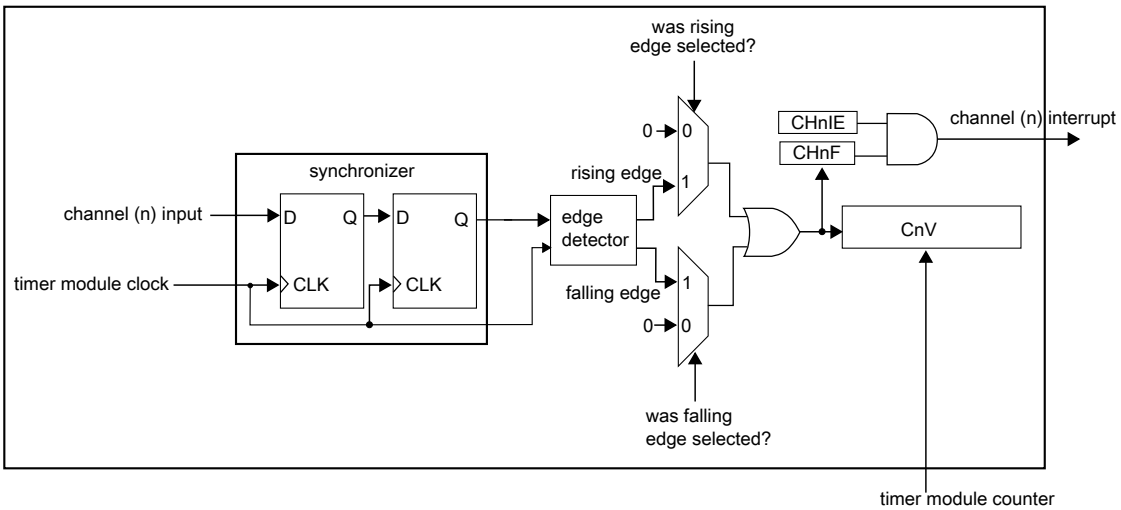


Figure 43-5. Input capture mode

The CHnF bit is set on the third rising edge of the counter clock after a valid edge occurs on the channel input.

43.5.5 Output Compare Mode

The output compare mode is selected when (CPWMS = 0), and (MSnB:MSnA = X:1).

In output compare mode, the TPM 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 if MSnB is clear. If MSnB is set then the channel (n) output is pulsed high or low for as long as the counter matches the value in the CnV register.

When a channel is initially configured to output compare mode, the channel output updates with its negated value (logic 0 for set/toggle/pulse high and logic one for clear/ pulse low).

The CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (TPM counter = CnV).

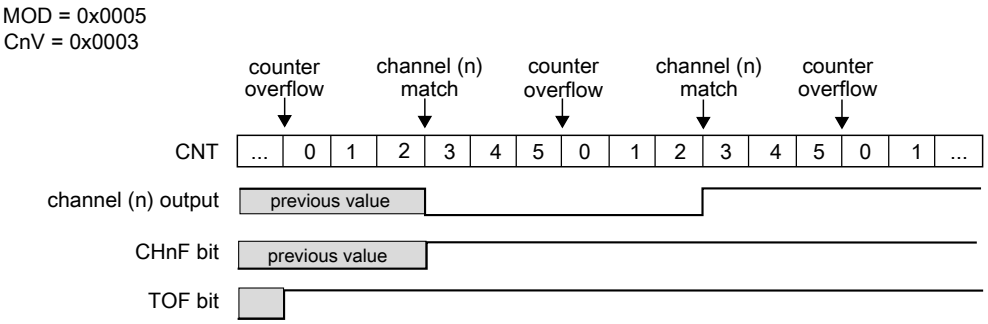


Figure 43-6. Example of the output compare mode when the match toggles the channel output

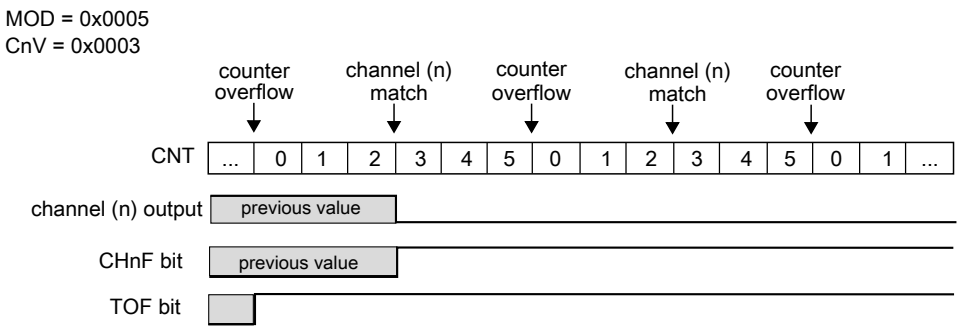


Figure 43-7. Example of the output compare mode when the match clears the channel output

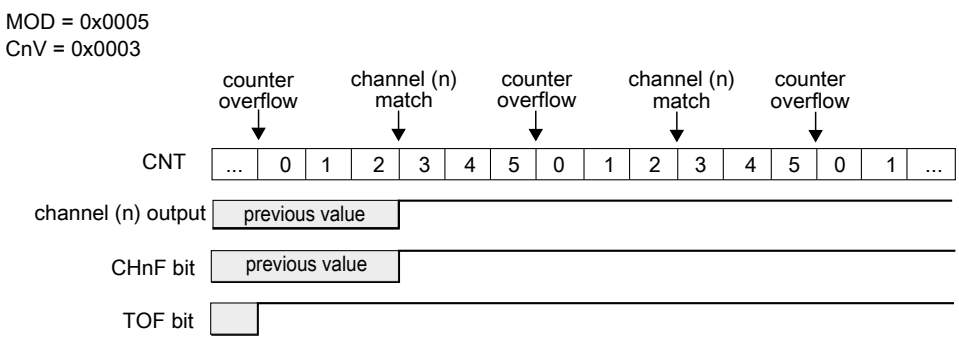


Figure 43-8. Example of the output compare mode when the match sets the channel output

It is possible to use the output compare mode with (ELSnB:ELSnA = 0:0). In this case, 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 TPM.

43.5.6 Edge-Aligned PWM (EPWM) Mode

The edge-aligned mode is selected when (CPWMS = 0), and (MSnB:MSnA = 1:0).

The EPWM period is determined by (MOD + 0x0001) and the pulse width (duty cycle) is determined by CnV.

The CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (TPM 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 TPM.

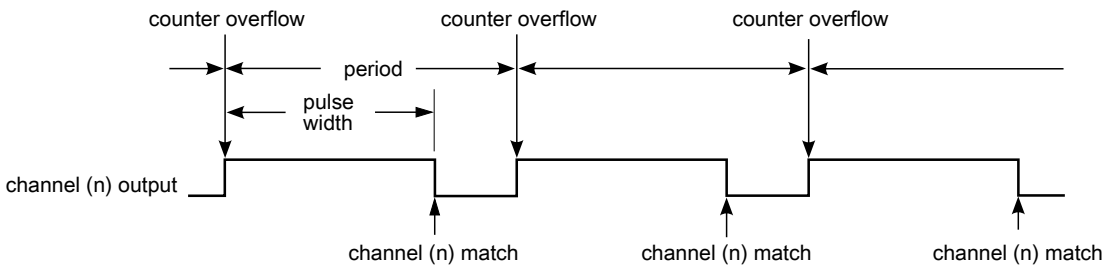


Figure 43-9. EPWM period and pulse width with ELSnB:ELSnA = 1:0

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 TPM.

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced high at the counter overflow (when the zero is loaded into the TPM counter), and it is forced low at the channel (n) match (TPM counter = CnV) (see the following figure).

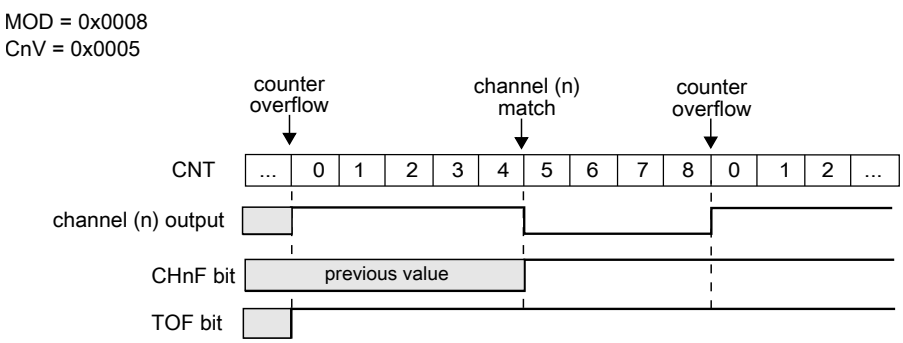


Figure 43-10. EPWM signal with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced low at the counter overflow (when zero is loaded into the TPM counter), and it is forced high at the channel (n) match (TPM counter = CnV) (see the following figure).

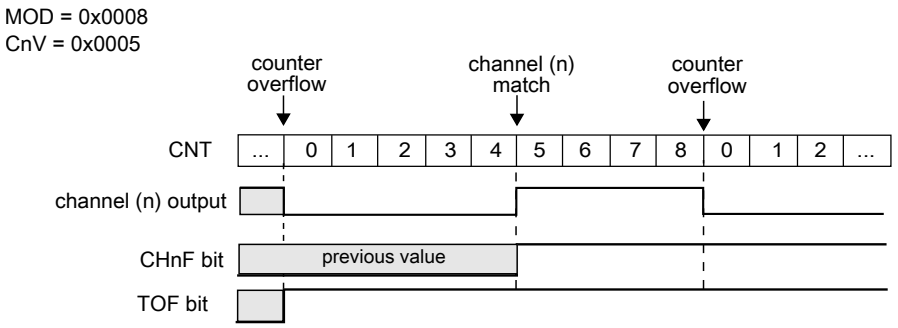


Figure 43-11. EPWM signal with ELSnB:ELSnA = X:1

If (CnV = 0x0000), then the channel (n) output is a 0% duty cycle EPWM signal. If (CnV > MOD), then the channel (n) output is a 100% duty cycle EPWM signal and CHnF bit is not set since there is never a channel (n) match. Therefore, MOD must be less than 0xFFFF in order to get a 100% duty cycle EPWM signal.

43.5.7 Center-Aligned PWM (CPWM) Mode

The center-aligned mode is selected when (CPWMS = 1) and (MSnB:MSnA = 1:0).

The CPWM pulse width (duty cycle) is determined by 2 × CnV and the period is determined by 2 × MOD (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 TPM counter counts up until it reaches MOD and then counts down until it reaches zero.

The CHnF bit is set and channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (TPM counter = CnV) when the TPM counting is down (at the begin of the pulse width) and when the TPM 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 when the TPM counter is zero.

The other channel modes are not designed to be used with the up-down counter (CPWMS = 1). Therefore, all TPM channels should be used in CPWM mode when (CPWMS = 1).

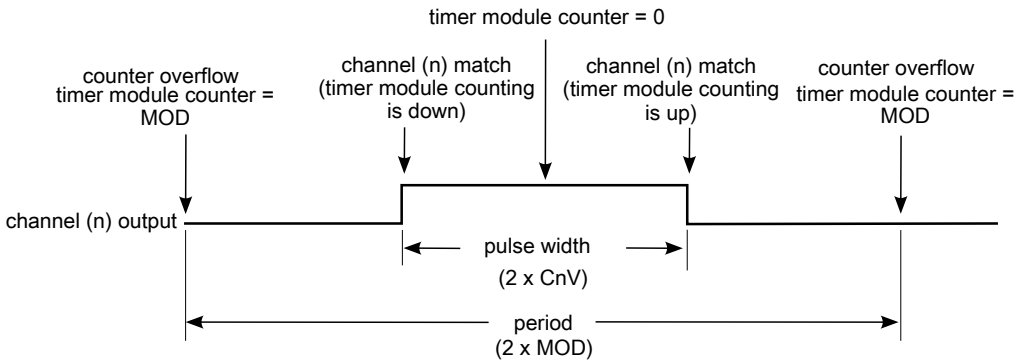


Figure 43-12. CPWM period and pulse width with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = 0:0) when the TPM 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 TPM.

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced high at the channel (n) match (TPM counter = CnV) when counting down, and it is forced low at the channel (n) match when counting up (see the following figure).

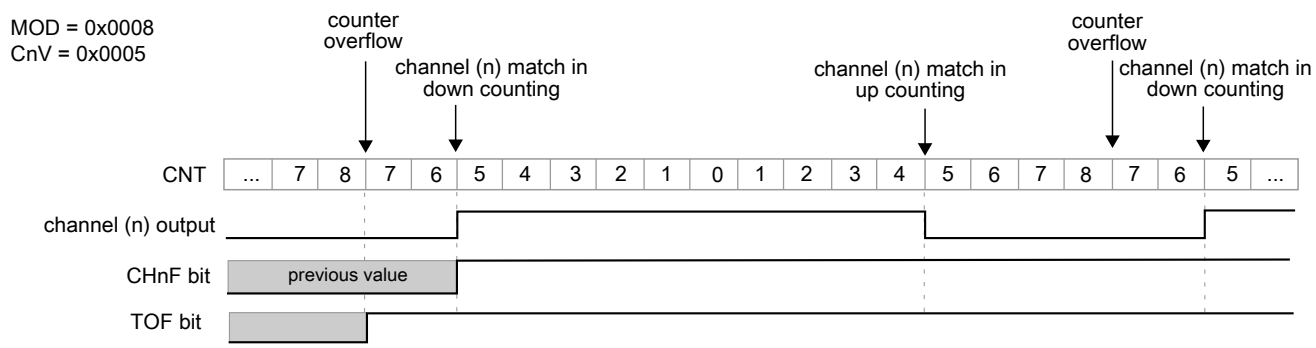


Figure 43-13. CPWM signal with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced low at the channel (n) match (TPM counter = CnV) when counting down, and it is forced high at the channel (n) match when counting up (see the following figure).

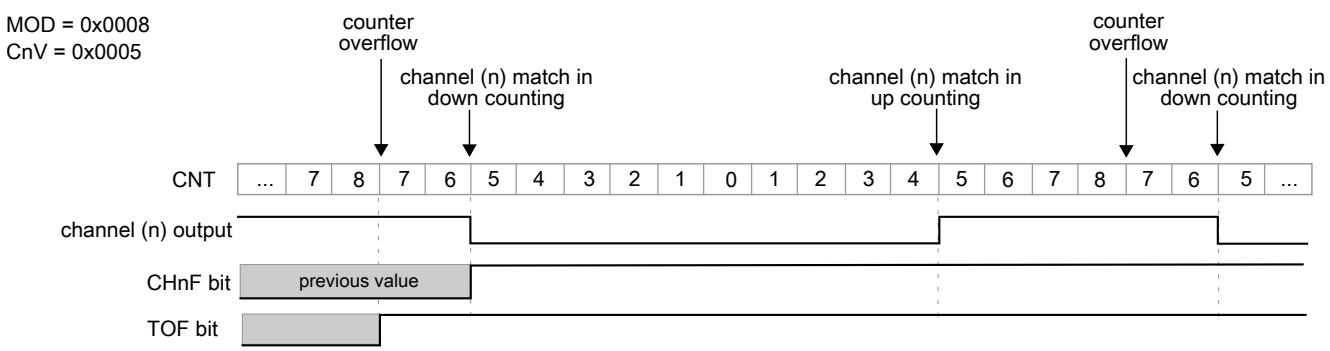


Figure 43-14. CPWM signal with ELSnB:ELSnA = X:1

If (CnV = 0x0000) then the channel (n) output is a 0% duty cycle CPWM signal.

If (CnV > MOD), then the channel (n) output is a 100% duty cycle CPWM signal, although the CHnF bit is set when the counter changes from incrementing to decrementing. Therefore, MOD must be less than 0xFFFF in order to get a 100% duty cycle CPWM signal.

43.5.8 Combine PWM mode

The Combine PWM mode is selected when:

- MSnB:MSnA = 10
- COMBINEn = 1
- QUADEN = 0, and
- CPWMS = 0

In Combine PWM 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 + 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 (TPM 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 (TPM counter = C(n+1)V).

If channel (n) (ELSnB:ELSnA = X:1), then the channel (n) output is forced low at the beginning of the period (TPM counter is zero) and at the channel (n+1) match (TPM counter = C(n+1)V). It is forced high at the channel (n) match (TPM counter = C(n)V).

If channel (n) (ELSnB:ELSnA = 1:0), then the channel (n) output is forced high at the beginning of the period (TPM counter is zero) and at the channel (n+1) match (TPM counter = C(n+1)V). It is forced low at the channel (n) match (TPM counter = C(n)V).

When (COMSWAPn = 1), then the channel (n) output is forced low or high at the beginning of the period (TPM counter is zero) and at the channel (n) match (TPM counter = C(n)V). It is forced high or low at the channel (n+1) match (TPM counter = C(n+1)V).

The channel (n+1) output is generated the same as the channel (n) output, but the output polarity is controlled by the channel (n+1) ELSnB:ELSnA configuration.

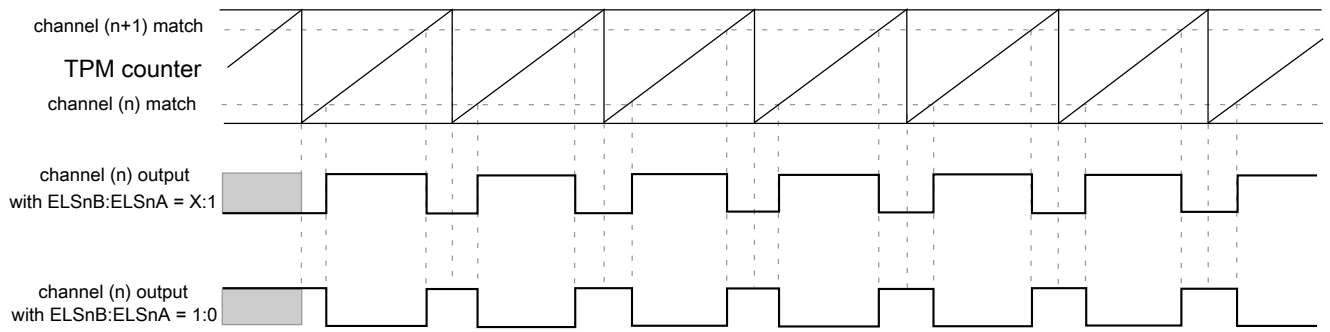


Figure 43-15. Combine mode

The following figures illustrate the PWM signals generation using Combine mode.

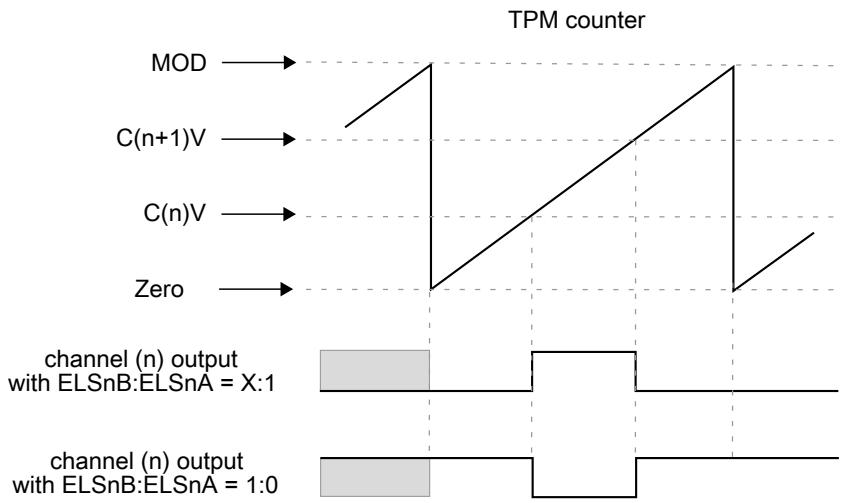


Figure 43-16. Channel (n) output if (C(n)V < MOD) and (C(n+1)V < MOD) and (C(n)V < C(n+1)V)

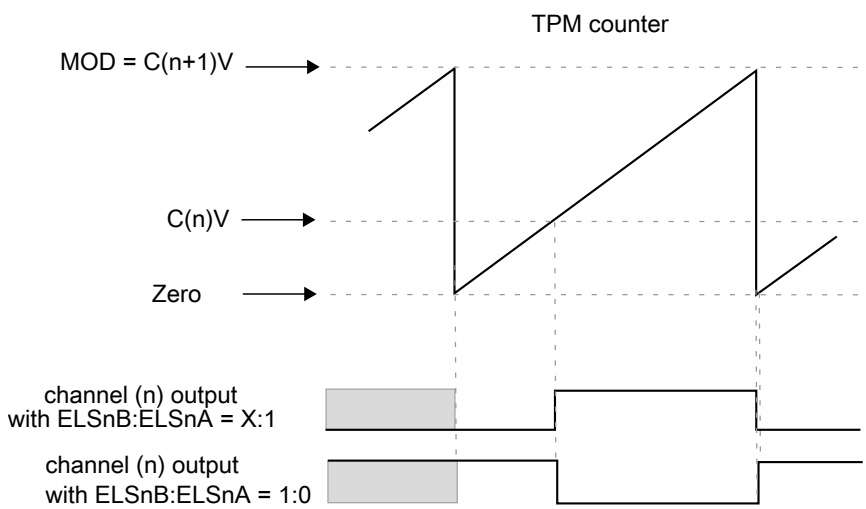


Figure 43-17. Channel (n) output if (C(n)V < MOD) and (C(n+1)V = MOD)

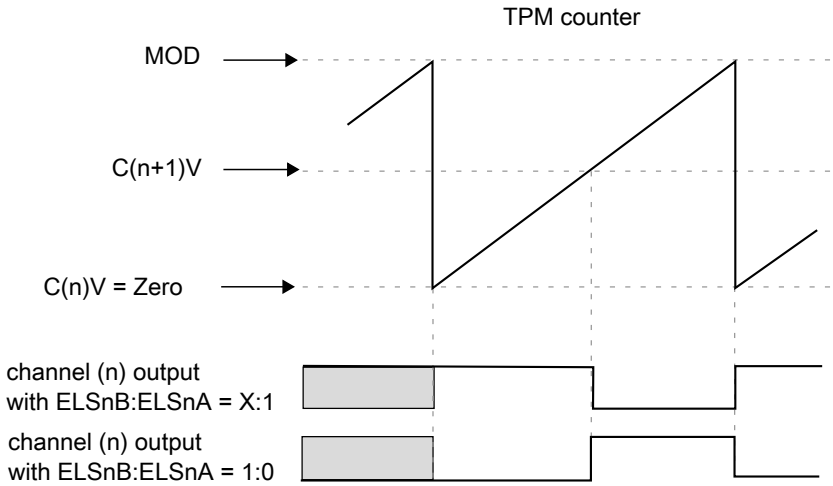


Figure 43-18. Channel (n) output if (C(n)V = zero) and (C(n+1)V < MOD)

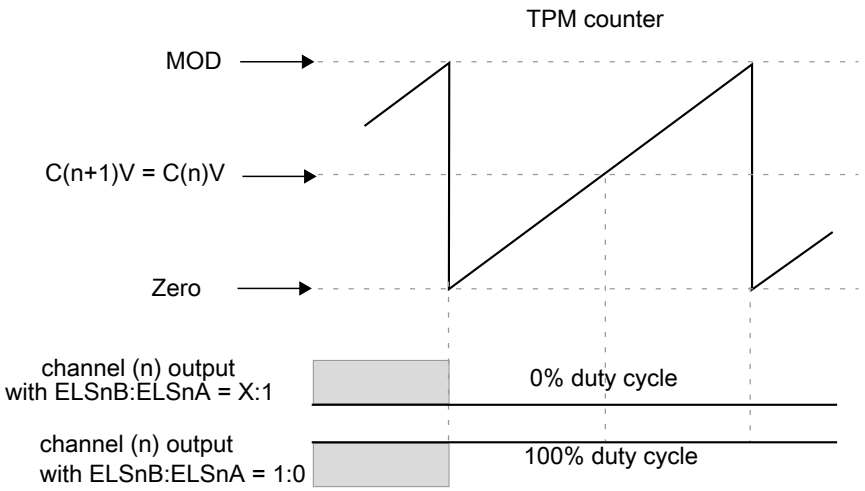


Figure 43-19. Channel (n) output if (C(n)V < MOD) and (C(n+1)V < MOD) and (C(n)V = C(n +1)V)

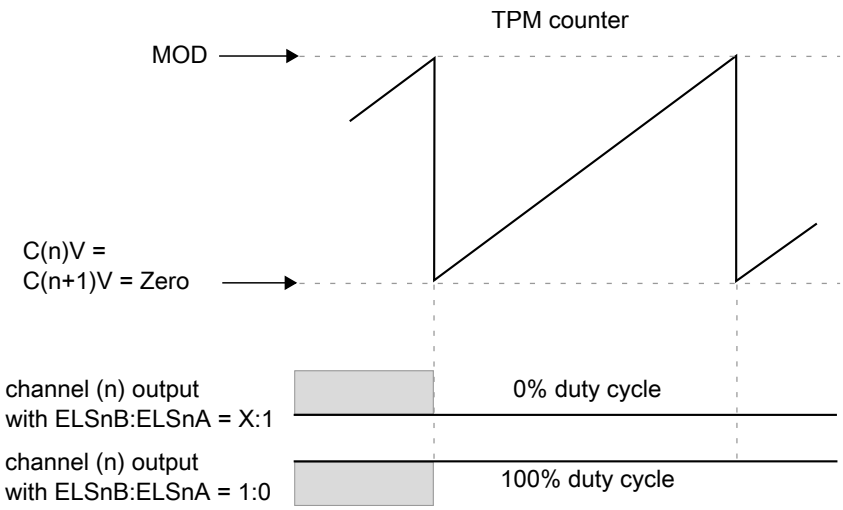


Figure 43-20. Channel (n) output if (C(n)V = C(n+1)V = zero)

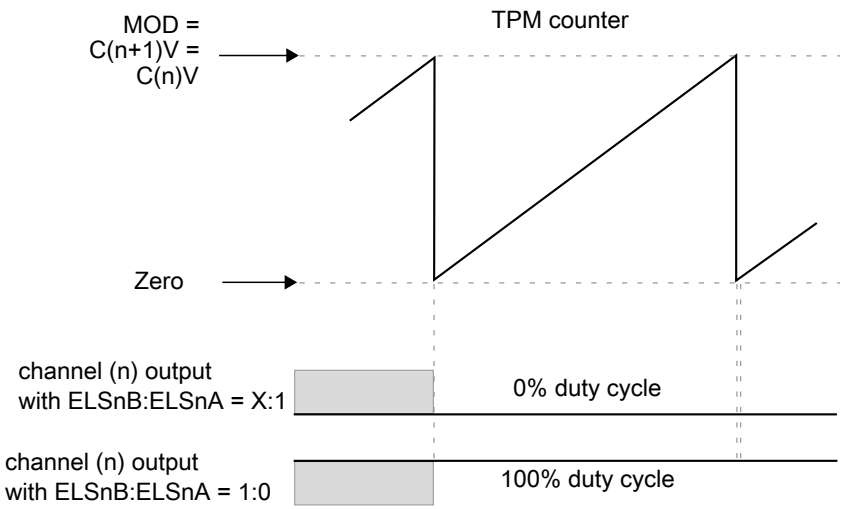


Figure 43-21. Channel (n) output if (C(n)V = C(n+1)V = MOD)

43.5.9 Combine Input Capture mode

The Combine Input Capture mode is selected if COMBINEn= 1 and MSnB:MSnA = 00 and ELSnB:ELSnA != 00. This mode allows to measure a pulse width of the signal on the input of channel (n) of a channel pair. The channel (n) filter can be active in this mode.

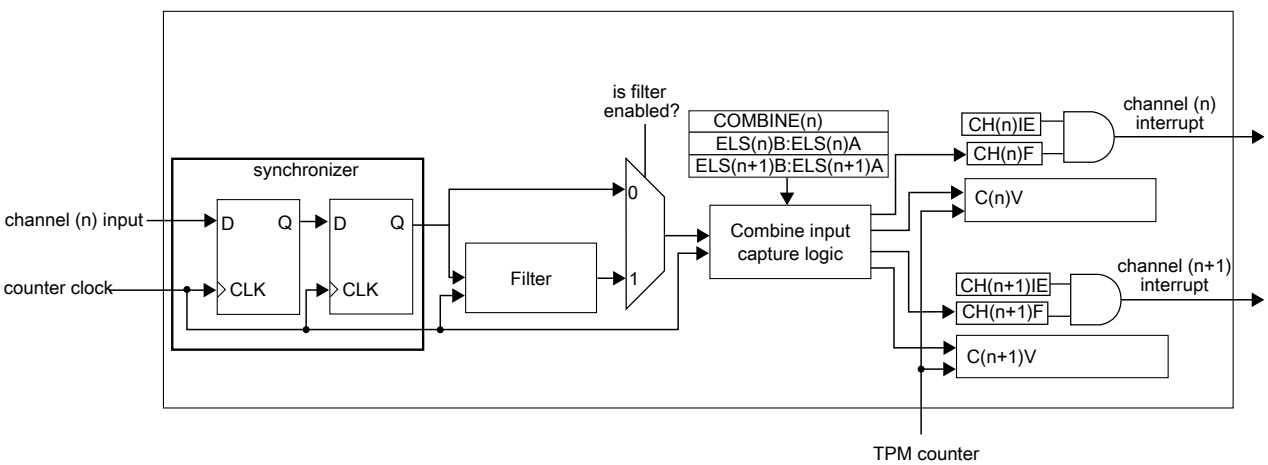


Figure 43-22. Combine Input Capture mode block diagram

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).

In the Combine Input Capture mode, only channel (n) input is used and channel (n+1) input is ignored, when COMSWAPn=1 then only channel (n+1) input is used and channel (n) 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, 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 TPM counter when the selected edge by channel (n) is detected at channel (n) input. The C(n+1)V register stores the value of TPM counter when the selected edge by channel (n+1) is detected at channel (n) input.

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 Combine Input 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.

43.5.10 Input Capture Filter

The input capture filter function is only in input capture mode, or in software compare mode when quadrature decoder mode is enabled.

First, the input signal is synchronized by the counter clock. Following synchronization, the input signal enters the filter block. See the following figure.

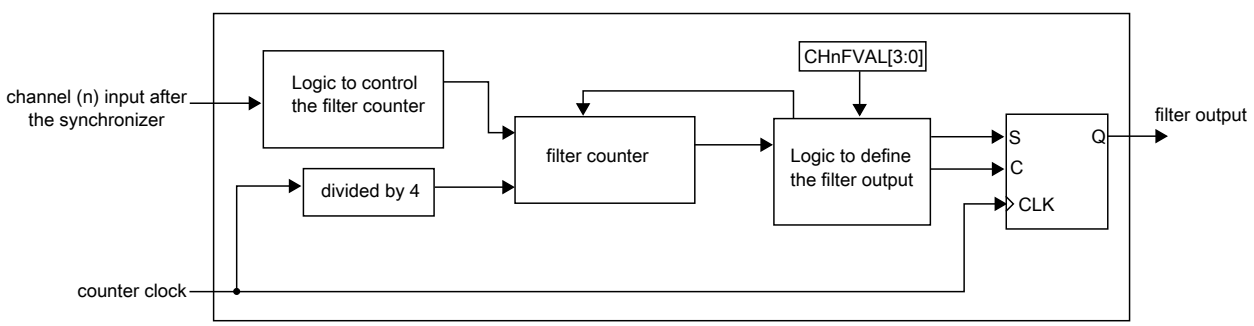


Figure 43-23. Channel input filter

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] × 4), the state change of the input signal is validated.

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 counter clocks) is regarded as a glitch and is not passed through the filter. 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 by 2 rising edges of the counter 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 3 rising edges of the system clock: two rising edges to the synchronizer, plus one more to the edge detector. In other words, CHnF is set (3 + 4 × CHnFVAL[3:0]) counter clock periods after a valid edge occurs on the channel input.

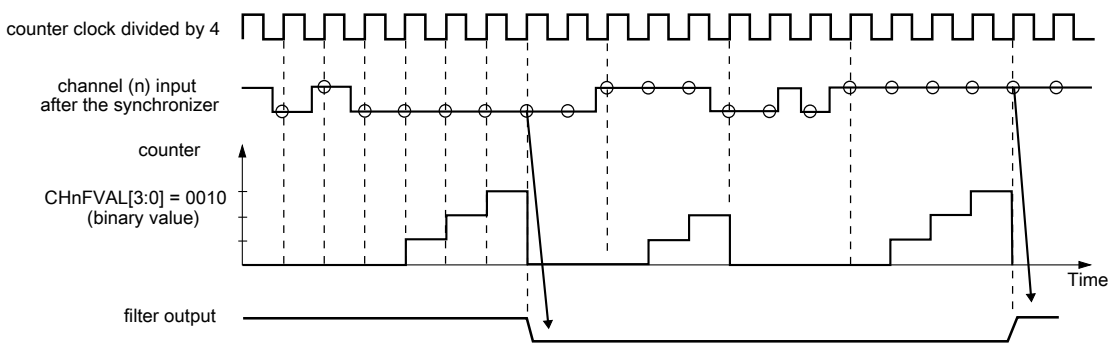


Figure 43-24. Channel input filter example

43.5.11 Deadtime insertion

The deadtime insertion is enabled in PWM combine modes when CHnFVAL is non-zero. The deadtime delay that is used for each TPM channel is defined as (CHnFVAL[3:0] x 4).

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) = 1, and the deadtime is enabled, then when the channel (n) match (TPM 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 (TPM 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) = 0, and the deadtime is enabled, then when the channel (n) match (TPM 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 (TPM 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.

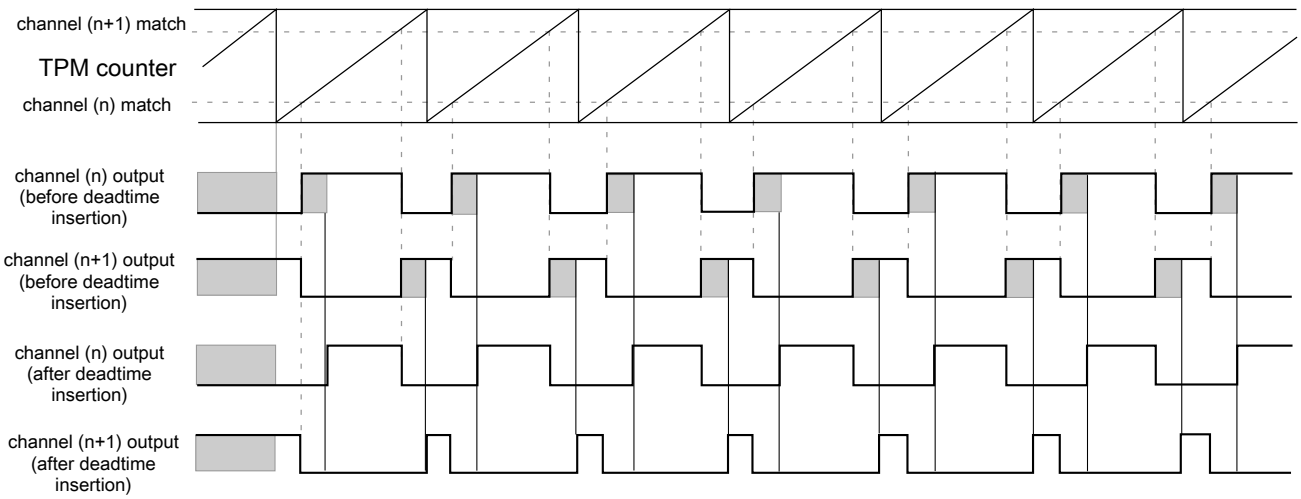


Figure 43-25. Deadtime insertion with ELSnB:ELSnA = X:1, POL(n) = 0, and POL(n+1) = 1

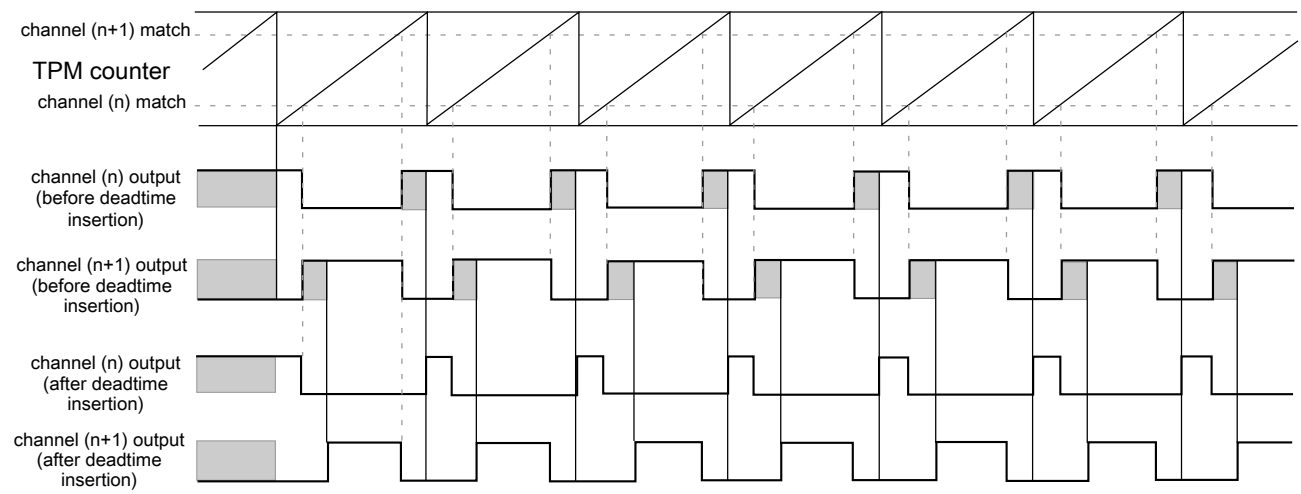


Figure 43-26. Deadtime insertion with ELSnB:ELSnA = 1:0, POL(n) = 0, and POL(n+1) = 1

43.5.12 Registers Updated from Write Buffers

43.5.12.1 MOD Register Update

If (CMOD[1:0] = 0:0) then MOD register is updated when MOD register is written.

If (CMOD[1:0] ≠ 0:0), then MOD register is updated according to the CPWMS bit, that is:

- If the selected mode is not CPWM then MOD register is updated after MOD register was written and the TPM counter changes from MOD to zero.
- If the selected mode is CPWM then MOD register is updated after MOD register was written and the TPM counter changes from MOD to (MOD – 1).

43.5.12.2 CnV Register Update

If (CMOD[1:0] = 0:0) then CnV register is updated when CnV register is written.

If (CMOD[1:0] ≠ 0:0), then CnV register is updated according to the selected mode, that is:

- If the selected mode is output compare then CnV register is updated on the next TPM counter increment (end of the prescaler counting) after CnV register was written.
- If the selected mode is EPWM then CnV register is updated after CnV register was written and the TPM counter changes from MOD to zero.
- If the selected mode is CPWM then CnV register is updated after CnV register was written and the TPM counter changes from MOD to (MOD – 1).

43.5.13 DMA

The channel and overflow flags generate a DMA transfer request according to DMA and CHnIE/TOIE bits.

See the following table for more information.

Table 43-5. DMA Transfer Request

DMA	CHnIE/ TOIE	Channel/Overflow DMA Transfer Request	Channel/Overflow Interrupt
0	0	The channel/overflow DMA transfer request is not generated.	The channel/overflow interrupt is not generated.
0	1	The channel/overflow DMA transfer request is not generated.	The channel/overflow interrupt is generated if (CHnF/TOF = 1).
1	0	The channel/overflow DMA transfer request is generated if (CHnF/TOF = 1).	The channel/overflow interrupt is not generated.
1	1	The channel/overflow DMA transfer request is generated if (CHnF/TOF = 1).	The channel/overflow interrupt is generated if (CHnF/TOF = 1).

If DMA = 1, the CHnF/TOF bit can be cleared either by DMA transfer done or writing a one to CHnF/TOF bit (see the following table).

Table 43-6. Clear CHnF/TOF Bit

DMA	How CHnF/TOF Bit Can Be Cleared
0	CHnF/TOF bit is cleared by writing a 1 to CHnF/TOF bit.
1	CHnF/TOF bit is cleared either when the DMA transfer is done or by writing a 1 to CHnF/TOF bit.

43.5.14 Output triggers

The TPM generates output triggers for the counter and each channel that can be used to trigger events in other peripherals.

The counter trigger asserts whenever the TOF is set and remains asserted until the next increment.

Each TPM channel generates both a pre-trigger output and a trigger output. The pre-trigger output asserts whenever the CHnF is set, the trigger output asserts on the first counter increment after the pre-trigger asserts, and then both the trigger and pre-trigger negate on the first counter increment after the trigger asserts.

When (COMBINEn = 1) in output compare modes, the pre-trigger output for both channel (n) and channel (n+1) will assert when CH(n)F is set and will negate when CH(n +1)F is set. The trigger continues to assert on the first counter increment after the pre-trigger asserts and negates at the same time as the pre-trigger negation.

43.5.15 Reset Overview

The TPM is reset whenever any chip reset occurs.

When the TPM exits from reset:

- the TPM counter and the prescaler counter are zero and are stopped (CMOD[1:0] = 0:0);
- the timer overflow interrupt is zero;
- the channels interrupts are zero;
- the channels are in input capture mode;
- the channels outputs are zero;
- the channels pins are not controlled by TPM (ELS(n)B:ELS(n)A = 0:0).

43.5.16 TPM Interrupts

This section describes TPM interrupts.

43.5.16.1 Timer Overflow Interrupt

The timer overflow interrupt is generated when (TOIE = 1) and (TOF = 1).

43.5.16.2 Channel (n) Interrupt

The channel (n) interrupt is generated when (CHnIE = 1) and (CHnF = 1).

Chapter 44
Low-Power Timer (LPTMR)

44.1 Chip-specific LPTMR information

44.1.1 Low power timer instantiations

This device contains 2 LPTMR modules with one 16-bit channel for each module.

LPT/HSCMP0 pulse counting:

LPTMR0_ALT0 input is the selectable source to count pulses resulting from HSCMP0 Output (LPT_ALT0 = HSCMP0 Output).

LPTMR1_ALT0 input can be configured to count the trigger out pulse from LPTMR0, so that LPTMR1 and LPTMR0 can be cascaded as one 32-bit timer.LPTMR0 and LPTMR1 should be configured identically, both connected to HSCMP0 for pulse counter.

44.1.2 LPTMRx 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 LPTMRx_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.

LPTMRx_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.

44.1.3 LPTMRx pulse counter input options

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

LPTMRx_CSR[TPS]	Pulse counter input number	Chip input
00	0	CMP0 output
01	1	LPTMRx_ALT1 pin
10	2	LPTMRx_ALT2 pin
11	3	LPTMRx_ALT3 pin

44.2 Introduction

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.

44.2.1 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

44.2.2 Modes of operation

The following table describes the operation of the LPTMR module in various modes.

Table 44-1. 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.

44.3 LPTMR signal descriptions

Table 44-2. LPTMR signal descriptions

Signal	I/O	Description
LPTMR0_ALT*n*	I	Pulse Counter Input pin

44.3.1 Detailed signal descriptions

Table 44-3. LPTMR interface—detailed signal descriptions

Signal	I/O	Description	
LPTMR_ALT*n*	I	Pulse Counter Input 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.

Table continues on the next page...

Table 44-3. LPTMR interface—detailed signal descriptions (continued)

Signal	I/O	Description	
			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.

44.4 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	44.4.1/1270
4004_0004	Low Power Timer Prescale Register (LPTMR0_PSR)	32	R/W	0000_0000h	44.4.2/1272
4004_0008	Low Power Timer Compare Register (LPTMR0_CMR)	32	R/W	0000_0000h	44.4.3/1273
4004_000C	Low Power Timer Counter Register (LPTMR0_CNR)	32	R/W	0000_0000h	44.4.4/1274
4004_4000	Low Power Timer Control Status Register (LPTMR1_CSR)	32	R/W	0000_0000h	44.4.1/1270
4004_4004	Low Power Timer Prescale Register (LPTMR1_PSR)	32	R/W	0000_0000h	44.4.2/1272
4004_4008	Low Power Timer Compare Register (LPTMR1_CMR)	32	R/W	0000_0000h	44.4.3/1273
4004_400C	Low Power Timer Counter Register (LPTMR1_CNR)	32	R/W	0000_0000h	44.4.4/1274

44.4.1 Low Power Timer Control Status Register (LPTMRx_CSR)

Address: Base address + 0h 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

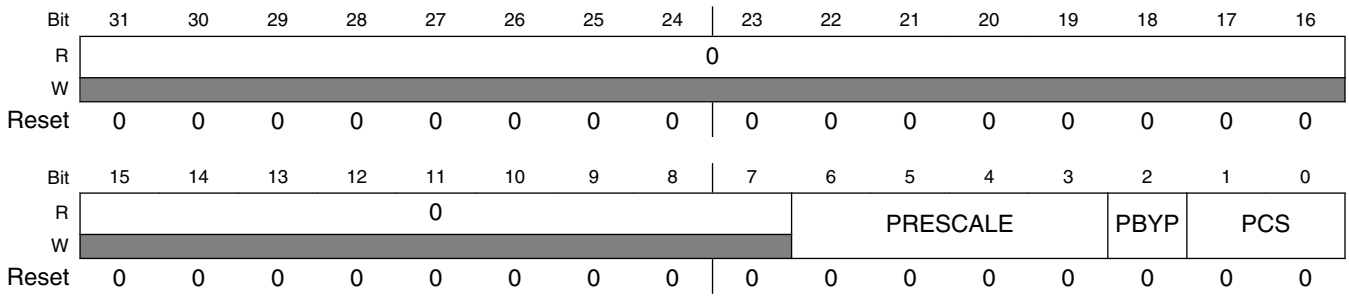
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								TCF	TIE	TPS		TPP	TFC	TMS	TEN
W									w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LPTMRx_CSR field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 TCF	Timer Compare Flag 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. 0 The value of CNR is not equal to CMR and increments. 1 The value of CNR is equal to CMR and increments.
6 TIE	Timer Interrupt Enable When TIE is set, the LPTMR Interrupt is generated whenever TCF is also set. 0 Timer interrupt disabled. 1 Timer interrupt enabled.
5–4 TPS	Timer Pin Select 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-specific LPTMR information for information on the connections to these inputs. 00 Pulse counter input 0 is selected. 01 Pulse counter input 1 is selected. 10 Pulse counter input 2 is selected. 11 Pulse counter input 3 is selected.
3 TPP	Timer Pin Polarity Configures the polarity of the input source in Pulse Counter mode. TPP must be changed only when the LPTMR is disabled. 0 Pulse Counter input source is active-high, and the CNR will increment on the rising-edge. 1 Pulse Counter input source is active-low, and the CNR will increment on the falling-edge.
2 TFC	Timer Free-Running Counter 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. 0 CNR is reset whenever TCF is set. 1 CNR is reset on overflow.
1 TMS	Timer Mode Select Configures the mode of the LPTMR. TMS must be altered only when the LPTMR is disabled. 0 Time Counter mode. 1 Pulse Counter mode.
0 TEN	Timer Enable 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. 0 LPTMR is disabled and internal logic is reset. 1 LPTMR is enabled.

44.4.2 Low Power Timer Prescale Register (LPTMRx_PSR)

Address: Base address + 4h offset



LPTMRx_PSR field descriptions

Field	Description
31–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–3 PRESCALE	Prescale Value 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. 0000 Prescaler divides the prescaler clock by 2; glitch filter does not support this configuration. 0001 Prescaler divides the prescaler clock by 4; glitch filter recognizes change on input pin after 2 rising clock edges. 0010 Prescaler divides the prescaler clock by 8; glitch filter recognizes change on input pin after 4 rising clock edges. 0011 Prescaler divides the prescaler clock by 16; glitch filter recognizes change on input pin after 8 rising clock edges. 0100 Prescaler divides the prescaler clock by 32; glitch filter recognizes change on input pin after 16 rising clock edges. 0101 Prescaler divides the prescaler clock by 64; glitch filter recognizes change on input pin after 32 rising clock edges. 0110 Prescaler divides the prescaler clock by 128; glitch filter recognizes change on input pin after 64 rising clock edges. 0111 Prescaler divides the prescaler clock by 256; glitch filter recognizes change on input pin after 128 rising clock edges. 1000 Prescaler divides the prescaler clock by 512; glitch filter recognizes change on input pin after 256 rising clock edges. 1001 Prescaler divides the prescaler clock by 1024; glitch filter recognizes change on input pin after 512 rising clock edges. 1010 Prescaler divides the prescaler clock by 2048; glitch filter recognizes change on input pin after 1024 rising clock edges. 1011 Prescaler divides the prescaler clock by 4096; glitch filter recognizes change on input pin after 2048 rising clock edges. 1100 Prescaler divides the prescaler clock by 8192; glitch filter recognizes change on input pin after 4096 rising clock edges. 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. 1111 Prescaler divides the prescaler clock by 65,536; glitch filter recognizes change on input pin after 32,768 rising clock edges.
2 PBYP	Prescaler Bypass 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. 0 Prescaler/glitch filter is enabled. 1 Prescaler/glitch filter is bypassed.
PCS	Prescaler Clock Select 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. **NOTE:** See the chip configuration details for information on the connections to these inputs. 00 Prescaler/glitch filter clock 0 selected. 01 Prescaler/glitch filter clock 1 selected. 10 Prescaler/glitch filter clock 2 selected. 11 Prescaler/glitch filter clock 3 selected.

44.4.3 Low Power Timer Compare Register (LPTMRx_CMR)

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																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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COMPARE	Compare Value 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.

44.4.4 Low Power Timer Counter Register (LPTMRx_CNR)

NOTE

See LPTMR counter for details on how to read counter value.

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																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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNTER	Counter Value

44.5 Functional description

44.5.1 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.

44.5.2 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 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.

44.5.3 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.

44.5.3.1 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.

44.5.3.2 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.

44.5.3.3 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.

44.5.3.4 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.

44.5.4 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.

44.5.5 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.

44.5.6 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.

44.5.7 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 45
Periodic Interrupt Timer (PIT)

45.1 Chip-specific PIT information

45.1.1 PIT Instantiations

This device contains one PIT module with four channels.

45.1.2 PIT/DMA Periodic Trigger Assignments

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

Table 45-1. 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

45.1.3 PIT/ADC Triggers

PIT triggers are selected as ADCx trigger sources using the SIM_SOPT7[ADCxTRGSEL] fields. For more details, refer to SIM chapter.

45.2 Introduction

The PIT module is an array of timers that can be used to raise interrupts and trigger DMA channels.

45.2.1 Block diagram

The following figure shows the block diagram of the PIT module.

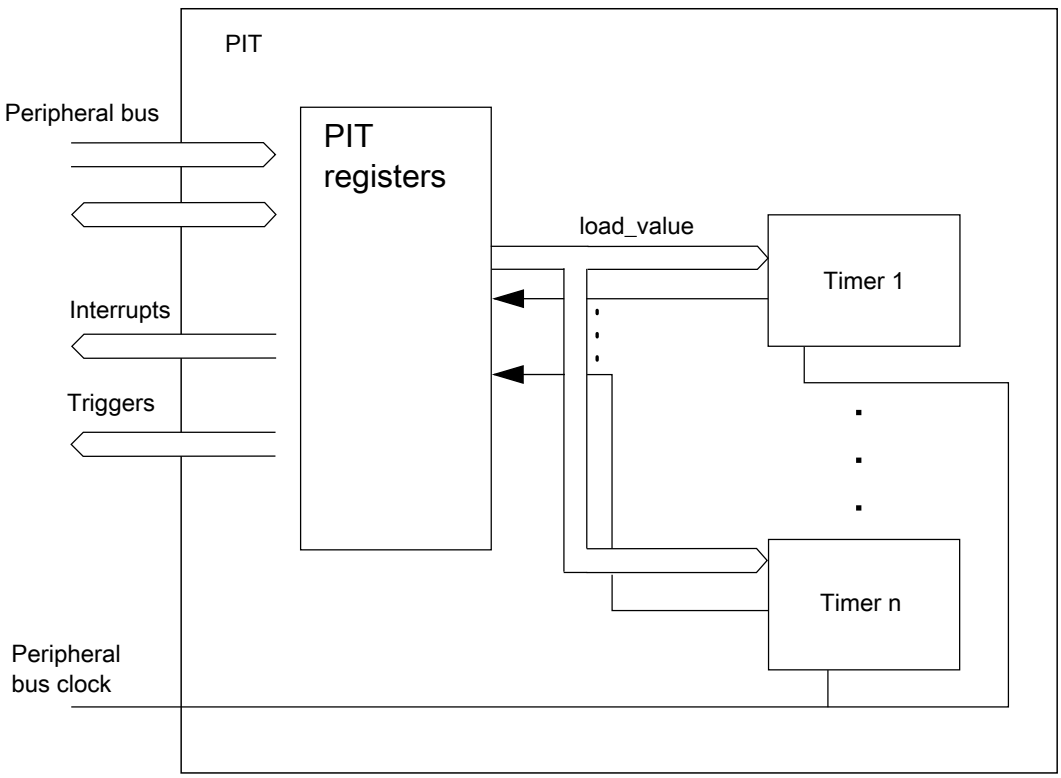


Figure 45-1. Block diagram of the PIT

NOTE

See the chip-specific PIT information for the number of PIT channels used in this MCU.

45.2.2 Features

The main features of this block are:

- Ability of timers to generate DMA trigger pulses

- Ability of timers to generate interrupts
- Maskable interrupts
- Independent timeout periods for each timer

45.3 Signal description

The PIT module has no external pins.

45.4 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-specific PIT information for the number of PIT channels used in this MCU.

PIT memory map

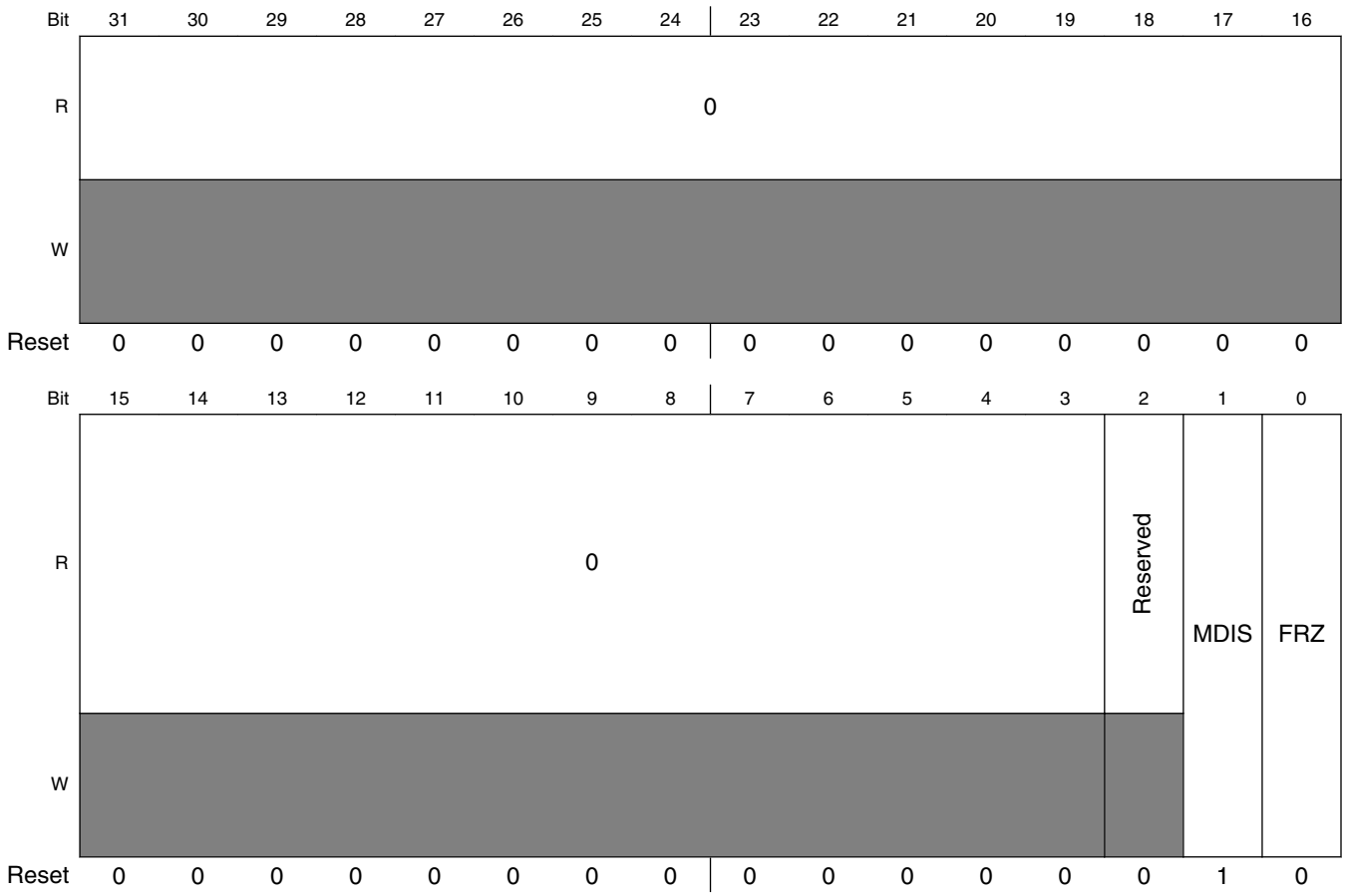
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_7000	PIT Module Control Register (PIT0_MCR)	32	R/W	0000_0002h	45.4.1/1282
4003_70E0	PIT Upper Lifetime Timer Register (PIT0_LTMR64H)	32	R	0000_0000h	45.4.2/1283
4003_70E4	PIT Lower Lifetime Timer Register (PIT0_LTMR64L)	32	R	0000_0000h	45.4.3/1283
4003_7100	Timer Load Value Register (PIT0_LDVAL0)	32	R/W	0000_0000h	45.4.4/1284
4003_7104	Current Timer Value Register (PIT0_CVAL0)	32	R	0000_0000h	45.4.5/1284
4003_7108	Timer Control Register (PIT0_TCTRL0)	32	R/W	0000_0000h	45.4.6/1285
4003_710C	Timer Flag Register (PIT0_TFLG0)	32	R/W	0000_0000h	45.4.7/1286
4003_7110	Timer Load Value Register (PIT0_LDVAL1)	32	R/W	0000_0000h	45.4.4/1284
4003_7114	Current Timer Value Register (PIT0_CVAL1)	32	R	0000_0000h	45.4.5/1284
4003_7118	Timer Control Register (PIT0_TCTRL1)	32	R/W	0000_0000h	45.4.6/1285
4003_711C	Timer Flag Register (PIT0_TFLG1)	32	R/W	0000_0000h	45.4.7/1286
4003_7120	Timer Load Value Register (PIT0_LDVAL2)	32	R/W	0000_0000h	45.4.4/1284
4003_7124	Current Timer Value Register (PIT0_CVAL2)	32	R	0000_0000h	45.4.5/1284
4003_7128	Timer Control Register (PIT0_TCTRL2)	32	R/W	0000_0000h	45.4.6/1285
4003_712C	Timer Flag Register (PIT0_TFLG2)	32	R/W	0000_0000h	45.4.7/1286
4003_7130	Timer Load Value Register (PIT0_LDVAL3)	32	R/W	0000_0000h	45.4.4/1284
4003_7134	Current Timer Value Register (PIT0_CVAL3)	32	R	0000_0000h	45.4.5/1284
4003_7138	Timer Control Register (PIT0_TCTRL3)	32	R/W	0000_0000h	45.4.6/1285
4003_713C	Timer Flag Register (PIT0_TFLG3)	32	R/W	0000_0000h	45.4.7/1286

45.4.1 PIT Module Control Register (PITx_MCR)

This register enables or disables the PIT timer clocks and controls the timers when the PIT enters the Debug mode.

Access: User read/write

Address: 4003_7000h base + 0h offset = 4003_7000h



PITx_MCR field descriptions

Field	Description
31–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 Reserved	This field is reserved.
1 MDIS	Module Disable - (PIT section) Disables the standard timers. This field must be enabled before any other setup is done. 0 Clock for standard PIT timers is enabled. 1 Clock for standard PIT timers is disabled.

Table continues on the next page...

PITx_MCR field descriptions (continued)

Field	Description
0 FRZ	Freeze Allows the timers to be stopped when the device enters the Debug mode. 0 Timers continue to run in Debug mode. 1 Timers are stopped in Debug mode.

45.4.2 PIT Upper Lifetime Timer Register (PITx_LTMR64H)

This register is intended for applications that chain timer 0 and timer 1 to build a 64-bit lifetimer.

Access: User read only

Address: 4003_7000h base + E0h offset = 4003_70E0h

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	LTH																															
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

PITx_LTMR64H field descriptions

Field	Description
LTH	Life Timer value Shows the timer value of timer 1. If this register is read at a time t1, LTMR64L shows the value of timer 0 at time t1.

45.4.3 PIT Lower Lifetime Timer Register (PITx_LTMR64L)

This register is intended for applications that chain timer 0 and timer 1 to build a 64-bit lifetimer.

To use LTMR64H and LTMR64L, timer 0 and timer 1 need to be chained. To obtain the correct value, first read LTMR64H and then LTMR64L. LTMR64H will have the value of CVAL1 at the time of the first access, LTMR64L will have the value of CVAL0 at the time of the first access, therefore the application does not need to worry about carry-over effects of the running counter.

Access: User read only

Address: 4003_7000h base + E4h offset = 4003_70E4h

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	LTL																															
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

PITx_LTMR64L field descriptions

Field	Description
LTL	Life Timer value Shows the value of timer 0 at the time LTMR64H was last read. It will only update if LTMR64H is read.

45.4.4 Timer Load Value Register (PITx_LDVAL*n*)

These registers select the timeout period for the timer interrupts.

Access: User read/write

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 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

PITx_LDVAL*n* field descriptions

Field	Description
TSV	Timer Start Value 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.

45.4.5 Current Timer Value Register (PITx_CVAL*n*)

These registers indicate the current timer position.

Access: User read only

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

PITx_CVALn field descriptions

Field	Description
TVL	Current Timer Value Represents the current timer value, if the timer is enabled. **NOTE:** • If the timer is disabled, do not use this field as its value is unreliable. • The timer uses a downcounter. The timer values are frozen in Debug mode if MCR[FRZ] is set.

45.4.6 Timer Control Register (PITx_TCTRLn)

These registers contain the control bits for each timer.

Access: User read/write

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

PITx_TCTRLn field descriptions

Field	Description
31–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 CHN	Chain Mode When activated, Timer n-1 needs to expire before timer n can decrement by 1. Timer 0 cannot be chained. 0 Timer is not chained. 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 TIE	Timer Interrupt Enable 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. 0 Interrupt requests from Timer n are disabled. 1 Interrupt will be requested whenever TIF is set.
0 TEN	Timer Enable Enables or disables the timer.

Table continues on the next page...

PITx_TCTRL*n* field descriptions (continued)

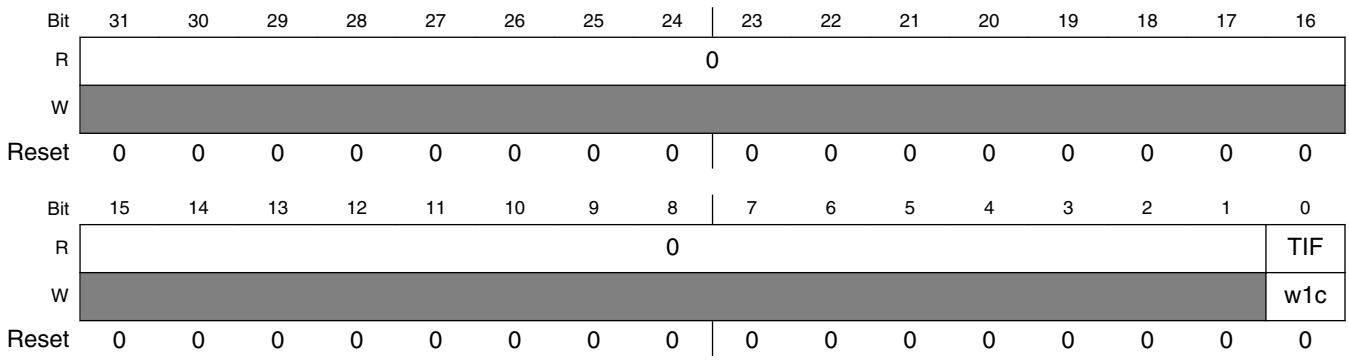
Field	Description
	0 Timer n is disabled. 1 Timer n is enabled.

45.4.7 Timer Flag Register (PITx_TFLG*n*)

These registers hold the PIT interrupt flags.

Access: User read/write

Address: 4003_7000h base + 10Ch offset + (16d × i), where i=0d to 3d



PITx_TFLG*n* field descriptions

Field	Description
31–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 TIF	Timer Interrupt Flag 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. 0 Timeout has not yet occurred. 1 Timeout has occurred.

45.5 Functional description

This section provides the functional description of the module.

45.5.1 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.

45.5.1.1 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.

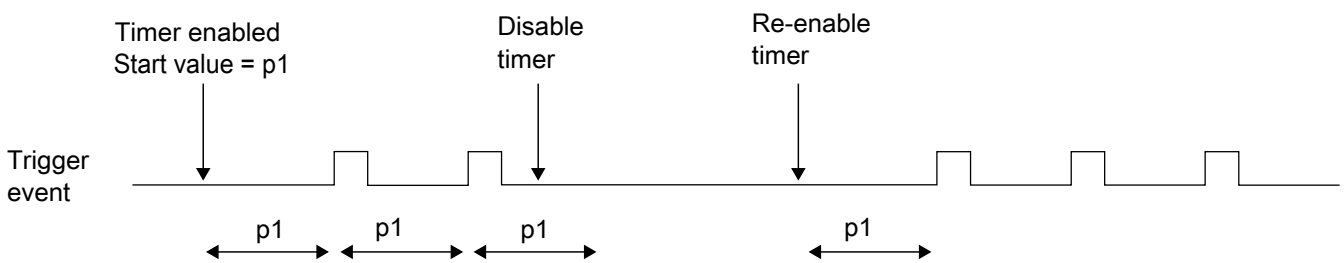


Figure 45-2. Stopping and starting a timer

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.

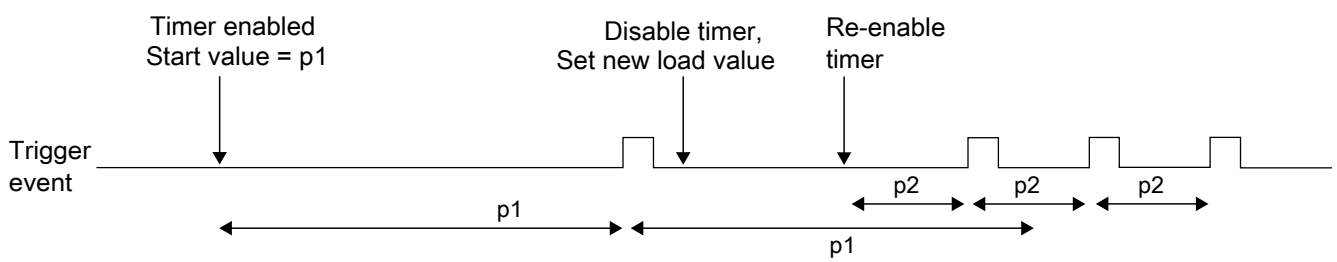


Figure 45-3. Modifying running timer period

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.

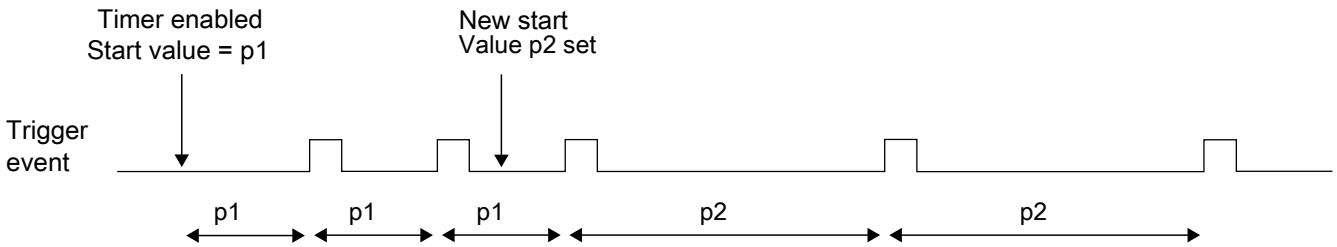


Figure 45-4. Dynamically setting a new load value

45.5.1.2 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.

45.5.2 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].

45.5.3 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.

45.6 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
```

45.7 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 minute.

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
```

45.8 Example configuration for the lifetime timer

To configure the lifetimer timer, channels 0 and 1 need to be chained together.

First the PIT module needs to be activated by writing a 0 to the MDIS bit in the CTRL register, then the LDVAL registers need to be set to the maximum value.

The timer is a downcounter.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// Timer 1
PIT_LDVAL1 = 0xFFFFFFFF; // setup timer 1 for maximum counting period
PIT_TCTRL1 = 0x0; // disable timer 1 interrupts
PIT_TCTRL1 |= CHN; // chain timer 1 to timer 0
PIT_TCTRL1 |= TEN; // start timer 1


// Timer 0
```

```
PIT_LDVAL0 = 0xFFFFFFFF; // setup timer 0 for maximum counting period
PIT_TCTRL0 = TEN; // start timer 0
```

To access the lifetime, read first LTMR64H and then LTMR64L.

```
current_uptime = PIT_LTMR64H<<32;
current_uptime = current_uptime + PIT_LTMR64L;
```

Chapter 46
Real Time Clock (RTC)

46.1 Chip-specific RTC information

46.1.1 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.

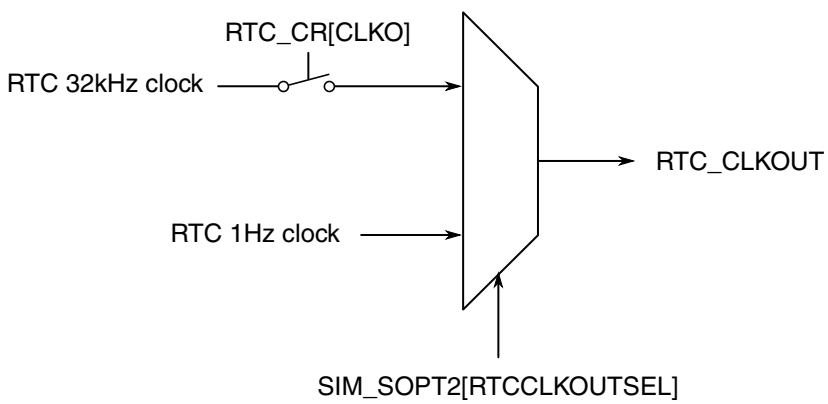


Figure 46-1. RTC_CLKOUT generation

NOTE

It is possible on variants of this device with RTC_WAKEUP pin to output the RTC clock signal to the RTC_WAKEUP pin. Please refer to the RTC Control Register for details.

46.2 Introduction

46.2.1 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 with optional interrupt

46.2.2 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.

46.2.3 RTC signal descriptions

Table 46-1. 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 or OSCERCLK	O
$\overline{\text{RTC\_WAKEUP}}$	Wakeup for external device	O

46.2.3.1 RTC clock output

The clock to the seconds counter is available on the RTC_CLKOUT signal. It is a 1 Hz square wave output. See RTC_CLKOUT options for details.

46.2.3.2 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.

NOTE

The wakeup pin is optional and may not be implemented on all devices.

46.3 Register definition

All registers must be accessed using 32-bit writes and all register accesses incur three wait states.

Write accesses to any register by non-supervisor mode software, when the supervisor access bit in the control register is clear, will terminate with a bus error.

Read accesses 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	46.3.1/1296
4003_D004	RTC Time Prescaler Register (RTC_TPR)	32	R/W	0000_0000h	46.3.2/1296
4003_D008	RTC Time Alarm Register (RTC_TAR)	32	R/W	0000_0000h	46.3.3/1297
4003_D00C	RTC Time Compensation Register (RTC_TCR)	32	R/W	0000_0000h	46.3.4/1297

Table continues on the next page...

RTC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_D010	RTC Control Register (RTC_CR)	32	R/W	0000_0000h	46.3.5/1299
4003_D014	RTC Status Register (RTC_SR)	32	R/W	0000_0001h	46.3.6/1301
4003_D018	RTC Lock Register (RTC_LR)	32	R/W	0000_00FFh	46.3.7/1302
4003_D01C	RTC Interrupt Enable Register (RTC_IER)	32	R/W	0000_0007h	46.3.8/1303
4003_D800	RTC Write Access Register (RTC_WAR)	32	R/W	0000_00FFh	46.3.9/1304
4003_D804	RTC Read Access Register (RTC_RAR)	32	R/W	0000_00FFh	46.3.10/ 1305

46.3.1 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 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
TSR	Time Seconds Register 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).

46.3.2 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

RTC_TPR field descriptions (continued)

Field	Description
TPR	Time Prescaler Register 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.

46.3.3 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 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
TAR	Time Alarm Register 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].

46.3.4 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 CIC	Compensation Interval Counter 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 TCV	Time Compensation Value 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).

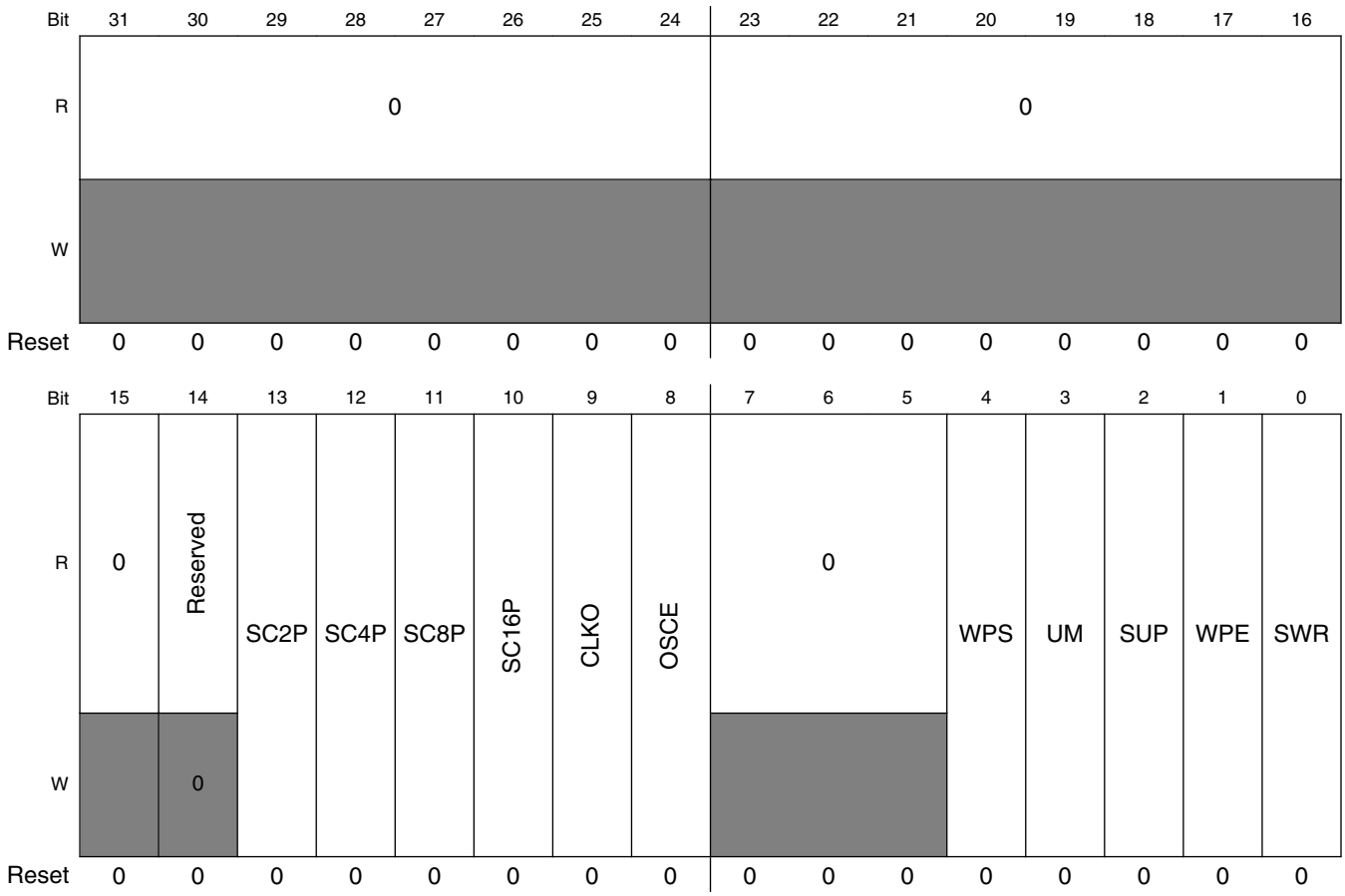
Table continues on the next page...

RTC_TCR field descriptions (continued)

Field	Description
15–8 CIR	Compensation Interval Register 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.
TCR	Time Compensation Register 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. 80h Time Prescaler Register overflows every 32896 clock cycles. ... ... FFh Time Prescaler Register overflows every 32769 clock cycles. 00h Time Prescaler Register overflows every 32768 clock cycles. 01h Time Prescaler Register overflows every 32767 clock cycles. .... 7Fh Time Prescaler Register overflows every 32641 clock cycles.

46.3.5 RTC Control Register (RTC_CR)

Address: 4003_D000h base + 10h offset = 4003_D010h



RTC_CR field descriptions

Field	Description
31–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 Reserved	This field is reserved. It must always be written to 0.
13 SC2P	Oscillator 2pF Load Configure 0 Disable the load. 1 Enable the additional load.
12 SC4P	Oscillator 4pF Load Configure 0 Disable the load. 1 Enable the additional load.

Table continues on the next page...

RTC_CR field descriptions (continued)

Field	Description
11 SC8P	Oscillator 8pF Load Configure 0 Disable the load. 1 Enable the additional load.
10 SC16P	Oscillator 16pF Load Configure 0 Disable the load. 1 Enable the additional load.
9 CLKO	Clock Output 0 The 32 kHz clock is output to other peripherals. 1 The 32 kHz clock is not output to other peripherals.
8 OSCE	Oscillator Enable 0 32.768 kHz oscillator is disabled. 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 WPS	Wakeup Pin Select The wakeup pin is optional and not available on all devices. 0 Wakeup pin asserts (active low, open drain) if the RTC interrupt asserts or the wakeup pin is turned on. 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 UM	Update Mode 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. 0 Registers cannot be written when locked. 1 Registers can be written when locked under limited conditions.
2 SUP	Supervisor Access 0 Non-supervisor mode write accesses are not supported and generate a bus error. 1 Non-supervisor mode write accesses are supported.
1 WPE	Wakeup Pin Enable The wakeup pin is optional and not available on all devices. 0 Wakeup pin is disabled. 1 Wakeup pin is enabled and wakeup pin asserts if the RTC interrupt asserts or the wakeup pin is turned on.
0 SWR	Software Reset 0 No effect. 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.

46.3.6 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

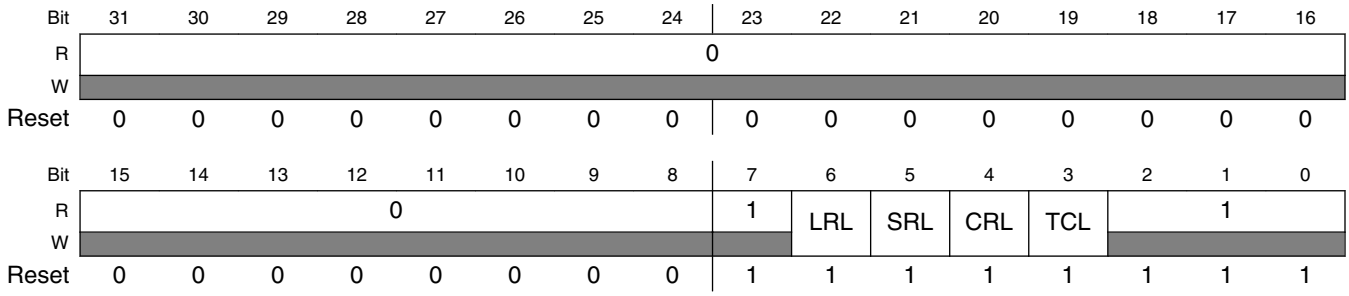
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0											TCE	0	TAF	TOF	TIF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

RTC_SR field descriptions

Field	Description
31–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 TCE	Time Counter Enable 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. 0 Time counter is disabled. 1 Time counter is enabled.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 TAF	Time Alarm Flag 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. 0 Time alarm has not occurred. 1 Time alarm has occurred.
1 TOF	Time Overflow Flag 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. 0 Time overflow has not occurred. 1 Time overflow has occurred and time counter is read as zero.
0 TIF	Time Invalid Flag 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. 0 Time is valid. 1 Time is invalid and time counter is read as zero.

46.3.7 RTC Lock Register (RTC_LR)

Address: 4003_D000h base + 18h offset = 4003_D018h

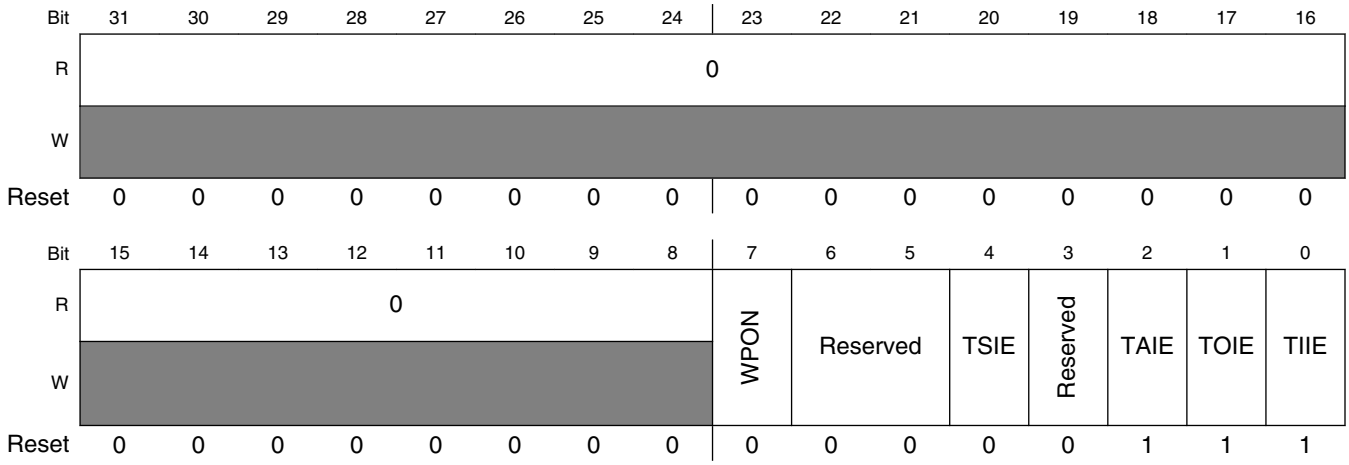


RTC_LR field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
6 LRL	Lock Register Lock After being cleared, this bit can be set only by VBAT POR or software reset. 0 Lock Register is locked and writes are ignored. 1 Lock Register is not locked and writes complete as normal.
5 SRL	Status Register Lock After being cleared, this bit can be set only by VBAT POR or software reset. 0 Status Register is locked and writes are ignored. 1 Status Register is not locked and writes complete as normal.
4 CRL	Control Register Lock After being cleared, this bit can only be set by VBAT POR. 0 Control Register is locked and writes are ignored. 1 Control Register is not locked and writes complete as normal.
3 TCL	Time Compensation Lock After being cleared, this bit can be set only by VBAT POR or software reset. 0 Time Compensation Register is locked and writes are ignored. 1 Time Compensation Register is not locked and writes complete as normal.
Reserved	This field is reserved. This read-only field is reserved and always has the value 1.

46.3.8 RTC Interrupt Enable Register (RTC_IER)

Address: 4003_D000h base + 1Ch offset = 4003_D01Ch



RTC_IER field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 WPON	Wakeup Pin On 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. 0 No effect. 1 If the wakeup pin is enabled, then the wakeup pin will assert.
6–5 Reserved	This field is reserved.
4 TSIE	Time Seconds Interrupt Enable 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). 0 Seconds interrupt is disabled. 1 Seconds interrupt is enabled.
3 Reserved	This field is reserved.
2 TAIE	Time Alarm Interrupt Enable 0 Time alarm flag does not generate an interrupt. 1 Time alarm flag does generate an interrupt.
1 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. 1 Time overflow flag does generate an interrupt.
0 TIIE	Time Invalid Interrupt Enable 0 Time invalid flag does not generate an interrupt. 1 Time invalid flag does generate an interrupt.

46.3.9 RTC Write Access Register (RTC_WAR)

Address: 4003_D000h base + 800h offset = 4003_D800h

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								IERW	LRW	SRW	CRW	TCRW	TARW	TPRW	TSRW
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

RTC_WAR field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 IERW	Interrupt Enable Register Write After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset. 0 Writes to the Interupt Enable Register are ignored. 1 Writes to the Interrupt Enable Register complete as normal.
6 LRW	Lock Register Write After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset. 0 Writes to the Lock Register are ignored. 1 Writes to the Lock Register complete as normal.
5 SRW	Status Register Write 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 Status Register are ignored. 1 Writes to the Status Register complete as normal.
4 CRW	Control Register Write After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset. 0 Writes to the Control Register are ignored. 1 Writes to the Control Register complete as normal.
3 TCRW	Time Compensation Register Write After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset. 0 Writes to the Time Compensation Register are ignored. 1 Writes to the Time Compensation Register complete as normal.
2 TARW	Time Alarm Register Write After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset. 0 Writes to the Time Alarm Register are ignored. 1 Writes to the Time Alarm Register complete as normal.
1 TPRW	Time Prescaler Register Write After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset. 0 Writes to the Time Prescaler Register are ignored. 1 Writes to the Time Prescaler Register complete as normal.
0 TSRW	Time Seconds Register Write After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset. 0 Writes to the Time Seconds Register are ignored. 1 Writes to the Time Seconds Register complete as normal.

46.3.10 RTC Read Access Register (RTC_RAR)

Address: 4003_D000h base + 804h offset = 4003_D804h

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								IERR	LRR	SRR	CRR	TCRR	TARR	TPRR	TSRR
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

RTC_RAR field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 IERR	Interrupt Enable Register Read After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset. 0 Reads to the Interrupt Enable Register are ignored. 1 Reads to the Interrupt Enable Register complete as normal.
6 LRR	Lock Register Read After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset. 0 Reads to the Lock Register are ignored. 1 Reads to the Lock Register complete as normal.
5 SRR	Status Register Read After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset. 0 Reads to the Status Register are ignored. 1 Reads to the Status Register complete as normal.
4 CRR	Control Register Read After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset. 0 Reads to the Control Register are ignored. 1 Reads to the Control Register complete as normal.
3 TCRR	Time Compensation Register Read After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset. 0 Reads to the Time Compensation Register are ignored. 1 Reads to the Time Compensation Register complete as normal.
2 TARR	Time Alarm Register Read After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset. 0 Reads to the Time Alarm Register are ignored. 1 Reads to the Time Alarm Register complete as normal.
1 TPRR	Time Prescaler Register Read After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset. 0 Reads to the Time Pprescaler Register are ignored. 1 Reads to the Time Prescaler Register complete as normal.
0 TSRR	Time Seconds Register Read After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset. 0 Reads to the Time Seconds Register are ignored. 1 Reads to the Time Seconds Register complete as normal.

46.4 Functional description

46.4.1 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.

NOTE

An attempt to access an RTC register, except the access control registers, results in a bus error when:

- VBAT is powered down,
- the RTC is electrically isolated, or
- VBAT POR is asserted.

46.4.1.1 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.

46.4.1.2 Software reset

Writing 1 to CR[SWR] forces the equivalent of a VBAT POR to the rest of the RTC module. 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.

46.4.1.3 Supervisor access

When the supervisor access control bit is clear, only supervisor mode software can write to the RTC registers, non-supervisor mode software will generate a bus error. Both supervisor and non-supervisor mode software can always read the RTC registers.

46.4.2 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.

Reading the time counter (either seconds or prescaler) while it is incrementing may return invalid data due to synchronization of the read data bus. If it is necessary for software to read the prescaler or seconds counter when they could be incrementing, it is recommended that two read accesses are performed and that software verifies that the same data was returned for both reads.

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.

46.4.3 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.

46.4.4 Time alarm

The Time Alarm register (TAR), SR[TAF], and IER[TAIE] allow the RTC to generate an interrupt at a predefined time. The 32-bit TAR is compared with the 32-bit Time Seconds register (TSR) each time it increments. SR[TAF] will set when TAR equals TSR and TSR increments.

SR[TAF] is cleared by writing TAR. This will usually be the next alarm value, although writing a value that is less than TSR, such as 0, will prevent SR[TAF] from setting again. SR[TAF] cannot otherwise be disabled, although the interrupt it generates is enabled or disabled by IER[TAIE].

46.4.5 Update mode

The Update Mode field in the Control register (CR[UM]) configures software write access to the Time Counter Enable (SR[TCE]) field. When CR[UM] is clear, SR[TCE] can be written only when LR[SRL] is set. When CR[UM] is set, 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].

46.4.6 Register lock

The Lock register (LR) can be used to block write accesses to certain registers until the next VBAT POR or software reset. Locking the Control register (CR) will disable the software reset. Locking LR will block future updates to LR.

Write accesses to a locked register are ignored and do not generate a bus error.

46.4.7 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.

46.4.8 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 47
Universal Serial Bus Full Speed OTG Controller (USBFSOTG)

47.1 Chip-specific USBFSOTG information

47.1.1 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.
- IRC48 with clock recovery block to eliminate the 48MHz crystal. This is available for USB device mode only.
- A configurable connection to allow LPUART2 transmit and receive pins to be connected to the Full Speed USB physical layer.

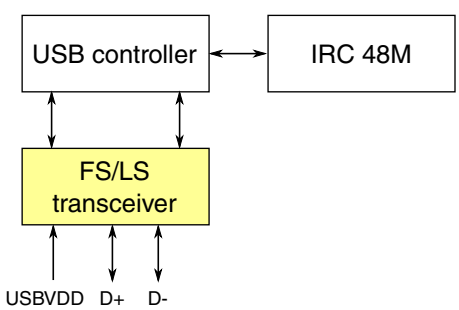


Figure 47-1. USB FS/LS Subsystem Overview

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.

47.1.2 FS/LS USB OTG Instantiation

The device does not contain a separate VBUS detect signal. To detect valid VBUS in device mode use a GPIO.

The USB transceiver includes 15k pulldowns on the D+ and D- lines for host mode functionality.

47.1.2.1 Wake-up functionality on USB

Following wake-up functionality is supported on USB:

- Wake on VBUS detect in all power modes. There is no explicit pin for VBUS detect, but you can use a GPIO for this function.
- Any activity on DP/DM wakes the system from low power mode, except in LLS/VLLS mode where USB is not powered.

47.1.3 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.

The following wakeup functionality is also supported:

- In LLS/VLLS, USB0_DP and USB0_DM are input pins to the LLWU and a transition on those pins can generate a wakeup provided the USB is powered and in host mode.

47.1.4 LPUART over USB capability

This device supports a connection whereby LPUART2 may be connected to the FS USB physical layer (and the USB controller disconnected) to allow simple debugging capabilities for applications which do not have direct physical access to LPUART2. For more details on the control bits, please refer to Introduction.

By setting the USBx_USBCTRL[UARTSEL] bit, LPUART2 will be connected to the FS USB physical layer. This configures the FS USB DP/DM signals to be configured as normal LPUART2 signals, which do not operate in differential mode. When USBx_USBCTRL[UARTCHLS] bit is set to 1b, FS USB DP/DM signal pins will be used as LPUART2 TX/RX. When USBx_USBCTRL[UARTCHLS] bit is set to 1b, FS USB DP/DM signal pins will be used as LPUART2 RX/TX.

47.1.5 USB Power Distribution

This chip includes a separate USBVDD supply pad that powers the USB transceiver. USBVDD can be powered at nominal USB voltage level (3.3V) while the main MCU supply pins are powered across the full operating range for the device.

47.1.5.1 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 off-chip regulator is enabled to power the USB transceiver.

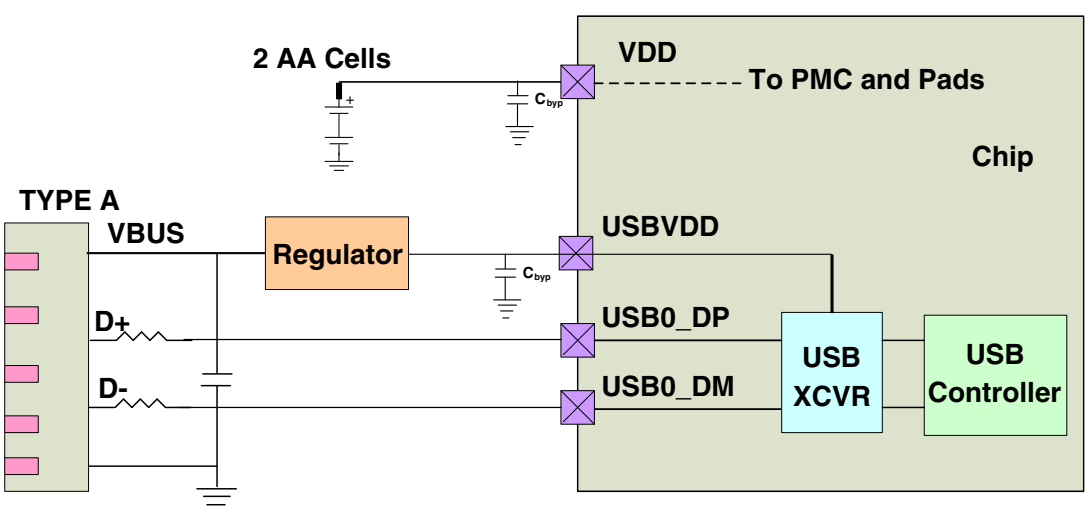


Figure 47-2. USB regulator AA cell usecase

47.1.5.2 Li-Ion battery power supply

The chip can also be powered by a single Li-ion battery. In this case, the regulator supplies VDD and USBVDD supply inputs. When connected to a USB host, the input source of this regulator is switched to the USB bus supply from the Li-ion battery.

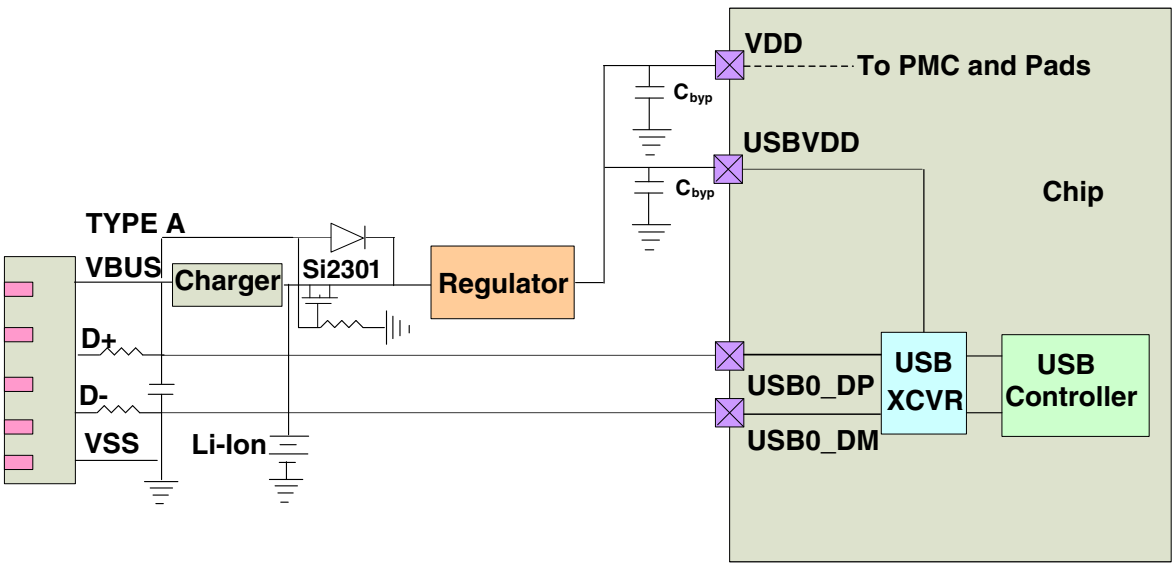


Figure 47-3. USB regulator Li-ion usecase

47.1.5.3 USB bus power supply

The chip can also be powered by the USB bus directly. In this case, the regulator supplies VDD and USBVDD supply inputs.

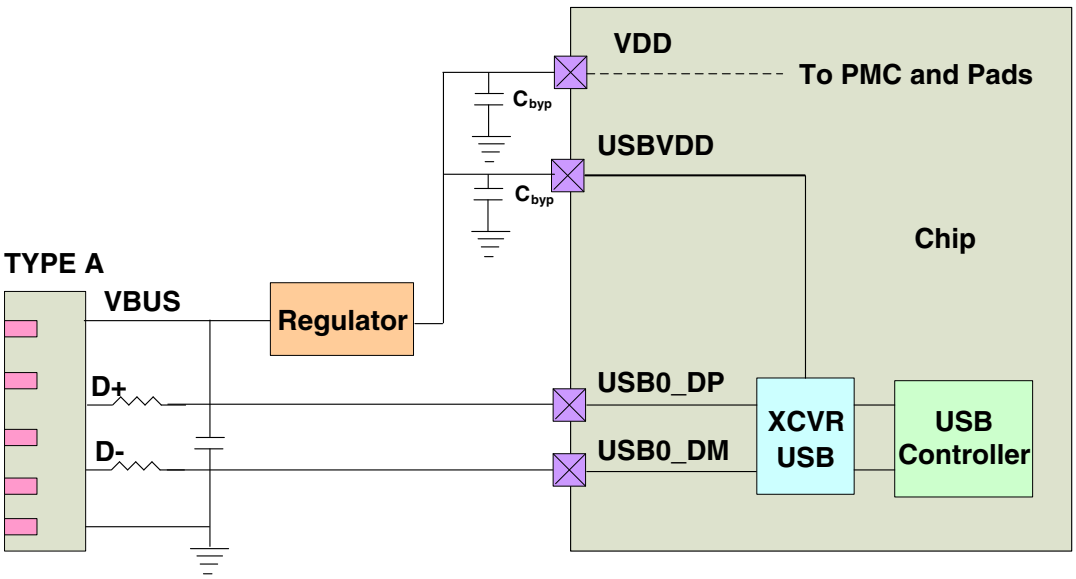


Figure 47-4. USB regulator bus supply

47.1.6 USB controller configuration

NOTE

When USB is not used in the application, it is recommended that the USB supply pad USBVDD remain floating.

47.2 Introduction

This chapter describes the USB full speed OTG controller. 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 logic implements features required by the *On-The-Go and Embedded Host Supplement to the USB 2.0 Specification* (usb.org, 2008). The USB full speed controller interfaces to a USBFS/LS transceiver.

NOTE

This chapter describes the following registers that have similar names: USB_OTGCTL, USB_CTL, USB_CTRL, and USB_CONTROL. These are all separate registers.

47.2.1 References

The following publications are referenced in this document. For copies or updates to these specifications, see the USB Implementers Forum, Inc. website at http://www.usb.org.

- *Universal Serial Bus Specification, Revision 2.0* , 2000, with amendments including those listed below
- *Errata for "USB Revision 2.0 April 27, 2000" as of 12/7/2000*
- *Errata for "USB Revision 2.0 April 27, 2000" as of 12/7/2000*
- *Pull-up / Pull-down Resistors* (USB Engineering Change Notice)
- *Suspend Current Limit Changes* (USB Engineering Change Notice)
- *Device Capacitance* (USB Engineering Change Notice)
- *USB 2.0 Connect Timing Update* (USB Engineering Change Notice as of April 4, 2013)
- *USB 2.0 VBUS Max Limit* (USB Engineering Change Notice)
- *On-The-Go and Embedded Host Supplement to the USB Revision 2.0 Specification*, Revision 2.0 version 1.1a, July 27, 2012
- *Maximum VBUS Voltage* (USB OTGEH Engineering Change Notice)
- *Universal Serial Bus Micro-USB Cables and Connectors Specification*, Revision 1.01, 2007

47.2.2 USB—Overview

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 to make 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 human interface devices, 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.

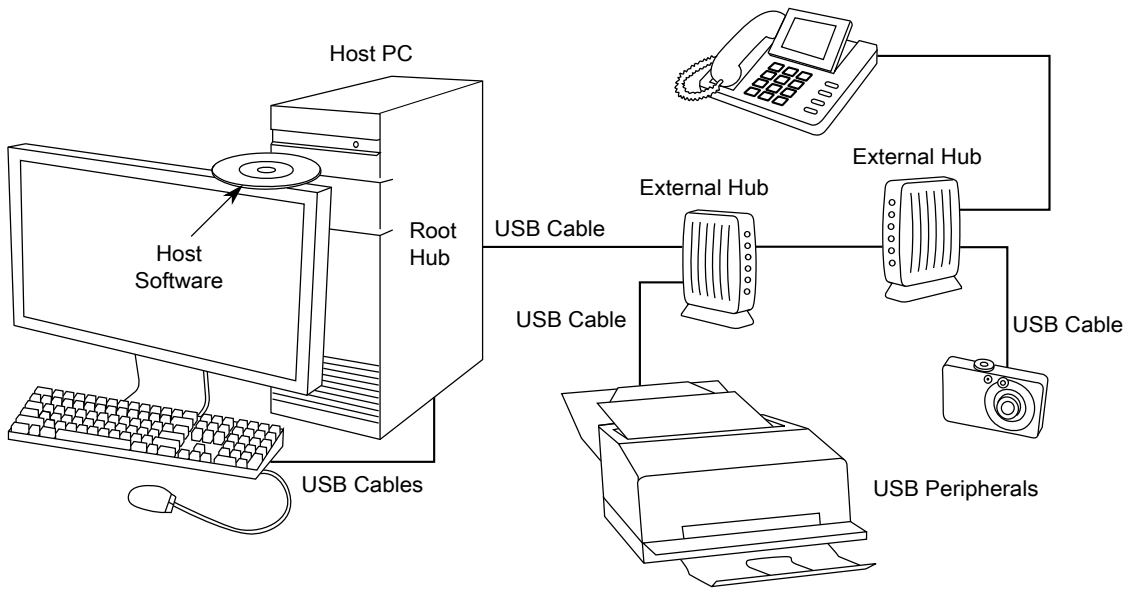


Figure 47-5. Example USB 2.0 system configuration

47.2.3 USB On-The-Go

USB is a popular standard for connecting peripherals and portable consumer electronic devices such as digital cameras and tablets 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 tablet 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 and Embedded Host Supplement to the USB 2.0 Specification*.

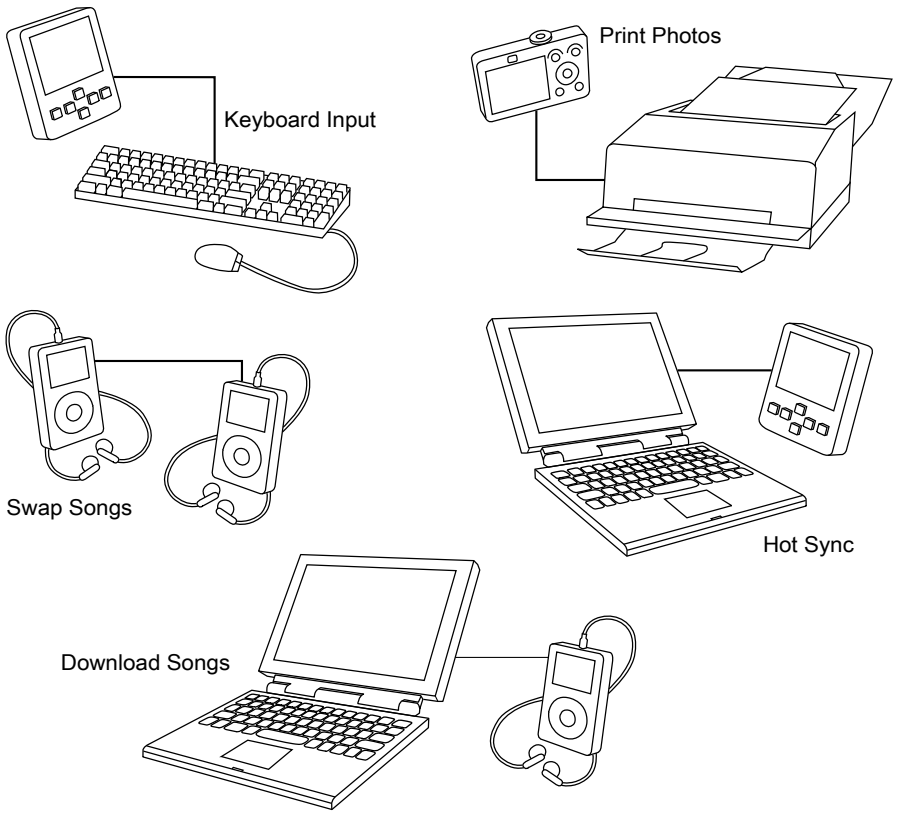


Figure 47-6. Example USB 2.0 On-The-Go configurations

47.2.4 USBFS Features

- USB 1.1 and 2.0 compatible FS device and FS/LS host controller with On-The-Go protocol logic
- 16 bidirectional endpoints
- DMA or FIFO data stream interfaces
- Low-power consumption
- IRC48M with clock-recovery is supported to eliminate the 48 MHz crystal. It is used for USB device-only implementation.
- Keep-alive feature is supported to power down system bus and CPU. USB can respond to IN with NAK and wake up for SETUP/OUT.

47.3 Functional description

USBOTG communicates with the processor core through status registers, control registers, and data structures in memory.

47.3.1 Data Structures

To efficiently manage USB endpoint communications, USBFS implements a Buffer Descriptor Table (BDT) in system memory. See Figure 47-10.

47.3.2 On-chip transceiver required external components

USB system operation requires external components to ensure that driver output impedance, eye diagram, and VBUS cable fault tolerance requirements are met. DM and DP I/O pads must connect through series resistors (approximately 33 Ω each) to the USB connector on the application printed circuit board (PCB). Additionally, signal quality optimizes when these 33 Ω resistors are mounted closer to the processor than to the USB board-level connector. The USB transceiver includes:

- An internal 1.5 kΩ pullup resistor on the USB_DP line for full-speed device (controlled by USB_CONTROL[DPPULLUPNONOTG] or USB_OTGCTL[DPHIGH])
- Internal 15 kΩ pulldown resistors on the USB_DP and USB_DM signals, which are primarily intended for Host mode operation but are also useful in Device mode as explained below.

NOTE

For device operation, the internal 15 kΩ pulldowns should be enabled to keep the DP and DM ports in a known quiescent state when the VBUS detection software determines that the USB connection is not activated, including the cases when no cable to a host is present or when the USB port is not used. The internal 15 kΩ pulldowns should be controlled by USB_CTRL[PDE] in this case.

For host operation, the internal 15 kΩ pulldowns are enabled automatically, as required by the USB 2.0 specification, when USB_CTL[HOSTMODEEN] is asserted high.

The following diagrams provide overviews of host-only, device-only, and dual-role connections, respectively. For more details, see the *Kinetis Peripheral Module Quick Reference*(KQRUG).

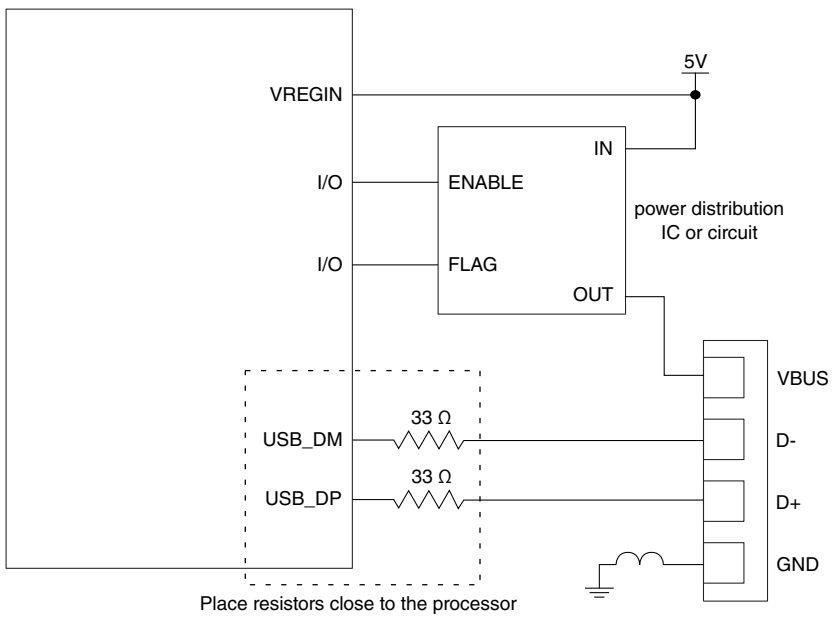


Figure 47-7. Host-only diagram

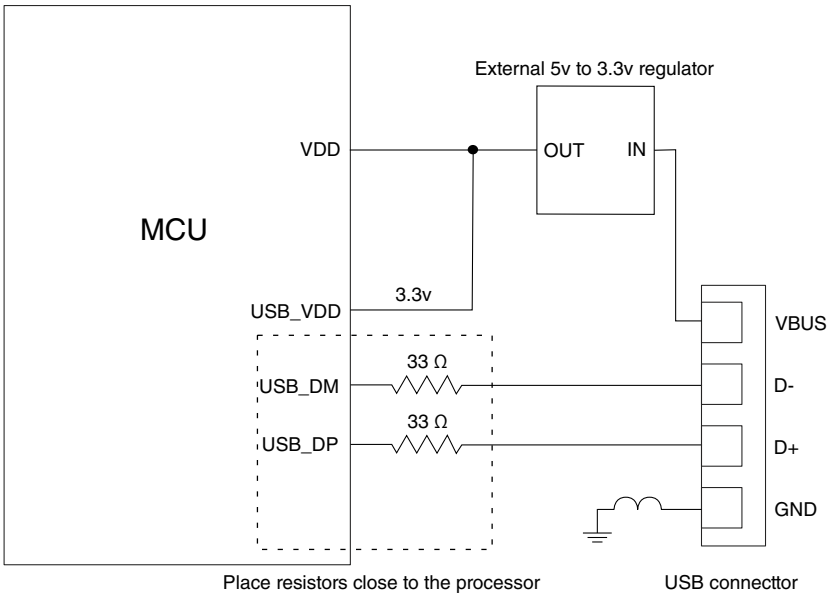


Figure 47-8. Typical Device-only block diagram (bus-powered with external regulator)

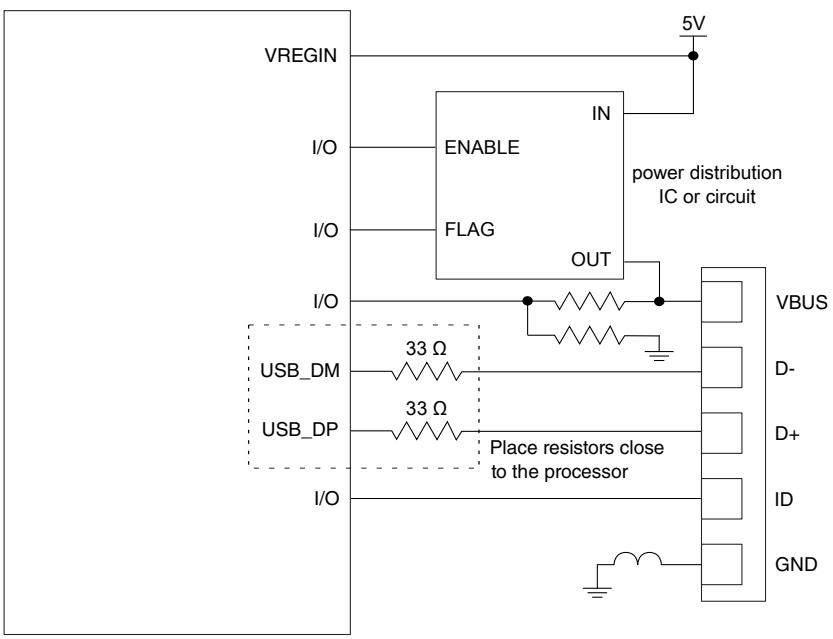


Figure 47-9. Dual-role diagram

47.4 Programmers interface

This section discusses the major components of the programming model for the USB module.

47.4.1 Buffer Descriptor Table

To efficiently manage USB endpoint communications USBFS 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 USBFS is processing the other BD. Double buffering BDs in this way allows USBFS to transfer data easily at the maximum throughput provided by USB.

Software should manage buffers for USBFS by updating the BDT when needed. This allows USBFS 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 USBFS, 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 USBFS. USBFS 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.

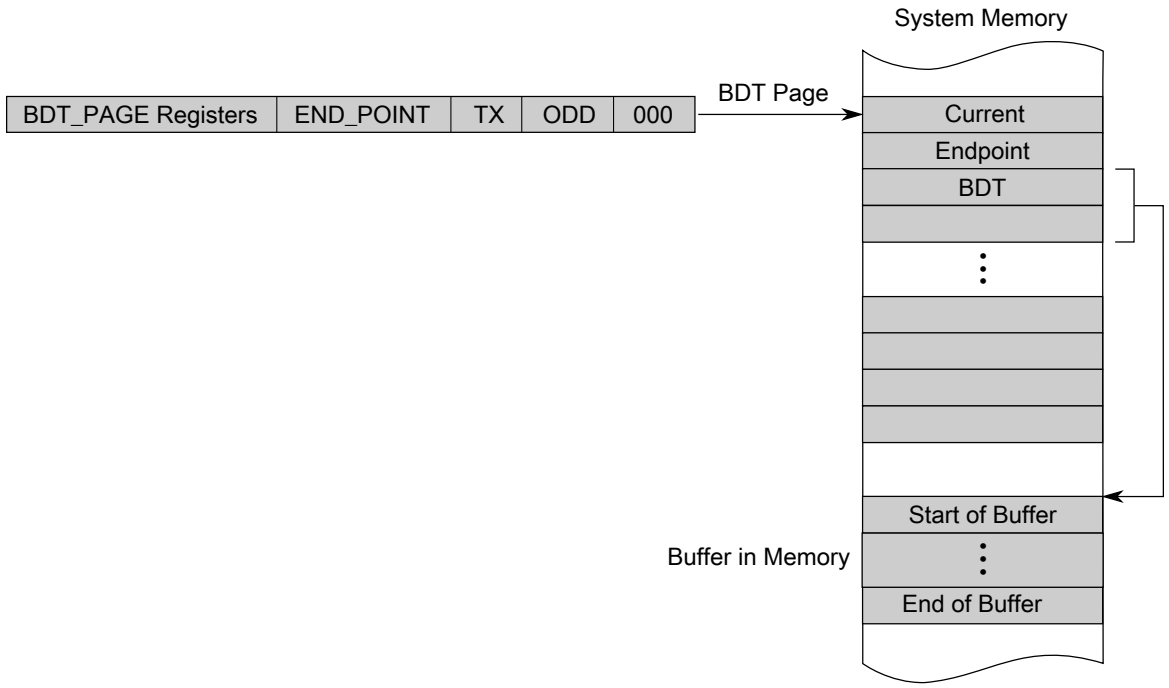


Figure 47-10. Buffer descriptor table

47.4.2 RX vs. TX as a USB peripheral device or USB host

The USBFS core uses software control to switch between two modes of operation:

- USB peripheral device
- USB hosts

In either mode, USB host or USB peripheral device, the same data paths and buffer descriptors are used for the transmission and reception of data. For this reason, a USBFS core-centric nomenclature is used to describe the direction of the data transfer between the USBFS core and USB:

- "RX" (or "receive") describes transfers that move data from USB to memory.
- "TX" (or "transmit") describes transfers that move data from memory to USB.

The following table shows how the data direction corresponds to the USB token type in host and peripheral device applications.

Table 47-1. Data direction for USB host or USB peripheral device

	RX	TX
Device	OUT or SETUP	IN
Host	IN	OUT or SETUP

47.4.3 Addressing BDT entries

An understanding of the addressing mechanism of the Buffer Descriptor Table is useful when accessing endpoint data via USBFS 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 fewer 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 USBFS or MCU core.

When a USB token on an enabled endpoint is received, USBFS uses its integrated DMA controller to interrogate the BDT. USBFS reads the corresponding endpoint BD entry to determine whether it owns the BD and corresponding buffer in system memory.

To compute the entry point into the BDT, the BDT_PAGE registers are concatenated with the current endpoint and the TX and ODD fields to form a 32-bit address. This address mechanism is shown in the following tables:

Table 47-2. BDT Address Calculation

31:24	23:16	15:9	8:5	4	3	2:0
BDT_PAGE_03	BDT_PAGE_02	BDT_PAGE_01[7:1]	Endpoint	TX	ODD	000

Table 47-3. BDT address calculation fields

Field	Description
BDT_PAGE	BDT_PAGE registers in the Control Register Block
ENDPOINT	ENDPOINT field from the USB TOKEN
TX	1 for transmit transfers and 0 for receive transfers
ODD	Maintained within the USBFS SIE. It corresponds to the buffer currently in use. The buffers are used in a ping-pong fashion.

47.4.4 Buffer Descriptors (BDs)

A buffer descriptor provides endpoint buffer control information for USBFS and the processor. The Buffer Descriptors have different meaning based on whether it is USBFS or the processor reading the BD in memory.

The USBFS 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 47-4. 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 47-5. Buffer descriptor fields

Field	Description
31–26 RSVD	Reserved
25–16 BC	Byte Count Represents the 10-bit byte count. The USBFS SIE changes this field upon the completion of a RX transfer with the byte count of the data received.
15–8 RSVD	Reserved

Table continues on the next page...

Table 47-5. Buffer descriptor fields (continued)

Field	Description
7 OWN	Determines whether the processor or USBFS currently owns the buffer. Except when KEEP=1, the SIE hands ownership back to the processor after completing the token by clearing this bit. This must always be the last byte of the BD that the processor updates when it initializes a BD. 0 The processor has access to the BD. USBFS ignores all other fields in the BD. 1 USBFS has access to the BD. While USBFS owns the BD, the processor should not modify any other fields in the BD.
6 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 USBFS.
5 KEEP/ TOK_PID[3]	This bit has two functions: • KEEP bit—When written by the processor, it serves as the KEEP bit. 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. 0 Allows USBFS to release the BD when a token has been processed. 1 This bit is unchanged by USBFS. Bit 3 of the current token PID is written back to the BD by USBFS. • TOK_PID[3]—If the OWN bit is also set, the BD remains owned by USBFS indefinitely; when written by USB, it serves as the TOK_PID[3] bit. 0 or 1 Bit 3 of the current token PID is written back to the BD by USBFS. 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.
4 NINC/ TOK_PID[2]	No Increment (NINC) 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. 0 USBFS writes bit 2 of the current token PID to the BD. 1 This bit is unchanged by USBFS.
3 DTS/ TOK_PID[1]	Setting this bit enables USBFS to perform Data Toggle Synchronization. • If KEEP=0, bit 1 of the current token PID is written back to the BD. • If KEEP=1, this bit is unchanged by USBFS. 0 Data Toggle Synchronization is disabled. 1 Enables USBFS to perform Data Toggle Synchronization.
2 BDT_STALL TOK_PID[0]	Setting this bit causes USBFS 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. • If KEEP=0, bit 0 of the current token PID is written back to the BD. • If KEEP=1, this bit is unchanged by USBFS. 0 No stall issued. 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 47-5. Buffer descriptor fields (continued)

Field	Description
	Setting BDT_STALL also causes the corresponding USB_ENDPT*n*[EPSTALL] bit to set. This causes USBOTG to issue a STALL handshake for both directions of the associated endpoint. To clear the stall condition: 1. Clear the associated USB_ENDPT*n*[EPSTALL] bit. 2. Write the BDT to clear OWN and BDT_STALL.
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 USBFS when a transfer completes. The values written back are the token PID values from the USB specification: • 0x1h for an OUT token. • 0x9h for an IN token. • 0xDh for a SETUP token. In host mode, this field is used to report the last returned PID or a transfer status indication. The possible values returned are: • 0x3h DATA0 • 0xBh DATA1 • 0x2h ACK • 0xEh STALL • 0xAh NAK • 0x0h Bus Timeout • 0xFh Data Error
1–0 Reserved	Reserved, should read as zeroes.
ADDR[31:0]	Address Represents the 32-bit buffer address in system memory. These bits are unchanged by USBFS.

47.4.5 USB transaction

When USBFS 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. USBFS 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, USBFS 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.

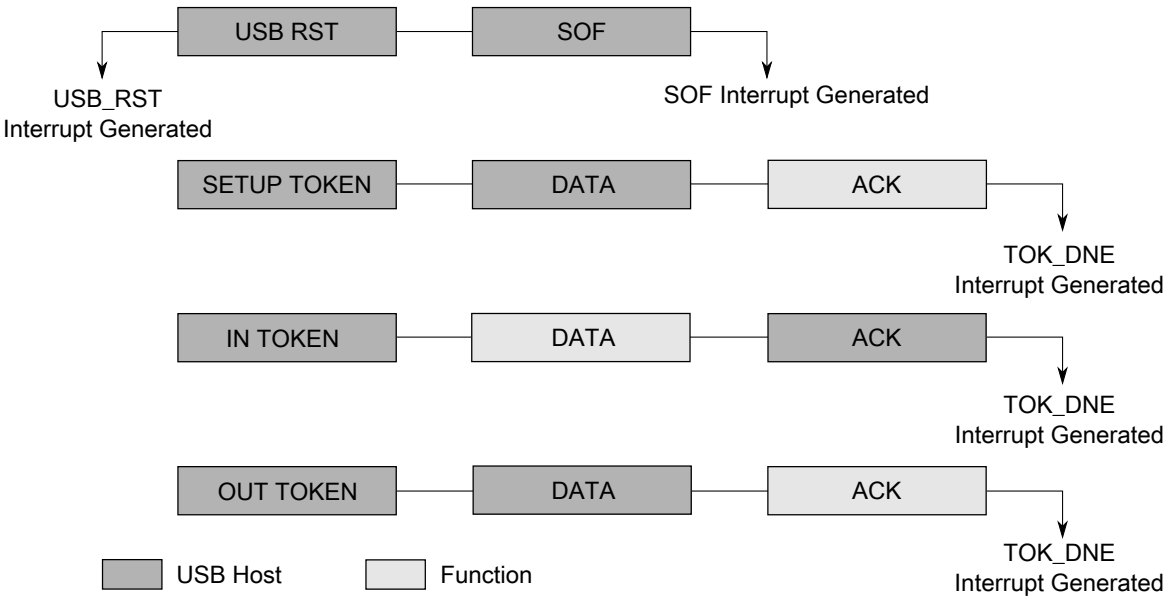


Figure 47-11. USB token transaction

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 47-6. 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.

Table continues on the next page...

Table 47-6. USB responses to DMA overrun errors (continued)

Errors due to Memory Latency	Errors due to Oversized Packets
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.
• 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. • 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.	

47.5 Memory map/Register definitions

This section provides the memory map and detailed descriptions of all USB interface registers.

47.5.1 USBFS memory mapping

The address space occupied by the USBFS registers is followed by bytes of embedded RAM used in Keep Alive mode. This USB RAM can also be used in Run mode.

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	47.5.2/1334
4007_2004	Peripheral ID Complement register (USB0_IDCOMP)	8	R	FBh	47.5.3/1334
4007_2008	Peripheral Revision register (USB0_REV)	8	R	33h	47.5.4/1335
4007_200C	Peripheral Additional Info register (USB0_ADDINFO)	8	R	01h	47.5.5/1335
4007_2010	OTG Interrupt Status register (USB0_OTGISTAT)	8	R/W	00h	47.5.6/1336
4007_2014	OTG Interrupt Control register (USB0_OTGICR)	8	R/W	00h	47.5.7/1337
4007_2018	OTG Status register (USB0_OTGSTAT)	8	R/W	00h	47.5.8/1338
4007_201C	OTG Control register (USB0_OTGCTL)	8	R/W	00h	47.5.9/1339
4007_2080	Interrupt Status register (USB0_ISTAT)	8	R/W	00h	47.5.10/ 1340

Table continues on the next page...

USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4007_2084	Interrupt Enable register (USB0_INTEN)	8	R/W	00h	47.5.11/ 1341
4007_2088	Error Interrupt Status register (USB0_ERRSTAT)	8	R/W	00h	47.5.12/ 1342
4007_208C	Error Interrupt Enable register (USB0_ERREN)	8	R/W	00h	47.5.13/ 1343
4007_2090	Status register (USB0_STAT)	8	R	00h	47.5.14/ 1344
4007_2094	Control register (USB0_CTL)	8	R/W	00h	47.5.15/ 1345
4007_2098	Address register (USB0_ADDR)	8	R/W	00h	47.5.16/ 1346
4007_209C	BDT Page register 1 (USB0_BDTPAGE1)	8	R/W	00h	47.5.17/ 1347
4007_20A0	Frame Number register Low (USB0_FRMNUML)	8	R/W	00h	47.5.18/ 1347
4007_20A4	Frame Number register High (USB0_FRMNUMH)	8	R/W	00h	47.5.19/ 1348
4007_20A8	Token register (USB0_TOKEN)	8	R/W	00h	47.5.20/ 1348
4007_20AC	SOF Threshold register (USB0_SOFTHLD)	8	R/W	00h	47.5.21/ 1349
4007_20B0	BDT Page Register 2 (USB0_BDTPAGE2)	8	R/W	00h	47.5.22/ 1350
4007_20B4	BDT Page Register 3 (USB0_BDTPAGE3)	8	R/W	00h	47.5.23/ 1350
4007_20C0	Endpoint Control register (USB0_ENDPT0)	8	R/W	00h	47.5.24/ 1351
4007_20C4	Endpoint Control register (USB0_ENDPT1)	8	R/W	00h	47.5.24/ 1351
4007_20C8	Endpoint Control register (USB0_ENDPT2)	8	R/W	00h	47.5.24/ 1351
4007_20CC	Endpoint Control register (USB0_ENDPT3)	8	R/W	00h	47.5.24/ 1351
4007_20D0	Endpoint Control register (USB0_ENDPT4)	8	R/W	00h	47.5.24/ 1351
4007_20D4	Endpoint Control register (USB0_ENDPT5)	8	R/W	00h	47.5.24/ 1351
4007_20D8	Endpoint Control register (USB0_ENDPT6)	8	R/W	00h	47.5.24/ 1351
4007_20DC	Endpoint Control register (USB0_ENDPT7)	8	R/W	00h	47.5.24/ 1351
4007_20E0	Endpoint Control register (USB0_ENDPT8)	8	R/W	00h	47.5.24/ 1351

Table continues on the next page...

USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4007_20E4	Endpoint Control register (USB0_ENDPT9)	8	R/W	00h	47.5.24/ 1351
4007_20E8	Endpoint Control register (USB0_ENDPT10)	8	R/W	00h	47.5.24/ 1351
4007_20EC	Endpoint Control register (USB0_ENDPT11)	8	R/W	00h	47.5.24/ 1351
4007_20F0	Endpoint Control register (USB0_ENDPT12)	8	R/W	00h	47.5.24/ 1351
4007_20F4	Endpoint Control register (USB0_ENDPT13)	8	R/W	00h	47.5.24/ 1351
4007_20F8	Endpoint Control register (USB0_ENDPT14)	8	R/W	00h	47.5.24/ 1351
4007_20FC	Endpoint Control register (USB0_ENDPT15)	8	R/W	00h	47.5.24/ 1351
4007_2100	USB Control register (USB0_USBCTRL)	8	R/W	C0h	47.5.25/ 1352
4007_2104	USB OTG Observe register (USB0_OBSERVE)	8	R	50h	47.5.26/ 1353
4007_2108	USB OTG Control register (USB0_CONTROL)	8	R/W	00h	47.5.27/ 1354
4007_210C	USB Transceiver Control register 0 (USB0_USBTRC0)	8	R/W	00h	47.5.28/ 1354
4007_2114	Frame Adjust Register (USB0_USBFRMADJUST)	8	R/W	00h	47.5.29/ 1356
4007_2124	Keep Alive mode control (USB0_KEEP_ALIVE_CTRL)	8	R/W	08h	47.5.30/ 1356
4007_2128	Keep Alive mode wakeup control (USB0_KEEP_ALIVE_WKCTRL)	8	R/W	01h	47.5.31/ 1357
4007_212C	Miscellaneous Control register (USB0_MISCCTRL)	8	R/W	00h	47.5.32/ 1358
4007_2130	Peripheral mode stall disable for endpoints 7 to 0 in IN direction (USB0_STALL_IL_DIS)	8	R/W	00h	47.5.33/ 1359
4007_2134	Peripheral mode stall disable for endpoints 15 to 8 in IN direction (USB0_STALL_IH_DIS)	8	R/W	00h	47.5.34/ 1360
4007_2138	Peripheral mode stall disable for endpoints 7 to 0 in OUT direction (USB0_STALL_OL_DIS)	8	R/W	00h	47.5.35/ 1361
4007_213C	Peripheral mode stall disable for endpoints 15 to 8 in OUT direction (USB0_STALL_OH_DIS)	8	R/W	00h	47.5.36/ 1362
4007_2140	USB Clock recovery control (USB0_CLK_RECOVER_CTRL)	8	R/W	00h	47.5.37/ 1363
4007_2144	IRC48M oscillator enable register (USB0_CLK_RECOVER_IRC_EN)	8	R/W	01h	47.5.38/ 1364
4007_2154	Clock recovery combined interrupt enable (USB0_CLK_RECOVER_INT_EN)	8	R/W	10h	47.5.39/ 1365

Table continues on the next page...

USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4007_215C	Clock recovery separated interrupt status (USB0_CLK_RECOVER_INT_STATUS)	8	w1c	00h	47.5.40/ 1366

47.5.2 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ID	Peripheral Identification This field always reads 0x4h.

47.5.3 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

Bit	7	6	5	4	3	2	1	0
Read	1		NID					
Write								
Reset	1	1	1	1	1	0	1	1

USBx_IDCOMP field descriptions

Field	Description
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
NID	Ones' complement of PERID[ID] bits.

47.5.4 Peripheral Revision register (USBx_REV)

Contains the revision number of the USB module.

Address: 4007_2000h base + 8h offset = 4007_2008h

Bit	7	6	5	4	3	2	1	0
Read	REV							
Write								
Reset	0	0	1	1	0	0	1	1

USBx_REV field descriptions

Field	Description
REV	Revision Indicates the revision number of the USB Core.

47.5.5 Peripheral Additional Info register (USBx_ADDINFO)

Reads back the value of the Host Enable bit.

Address: 4007_2000h base + Ch offset = 4007_200Ch

Bit	7	6	5	4	3	2	1	0
Read	0					0		IEHOST
Write								
Reset	0	0	0	0	0	0	0	1

USBx_ADDINFO field descriptions

Field	Description
7–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 IEHOST	This bit is set if host mode is enabled.

47.5.6 OTG Interrupt Status register (USBx_OTGISTAT)

Records changes to the state of the one-millisecond timer and line state stable logic. Software can read this register to determine the event that triggers an 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	Reserved	ONEMSEC	LINE_ STATE_ CHG	0	Reserved	Reserved	0	Reserved
Write	w1c	w1c	w1c		w1c	w1c		w1c
Reset	0	0	0	0	0	0	0	0

USBx_OTGISTAT field descriptions

Field	Description
7 Reserved	This field is reserved. Software must not change the value of this bitfield.
6 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 LINE_STATE_ CHG	This interrupt is set when the USB line state (CTL[SE0] and CTL[JSTATE] bits) are stable without change for 1 millisecond, and the value of the line state is different from the last time when the line state was stable. It is set on transitions between SE0 and J-state, SE0 and K-state, and J-state and K-state. Changes in J-state while SE0 is true do not cause an interrupt. This interrupt can be used in detecting Reset, Resume, Connect, and Data Line Pulse signaling.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	This field is reserved. Software must not change the value of this bitfield.
2 Reserved	This field is reserved. Software must not change the value of this bitfield.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 Reserved	This field is reserved. Software must not change the value of this bitfield.

47.5.7 OTG Interrupt Control register (USB*x*_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	Reserved	ONEMSECEN	LINESTATEEN	0	Reserved	Reserved	0	Reserved
Write								
Reset	0	0	0	0	0	0	0	0

USB*x*_OTGICR field descriptions

Field	Description
7 Reserved	Software must not change the value of this bitfield. This field is reserved.
6 ONEMSECEN	One Millisecond Interrupt Enable 0 Diables the 1ms timer interrupt. 1 Enables the 1ms timer interrupt.
5 LINESTATEEN	Line State Change Interrupt Enable 0 Disables the LINE_STAT_CHG interrupt. 1 Enables the LINE_STAT_CHG interrupt.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	Software must not change the value of this bitfield. This field is reserved.
2 Reserved	Software must not change the value of this bitfield. This field is reserved.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 Reserved	Software must not change the value of this bitfield. This field is reserved.

47.5.8 OTG Status register (USB*x*_OTGSTAT)

Displays the actual value from the one-millisecond timer and line state stable logic.

Address: 4007_2000h base + 18h offset = 4007_2018h

Bit	7	6	5	4
Read	0	ONEMSECEN	LINESTATESTABLE	0
Write				
Reset	0	0	0	0

Bit	3	2	1	0
Read	0	0	0	0
Write				
Reset	0	0	0	0

USB*x*_OTGSTAT field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 ONEMSECEN	This bit is reserved for the 1ms count, but it is not useful to software.
5 LINESTATESTABLE	Indicates that the internal signals that control the LINE_STATE_CHG field of OTGISTAT are stable for at least 1 ms. This bit is used to provide a hardware debounce of the linestate in detection of Connect, Disconnect and Resume signaling. 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. 0 The LINE_STAT_CHG bit is not yet stable. 1 The LINE_STAT_CHG bit has been debounced and is stable.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

47.5.9 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 DPHIGH	D+ Data Line pullup resistor enable 0 D+ pullup resistor is not enabled 1 D+ pullup resistor is enabled
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 DPLOW	D+ Data Line pull-down resistor enable This bit should always be enabled together with bit 4 (DMLOW) 0 D+ pulldown resistor is not enabled. 1 D+ pulldown resistor is enabled.
4 DMLOW	D– Data Line pull-down resistor enable 0 D- pulldown resistor is not enabled. 1 D- pulldown resistor is enabled.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 OTGEN	On-The-Go pullup/pulldown resistor enable 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. 1 The pull-up and pull-down controls in this register are used.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

47.5.10 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 STALL	Stall Interrupt In Device mode this bit is asserted when a STALL handshake is sent by the SIE. 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 ATTACH	Attach Interrupt This field is set when the USB Module detects an attach of a USB device. This field is only valid if CTL[HOSTMODEEN]=1. This interrupt signifies that a peripheral is now present and must be configured; it is asserted if there have been no transitions on the USB for 2.5 µs and the current bus state is not SE0." 0 No Attach is detected since the last time the ATTACH bit was cleared. 1 A peripheral is now present and must be configured (a stable non-SE0 state is detected for more than 2.5 µs).
5 RESUME	This bit is set when a K-state is observed on the DP/DM signals for 2.5 µs. When not in suspend mode this interrupt must be disabled.
4 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 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 SOFTOK	This bit is set when the USB Module receives a Start Of Frame (SOF) token. In Host mode this field is set when the SOF threshold is reached (MISCCTRL[SOFBUSSET]=0), or when the SOF counter reaches 0 (MISCCTRL[SOFBUSSET]=1), so that software can prepare for the next SOF.

Table continues on the next page...

USBx_ISTAT field descriptions (continued)

Field	Description
1 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 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.

47.5.11 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 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 STALLEN	STALL Interrupt Enable 0 Diasbles the STALL interrupt. 1 Enables the STALL interrupt.
6 ATTACHEN	ATTACH Interrupt Enable 0 Disables the ATTACH interrupt. 1 Enables the ATTACH interrupt.
5 RESUMEEN	RESUME Interrupt Enable 0 Disables the RESUME interrupt. 1 Enables the RESUME interrupt.
4 SLEEPEN	SLEEP Interrupt Enable 0 Disables the SLEEP interrupt. 1 Enables the SLEEP interrupt.
3 TOKDNEEN	TOKDNE Interrupt Enable 0 Disables the TOKDNE interrupt. 1 Enables the TOKDNE interrupt.
2 SOFTOKEN	SOFTOK Interrupt Enable 0 Disbles the SOFTOK interrupt. 1 Enables the SOFTOK interrupt.

Table continues on the next page...

USBx_INTEN field descriptions (continued)

Field	Description
1 ERROREN	ERROR Interrupt Enable 0 Disables the ERROR interrupt. 1 Enables the ERROR interrupt.
0 USBRSTEN	USBRST Interrupt Enable 0 Disables the USBRST interrupt. 1 Enables the USBRST interrupt.

47.5.12 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 condition 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	OWNERR	DMAERR	BTOERR	DFN8	CRC16	CRC5EOF	PIDERR
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

USBx_ERRSTAT field descriptions

Field	Description
7 BTSERR	This bit is set when a bit stuff error is detected. If set, the corresponding packet is rejected due to the error.
6 OWNERR	This field is valid when the USB Module is operating in peripheral mode (CTL[HOSTMODEEN]=0). It is set if the USB Module requires a new BD for SETUP, ISO IN, or ISO OUT transfer while a new BD is not available.
5 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 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 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.

Table continues on the next page...

USBx_ERRSTAT field descriptions (continued)

Field	Description
3 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 CRC16	This bit is set when a data packet is rejected due to a CRC16 error.
1 CRC5EOF	This error interrupt has two functions. When the USB Module is operating in peripheral mode (CTL[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. When the USB Module is operating in host mode (CTL[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 PIDERR	This bit is set when the PID check field fails.

47.5.13 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 condition 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 Write	BTSERREN	OWNERREN	DMAERREN	BTOERREN	DFN8EN	CRC16EN	CRC5EOFEN	PIDERREN
Reset	0	0	0	0	0	0	0	0

USBx_ERREN field descriptions

Field	Description
7 BTSERREN	BTSERR Interrupt Enable 0 Disables the BTSERR interrupt. 1 Enables the BTSERR interrupt.
6 OWNERREN	OWNERR Interrupt Enable This field is valid when the USB module is operating in peripheral mode (CTL[HOSTMODEEN]=0). 0 Disables the OWNERR interrupt. 1 Enables the OWNERR interrupt.
5 DMAERREN	DMAERR Interrupt Enable 0 Disables the DMAERR interrupt. 1 Enables the DMAERR interrupt.

Table continues on the next page...

USBx_ERREN field descriptions (continued)

Field	Description
4 BTOERREN	BTOERR Interrupt Enable 0 Disables the BTOERR interrupt. 1 Enables the BTOERR interrupt.
3 DFN8EN	DFN8 Interrupt Enable 0 Disables the DFN8 interrupt. 1 Enables the DFN8 interrupt.
2 CRC16EN	CRC16 Interrupt Enable 0 Disables the CRC16 interrupt. 1 Enables the CRC16 interrupt.
1 CRC5EOFEN	CRC5/EOF Interrupt Enable 0 Disables the CRC5/EOF interrupt. 1 Enables the CRC5/EOF interrupt.
0 PIDERREN	PIDERR Interrupt Enable 0 Disables the PIDERR interrupt. 1 Enters the PIDERR interrupt.

47.5.14 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 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 TX	Transmit Indicator 0 The most recent transaction was a receive operation. 1 The most recent transaction was a transmit operation.
2 ODD	This bit is set if the last buffer descriptor updated was in the odd bank of the BDT.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

47.5.15 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 Write	JSTATE	SE0	TXSUSPENDTOKENBUSY	RESET
Reset	0	0	0	0

Bit	3	2	1	0
Read Write	HOSTMODEEN	RESUME	ODDRST	USBENSOFEN
Reset	0	0	0	0

USBx_CTL field descriptions

Field	Description
7 JSTATE	Live USB differential receiver JSTATE signal The polarity of this signal is affected by the current state of LSEN .
6 SE0	Live USB Single Ended Zero signal
5 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. In Device 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 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 (CTL[HOSTMODEEN]=1). Software must set RESET to 1 for the required amount of time and then clear it to 0 to end reset signaling.

Table continues on the next page...

USBx_CTL field descriptions (continued)

Field	Description
3 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 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 HOSTMODEENis set, the USB module appends a Low Speed End of Packet to the Resume signaling when the RESUME bit is cleared.
1 ODDRST	Setting this bit to 1 resets all the BDT ODD ping/pong fields to 0, which then specifies the EVEN BDT bank.
0 USBENSOFEN	USB Enable Setting this bit enables the USB-FS to operate; clearing it disables the USB-FS. Setting the 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. 0 Disables the USB Module. 1 Enables the USB Module.

47.5.16 Address register (USBx_ADDR)

Holds the unique USB address that the USB module decodes when in Peripheral mode (CTL[HOSTMODEEN]=0). When operating in Host mode (CTL[HOSTMODEEN]=1) the USB module transmits this address with a TOKEN packet. This enables the USB module to uniquely address any USB peripheral. In either mode, CTL[USBENSOFEN] 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 Write	LSEN	ADDR						
Reset	0	0	0	0	0	0	0	0

USBx_ADDR field descriptions

Field	Description
7 LSEN	Low Speed Enable bit 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.
ADDR	USB Address

Table continues on the next page...

USBx_ADDR field descriptions (continued)

Field	Description
	Defines the USB address that the USB module decodes in peripheral mode, or transmits when in host mode.

47.5.17 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. See Buffer Descriptor Table. 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 BDTBA	Provides address bits 15 through 9 of the BDT base address.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

47.5.18 Frame Number register Low (USBx_FRMNUML)

The Frame Number registers (low and high) contain the 11-bit frame number. These registers are updated with the current frame number whenever a SOF TOKEN is received.

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
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.

47.5.19 Frame Number register High (USBx_FRMNUMH)

The Frame Number registers (low and high) contain the 11-bit frame number. These registers are updated with the current frame number whenever a SOF TOKEN is received.

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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
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.

47.5.20 Token register (USBx_TOKEN)

Used to initiate USB transactions when in host mode (CTL[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 Write	TOKENPID				TOKENENDPT			
Reset	0	0	0	0	0	0	0	0

USBx_TOKEN field descriptions

Field	Description
7–4 TOKENPID	Contains the token type executed by the USB module. 0001 OUT Token. USB Module performs an OUT (TX) transaction. 1001 IN Token. USB Module performs an In (RX) transaction. 1101 SETUP Token. USB Module performs a SETUP (TX) transaction
TOKENENDPT	Holds the Endpoint address for the token command. The four bit value written must be a valid endpoint.

47.5.21 SOF Threshold register (USBx_SOFTHLD)

The SOF Threshold Register is used only in Host mode (CTL[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.

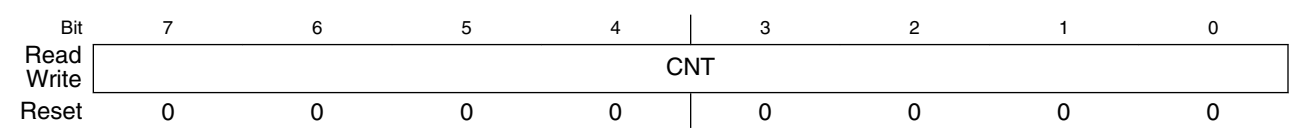
The SOF threshold register is used to program the number of USB byte times when SOFDYNTHLD=0, or 8 byte times when SOFDYNTHLD=1, before the SOF stops initiating token packet transactions. This register must be set to a value that ensures that other packets are not actively being transmitted when the 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 peripheral device 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
CNT	Represents the SOF count threshold in byte times when SOFDYNTHLD=0 or 8 byte times when SOFDYNTHLD=1.

47.5.22 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. See Buffer Descriptor Table.

Address: 4007_2000h base + B0h offset = 4007_20B0h

Bit	7	6	5	4	3	2	1	0
Read Write	BDTBA							
Reset	0	0	0	0	0	0	0	0

USBx_BDTPAGE2 field descriptions

Field	Description
BDTBA	Provides address bits 23 through 16 of the BDT base address that defines the location of Buffer Descriptor Table resides in system memory.

47.5.23 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. See Buffer Descriptor Table.

Address: 4007_2000h base + B4h offset = 4007_20B4h

Bit	7	6	5	4	3	2	1	0
Read Write	BDTBA							
Reset	0	0	0	0	0	0	0	0

USBx_BDTPAGE3 field descriptions

Field	Description
BDTBA	Provides address bits 31 through 24 of the BDT base address that defines the location of Buffer Descriptor Table resides in system memory.

47.5.24 Endpoint Control register (USB*x*_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.

The three bits EPCTLDIS, EPRXEN, and EPTXEN define if an endpoint is enabled and define the direction of the endpoint. The endpoint enable/direction control is defined in the following table.

Table 47-7. Endpoint enable and direction control

EPCTLDIS	EPRXEN	EPTXEN	Endpoint enable/direction control
X	0	0	Disable endpoint
X	0	1	Enable endpoint for Tx transfers only
X	1	0	Enable endpoint for Rx transfers only
1	1	1	Enable endpoint for Rx and Tx transfers
0	1	1	Enable Endpoint for RX and TX as well as control (SETUP) transfers.

Address: 4007_2000h base + C0h offset + (4d × i), where i=0d to 15d

Bit	7	6	5	4	3	2	1	0
Read	HOSTWOHUB	RETRYDIS	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
Write								
Reset	0	0	0	0	0	0	0	0

USB*x*_ENDPT*n* field descriptions

Field	Description
7 HOSTWOHUB	Host without a hub This is a Host mode only field and is present in the control register for endpoint 0 (ENDPT0) only.

Table continues on the next page...

USBx_ENDPTn field descriptions (continued)

Field	Description
	0 Low-speed device connected to Host through a hub. PRE_PID will be generated as required. 1 Low-speed device directly connected. No hub, or no low-speed device attached.
6 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 are retried in hardware. This bit must be set when the host is attempting to poll an interrupt endpoint.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 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. See Table 47-7
3 EPRXEN	This bit, when set, enables the endpoint for RX transfers. See Table 47-7
2 EPTXEN	This bit, when set, enables the endpoint for TX transfers. See Table 47-7
1 EPSTALL	When set, this bit indicates that the endpoint is stalled. This bit has priority over all other control bits in this 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 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.

47.5.25 USB Control register (USBx_USBCTRL)

Address: 4007_2000h base + 100h offset = 4007_2100h

Bit	7	6	5	4	3	2	1	0
Read	SUSP	PDE	UARTCHLS	UARTSEL	0			
Write								
Reset	1	1	0	0	0	0	0	0

USBx_USBCTRL field descriptions

Field	Description
7 SUSP	Places the USB transceiver into the suspend state. 0 USB transceiver is not in suspend state. 1 USB transceiver is in suspend state.
6 PDE	Enables the weak pulldowns on the USB transceiver. 0 Weak pulldowns are disabled on D+ and D–. 1 Weak pulldowns are enabled on D+ and D–.
5 UARTCHLS	UART Signal Channel Select This field is valid only when USB signals are selected to be used as UART signals.

Table continues on the next page...

USBx_USBCTRL field descriptions (continued)

Field	Description
	0 USB DP/DM signals used as UART TX/RX. 1 USB DP/DM signals used as UART RX/TX.
4 UARTSEL	Selects USB signals to be used as UART signals. 0 USB signals not used as UART signals. 1 USB signals used as UART signals.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

47.5.26 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			
Write								
Reset	0	1	0	1	0	0	0	0

USBx_OBSERVE field descriptions

Field	Description
7 DPPU	Provides observability of the D+ Pullup enable at the USB transceiver. 0 D+ pullup disabled. 1 D+ pullup enabled.
6 DPPD	Provides observability of the D+ Pulldown enable at the USB transceiver. 0 D+ pulldown disabled. 1 D+ pulldown enabled.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 DMPD	Provides observability of the D- Pulldown enable at the USB transceiver. 0 D– pulldown disabled. 1 D– pulldown enabled.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

47.5.27 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 DPPULLUPNONOTG	Provides control of the DP Pullup in USBOTG, if USB is configured in non-OTG device mode. 0 DP Pullup in non-OTG device mode is not enabled. 1 DP Pullup in non-OTG device mode is enabled.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

47.5.28 USB Transceiver Control register 0 (USBx_USBTRC0)

Includes signals for basic operation of the on-chip USB Full Speed transceiver and configuration of the USB data connection that are not otherwise included in the USB Full Speed controller registers. VREGIN interrupt detection using the VFEDG_DET and VREDG_DET bitfields may be used for VBUS detection in bus-powered device use cases when the USB receptacle VBUS pin is connected to VREGIN.

Address: 4007_2000h base + 10Ch offset = 4007_210Ch

Bit	7	6	5	4
Read		0	USBRESMEN	VFEDG_DET
Write	USBRESET			
Reset	0	0	0	0

Bit	3	2	1	0
Read	VREDG_DET	USB_CLK_ RECOVERY_INT	SYNC_DET	USB_RESUME_INT
Write				
Reset	0	0	0	0

USB*x*_USBTRC0 field descriptions

Field	Description
7 USBRESET	USB Reset Generates a hard reset to USBOTG. After this bit is set and the reset occurs, this bit is automatically cleared. **NOTE: This bit is always read as zero. Wait two USB clock cycles after setting this bit before accessing other USB register bit fields.** 0 Normal USB module operation. 1 Returns the USB module to its reset state.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 USBRESMEN	Asynchronous Resume Interrupt Enable 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. 0 USB asynchronous wakeup from suspend mode disabled. 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 interrupt should only be enabled when the Transceiver is suspended.
4 VFEDG_DET	VREGIN Falling Edge Interrupt Detect Use USB*x*_MISCCTRL[VFEDG_EN] to enable this bitfield. 0 VREGIN falling edge interrupt has not been detected. 1 VREGIN falling edge interrupt has been detected.
3 VREDG_DET	VREGIN Rising Edge Interrupt Detect Use USB*x*_MISCCTRL[VREDG_EN] to enable this bitfield. 0 VREGIN rising edge interrupt has not been detected. 1 VREGIN rising edge interrupt has been detected.
2 USB_CLK_ RECOVERY_INT	Combined USB Clock Recovery interrupt status This read-only field will be set to value high at 1'b1 when any of USB clock recovery interrupt conditions are detected and those interrupts are unmasked. For customer use the only unmasked USB clock recovery interrupt condition results from an overflow of the frequency trim setting values indicating that the frequency trim calculated is out of the adjustment range of the IRC48M output clock. To clear this bit after it has been set, Write 0xFF to register USB_CLK_RECOVER_INT_STATUS.
1 SYNC_DET	Synchronous USB Interrupt Detect 0 Synchronous interrupt has not been detected. 1 Synchronous interrupt has been detected.
0 USB_RESUME_ INT	USB Asynchronous Interrupt 0 No interrupt was generated. 1 Interrupt was generated because of the USB asynchronous interrupt.

47.5.29 Frame Adjust Register (USBx_USBFRMADJUST)

Address: 4007_2000h base + 114h offset = 4007_2114h

Bit	7	6	5	4	3	2	1	0
Read Write	ADJ							
Reset	0	0	0	0	0	0	0	0

USBx_USBFRMADJUST field descriptions

Field	Description
ADJ	Frame Adjustment 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 start of the next frame.

47.5.30 Keep Alive mode control (USBx_KEEP_ALIVE_CTRL)

Address: 4007_2000h base + 124h offset = 4007_2124h

Bit	7	6	5	4
Read	WAKE_INT_STS	0		WAKE_INT_EN
Write				
Reset	0	0	0	0

Bit	3	2	1	0
Read	WAKE_REQ_EN	0	OWN_OVERRD_EN	KEEP_ALIVE_EN
Write				
Reset	1	0	0	0

USBx_KEEP_ALIVE_CTRL field descriptions

Field	Description
7 WAKE_INT_STS	Wakeup Interrupt Status. If KEEP_ALIVE mode is enabled and a SETUP token or software assigned token is received, this interrupt is set only when the token is processed and the USB state is IDLE. Reset to 0. Software read only, and write 1 to clear.
6–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 WAKE_INT_EN	Wakeup Interrupt Enable. Set to 1 to enable WAKE_INT_STS. Reset to 0. Software RW.
3 WAKE_REQ_EN	During KEEP_ALIVE mode, a bus access by the USB controller to a memory location outside the USB SRAM will cause the bus access to stall until KEEP_ALIVE mode is exited. When WAKE_REQ_EN is set,

Table continues on the next page...

USB*x*_KEEP_ALIVE_CTRL field descriptions (continued)

Field	Description
	a bus access by the USB controller to a memory location outside the USB SRAM will generate a wakeup allowing the bus access to complete. Once the bus access completes, the USB will reenter the KEEP_ALIVE mode provided no other wakeups have been detected. An interrupt is not generated as a result of the wakeup request. 0 USB bus wakeup request is disabled. 1 USB bus wakeup request is enabled.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 OWN_OVERRD_EN	When set to 1, during KEEP_ALIVE mode, if received token is not SETUP, the OWN bit of current BD will be forced to 0, so usb core will respond with NAK. Reset to 0. Software RW.
0 KEEP_ALIVE_EN	Global enable for USB_KEEP_ALIVE mode. When assert, usb shall enter USB_KEEP_ALIVE mode after asserting ipg_stop; when de-assert, everything is same as before. Reset to 0. Software RW.

47.5.31 Keep Alive mode wakeup control (USB*x*_KEEP_ALIVE_WKCTRL)

Address: 4007_2000h base + 128h offset = 4007_2128h

Bit	7	6	5	4	3	2	1	0
Read	WAKE_ENDPT				WAKE_ON_THIS			
Write								
Reset	0	0	0	0	0	0	0	1

USB*x*_KEEP_ALIVE_WKCTRL field descriptions

Field	Description
7–4 WAKE_ENDPT	Indicates which endpoint causes the wakeup interrupt. Reset to 0, software read only.
WAKE_ON_THIS	Software configure it to which token can wakeup usb during KEEP_ALIVE mode. Reset to 4'b0001, software RW. 0001 Wake up on receiving OUT/SETUP token packet. 1101 Wake up on receiving SETUP token packet. All other values are reserved.

47.5.32 Miscellaneous Control register (USBx_MISCCTRL)

Address: 4007_2000h base + 12Ch offset = 4007_212Ch

Bit	7	6	5	4
Read	STL_ADJ_EN	0		VFEDG_EN
Write				
Reset	0	0	0	0

Bit	3	2	1	0
Read	VREDG_EN	OWNERRISODIS	SOFBUSSET	SOFDYNTHLD
Write				
Reset	0	0	0	0

USBx_MISCCTRL field descriptions

Field	Description
7 STL_ADJ_EN	USB Peripheral mode Stall Adjust Enable This field is valid only in peripheral mode (CTL[HOSTMODEEN]=0). By default (STL_ADJ_EN = 0), when an endpoint is stalled (ENDPTn[END_STALL]=1), both IN and OUT directions of the endpoint are stalled. If STL_ADJ_EN = 1, then when an endpoint is stalled (ENDPTn[END_STALL]=1), then the USBx_STALL_xx_DIS registers can be used to control which endpoint direction(s) are affected. 0 If USB_ENDPTn[END_STALL] = 1, both IN and OUT directions for the associated endpoint will be stalled 1 If USB_ENDPTn[END_STALL] = 1, the USB_STALL_xx_DIS registers control which directions for the associated endpoint will be stalled.
6–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 VFEDG_EN	VREGIN Falling Edge Interrupt Enable 0 VREGIN falling edge interrupt disabled. 1 VREGIN falling edge interrupt enabled.
3 VREDG_EN	VREGIN Rising Edge Interrupt Enable 0 VREGIN rising edge interrupt disabled. 1 VREGIN rising edge interrupt enabled.
2 OWNERRISODIS	OWN Error Detect for ISO IN / ISO OUT Disable This field is only valid for Peripheral mode, that is, CTL[HOSTMODEEN]=0. 0 OWN error detect for ISO IN / ISO OUT is not disabled. 1 OWN error detect for ISO IN / ISO OUT is disabled.
1 SOFBUSSET	SOF_TOK Interrupt Generation Mode Select This field is only valid for Host mode, that is, CTL[HOSTMODEEN]=1. 0 SOF_TOK interrupt is set according to SOF threshold value. 1 SOF_TOK interrupt is set when SOF counter reaches 0.
0 SOFDYNTHLD	Dynamic SOF Threshold Compare mode This field is only valid for Host mode, that is, CTL[HOSTMODEEN]=1.

Table continues on the next page...

USBx_MISCCTRL field descriptions (continued)

Field	Description
	0 SOF_TOK interrupt is set when byte times SOF threshold is reached. 1 SOF_TOK interrupt is set when 8 byte times SOF threshold is reached or overstepped.

47.5.33 Peripheral mode stall disable for endpoints 7 to 0 in IN direction (USBx_STALL_IL_DIS)

This register is valid only in Peripheral mode(CTL[HOSTMODEEN]=0) and when stall adjust enable bit is set (MISCCTRL[STL_ADJ_EN]=1).

When an endpoint is stalled (ENDPTn[END_STALL]=1), the fields in this register enable or disable stalling of the IN direction of the endpoints 7 to 0.

Address: 4007_2000h base + 130h offset = 4007_2130h

Bit	7	6	5	4	3	2	1	0
Read Write	STALL_I_DIS7	STALL_I_DIS6	STALL_I_DIS5	STALL_I_DIS4	STALL_I_DIS3	STALL_I_DIS2	STALL_I_DIS1	STALL_I_DIS0
Reset	0	0	0	0	0	0	0	0

USBx_STALL_IL_DIS field descriptions

Field	Description
7 STALL_I_DIS7	Disable endpoint 7 IN direction. 0 Endpoint 7 IN direction stall is enabled. 1 Endpoint 7 IN direction stall is disabled.
6 STALL_I_DIS6	Disable endpoint 6 IN direction. 0 Endpoint 6 IN direction stall is enabled. 1 Endpoint 6 IN direction stall is disabled.
5 STALL_I_DIS5	Disable endpoint 5 IN direction. 0 Endpoint 5 IN direction stall is enabled. 1 Endpoint 5 IN direction stall is disabled.
4 STALL_I_DIS4	Disable endpoint 4 IN direction. 0 Endpoint 4 IN direction stall is enabled. 1 Endpoint 4 IN direction stall is disabled.
3 STALL_I_DIS3	Disable endpoint 3 IN direction. 0 Endpoint 3 IN direction stall is enabled. 1 Endpoint 3 IN direction stall is disabled.
2 STALL_I_DIS2	Disable endpoint 2 IN direction. 0 Endpoint 2 IN direction stall is enabled. 1 Endpoint 2 IN direction stall is disabled.

Table continues on the next page...

USBx_STALL_IL_DIS field descriptions (continued)

Field	Description
1 STALL_I_DIS1	Disable endpoint 1 IN direction. 0 Endpoint 1 IN direction stall is enabled. 1 Endpoint 1 IN direction stall is disabled.
0 STALL_I_DIS0	Disable endpoint 0 IN direction. 0 Endpoint 0 IN direction stall is enabled. 1 Endpoint 0 IN direction stall is disabled.

47.5.34 Peripheral mode stall disable for endpoints 15 to 8 in IN direction (USBx_STALL_IH_DIS)

This register is valid only in Peripheral mode(CTL[HOSTMODEEN]=0) and when stall adjust enable bit is set (MISCCTRL[STL_ADJ_EN]=1).

When an endpoint is stalled (ENDPTn[END_STALL]=1), the fields in this register enable or disable stalling of the IN direction of the endpoints 15 to 8.

Address: 4007_2000h base + 134h offset = 4007_2134h

Bit	7	6	5	4	3	2	1	0
Read Write	STALL_I_ DIS15	STALL_I_ DIS14	STALL_I_ DIS13	STALL_I_ DIS12	STALL_I_ DIS11	STALL_I_ DIS10	STALL_I_ DIS9	STALL_I_ DIS8
Reset	0	0	0	0	0	0	0	0

USBx_STALL_IH_DIS field descriptions

Field	Description
7 STALL_I_DIS15	Disable endpoint 15 IN direction. 0 Endpoint 15 IN direction stall is enabled. 1 Endpoint 15 IN direction stall is disabled.
6 STALL_I_DIS14	Disable endpoint 14 IN direction. 0 Endpoint 14 IN direction stall is enabled. 1 Endpoint 14 IN direction stall is disabled.
5 STALL_I_DIS13	Disable endpoint 13 IN direction. 0 Endpoint 13 IN direction stall is enabled. 1 Endpoint 13 IN direction stall is disabled.
4 STALL_I_DIS12	Disable endpoint 12 IN direction. 0 Endpoint 12 IN direction stall is enabled. 1 Endpoint 12 IN direction stall is disabled.
3 STALL_I_DIS11	Disable endpoint 11 IN direction.

Table continues on the next page...

USBx_STALL_IH_DIS field descriptions (continued)

Field	Description
	0 Endpoint 11 IN direction stall is enabled. 1 Endpoint 11 IN direction stall is disabled.
2 STALL_I_DIS10	Disable endpoint 10 IN direction. 0 Endpoint 10 IN direction stall is enabled. 1 Endpoint 10 IN direction stall is disabled.
1 STALL_I_DIS9	Disable endpoint 9 IN direction. 0 Endpoint 9 IN direction stall is enabled. 1 Endpoint 9 IN direction stall is disabled.
0 STALL_I_DIS8	Disable endpoint 8 IN direction. 0 Endpoint 8 IN direction stall is enabled. 1 Endpoint 8 IN direction stall is disabled.

47.5.35 Peripheral mode stall disable for endpoints 7 to 0 in OUT direction (USBx_STALL_OL_DIS)

This register is valid only in Peripheral mode(CTL[HOSTMODEEN]=0) and when stall adjust enable bit is set (MISCCTRL[STL_ADJ_EN]=1).

When an endpoint is stalled (ENDPTn[END_STALL]=1), the fields in this register enable or disable stalling of the OUT direction for the endpoints 7 to 0.

Address: 4007_2000h base + 138h offset = 4007_2138h

Bit	7	6	5	4	3	2	1	0
Read Write	STALL_O_DIS7	STALL_O_DIS6	STALL_O_DIS5	STALL_O_DIS4	STALL_O_DIS3	STALL_O_DIS2	STALL_O_DIS1	STALL_O_DIS0
Reset	0	0	0	0	0	0	0	0

USBx_STALL_OL_DIS field descriptions

Field	Description
7 STALL_O_DIS7	Disable endpoint 7 OUT direction. 0 Endpoint 7 OUT direction stall is enabled. 1 Endpoint 7 OUT direction stall is disabled.
6 STALL_O_DIS6	Disable endpoint 6 OUT direction. 0 Endpoint 6 OUT direction stall is enabled. 1 Endpoint 6 OUT direction stall is disabled.
5 STALL_O_DIS5	Disable endpoint 5 OUT direction. 0 Endpoint 5 OUT direction stall is enabled. 1 Endpoint 5 OUT direction stall is disabled.

Table continues on the next page...

USBx_STALL_OL_DIS field descriptions (continued)

Field	Description
4 STALL_O_DIS4	Disable endpoint 4 OUT direction. 0 Endpoint 4 OUT direction stall is enabled. 1 Endpoint 4 OUT direction stall is disabled.
3 STALL_O_DIS3	Disable endpoint 3 OUT direction. 0 Endpoint 3 OUT direction stall is enabled. 1 Endpoint 3 OUT direction stall is disabled.
2 STALL_O_DIS2	Disable endpoint 2 OUT direction. 0 Endpoint 2 OUT direction stall is enabled. 1 Endpoint 2 OUT direction stall is disabled.
1 STALL_O_DIS1	Disable endpoint 1 OUT direction. 0 Endpoint 1 OUT direction stall is enabled. 1 Endpoint 1 OUT direction stall is disabled.
0 STALL_O_DIS0	Disable endpoint 0 OUT direction. 0 Endpoint 0 OUT direction stall is enabled. 1 Endpoint 0 OUT direction stall is disabled.

47.5.36 Peripheral mode stall disable for endpoints 15 to 8 in OUT direction (USBx_STALL_OH_DIS)

This register is valid only in Peripheral mode(CTL[HOSTMODEEN]=0) and when stall adjust enable bit is set (MISCCTRL[STL_ADJ_EN]=1).

When an endpoint is stalled (ENDPTn[END_STALL]=1), the fields in this register enable or disable stalling of the OUT direction for the endpoints 15 to 8.

Address: 4007_2000h base + 13Ch offset = 4007_213Ch

Bit	7	6	5	4	3	2	1	0
Read Write	STALL_O_DIS15	STALL_O_DIS14	STALL_O_DIS13	STALL_O_DIS12	STALL_O_DIS11	STALL_O_DIS10	STALL_O_DIS9	STALL_O_DIS8
Reset	0	0	0	0	0	0	0	0

USBx_STALL_OH_DIS field descriptions

Field	Description
7 STALL_O_DIS15	Disable endpoint 15 OUT direction. 0 Endpoint 15 OUT direction stall is enabled. 1 Endpoint 15 OUT direction stall is disabled.
6 STALL_O_DIS14	Disable endpoint 14 OUT direction.

Table continues on the next page...

USBx_STALL_OH_DIS field descriptions (continued)

Field	Description
	0 Endpoint 14 OUT direction stall is enabled. 1 Endpoint 14 OUT direction stall is disabled.
5 STALL_O_DIS13	Disable endpoint 13 OUT direction. 0 Endpoint 13 OUT direction stall is enabled. 1 Endpoint 13 OUT direction stall is disabled.
4 STALL_O_DIS12	Disable endpoint 12 OUT direction. 0 Endpoint 12 OUT direction stall is enabled. 1 Endpoint 12 OUT direction stall is disabled.
3 STALL_O_DIS11	Disable endpoint 11 OUT direction. 0 Endpoint 11 OUT direction stall is enabled. 1 Endpoint 11 OUT direction stall is disabled.
2 STALL_O_DIS10	Disable endpoint 10 OUT direction. 0 Endpoint 10 OUT direction stall is enabled. 1 Endpoint 10 OUT direction stall is disabled.
1 STALL_O_DIS9	Disable endpoint 9 OUT direction. 0 Endpoint 9 OUT direction stall is enabled. 1 Endpoint 9 OUT direction stall is disabled.
0 STALL_O_DIS8	Disable endpoint 8 OUT direction. 0 Endpoint 8 OUT direction stall is enabled. 1 Endpoint 8 OUT direction stall is disabled.

47.5.37 USB Clock recovery control (USBx_CLK_RECOVER_CTRL)

Signals in this register control the crystal-less USB clock mode in which the internal IRC48M oscillator is tuned to match the clock extracted from the incoming USB data stream.

The IRC48M internal oscillator module must be enabled in register USB_CLK_RECOVER_IRC_EN for this mode.

Address: 4007_2000h base + 140h offset = 4007_2140h

Bit	7	6	5	4	3	2	1	0
Read Write	CLOCK_RECOVER_EN	RESET_RESUME_ROUGH_EN	RESTART_IFRTRIM_EN	Reserved		Reserved	Reserved	Reserved
Reset	0	0	0	0	0	0	0	0

USBx_CLK_RECOVER_CTRL field descriptions

Field	Description
7 CLOCK_ RECOVER_EN	Crystal-less USB enable This bit must be enabled if user wants to use the crystal-less USB mode for the Full Speed USB controller and transceiver. **NOTE: This bit should not be set for USB host mode or OTG.** 0 Disable clock recovery block (default) 1 Enable clock recovery block
6 RESET_ RESUME_ ROUGH_EN	Reset/resume to rough phase enable The clock recovery block tracks the IRC48M to get an accurate 48Mhz clock. It has two phases after user enables clock_recover_en bit, rough phase and tracking phase. The step to fine tune the IRC48M by adjusting the trim fine value is different during these 2 phases. The step in rough phase is larger than that in tracking phase. Switch back to rough stage whenever USB bus reset or bus resume occurs. 0 Always works in tracking phase after the first time rough to track transition (default) 1 Go back to rough stage whenever bus reset or bus resume occurs
5 RESTART_ IFRTRIM_EN	Restart from IFR trim value IRC48M has a default trim fine value whose default value is factory trimmed (the IFR trim value). Clock recover block tracks the accuracy of the clock 48Mhz and keeps updating the trim fine value accordingly 0 Trim fine adjustment always works based on the previous updated trim fine value (default) 1 Trim fine restarts from the IFR trim value whenever bus_reset/bus_resume is detected or module enable is desasserted
4–3 Reserved	This field is reserved.
2 Reserved	This field is reserved. This bit is for Freescale use only. Customers should not change this bit from its default state.
1 Reserved	This field is reserved. This bit is for Freescale use only. Customers should not change this bit from its default state.
0 Reserved	This field is reserved. Default should not be changed

47.5.38 IRC48M oscillator enable register (USBx_CLK_RECOVER_IRC_EN)

Controls basic operation of the on-chip IRC48M module used to produce nominal 48MHz clocks for USB crystal-less operation and other functions.

See additional information about the IRC48M operation in the Clock Distribution chapter.

Address: 4007_2000h base + 144h offset = 4007_2144h

Bit	7	6	5	4	3	2	1	0
Read Write	Reserved						IRC_EN	REG_EN
Reset	0	0	0	0	0	0	0	1

USBx_CLK_RECOVER_IRC_EN field descriptions

Field	Description
7–2 Reserved	This field is reserved.
1 IRC_EN	IRC48M enable This bit is used to enable the on-chip IRC48M module to generate clocks for crystal-less USB. It can be used for FS USB device mode operation. This bit must be set before using the crystal-less USB clock configuration. 0 Disable the IRC48M module (default) 1 Enable the IRC48M module
0 REG_EN	IRC48M regulator enable This bit is used to enable the local analog regulator for IRC48M module. This bit must be set if user wants to use the crystal-less USB clock configuration. 0 IRC48M local regulator is disabled 1 IRC48M local regulator is enabled (default)

47.5.39 Clock recovery combined interrupt enable (USBx_CLK_RECOVER_INT_EN)

Enables or masks the individual interrupt flags which are logically OR'ed together to produce the combined interrupt indication on the USB_CLK_RECOVERY_INT bit in the USB_USBTRC0 register if the indicated conditions have been detected in the USB clock recovery algorithm operation.

Address: 4007_2000h base + 154h offset = 4007_2154h

Bit	7	6	5	4	3	2	1	0
Read Write	Reserved			OVF_ ERROR_EN	Reserved			
Reset	0	0	0	1	0	0	0	0

USBx_CLK_RECOVER_INT_EN field descriptions

Field	Description
7–5 Reserved	This field is reserved. Should always be written as 0.
4 OVF_ERROR_ EN	Determines whether OVF_ERROR condition signal is used in generation of USB_CLK_RECOVERY_INT. 0 The interrupt will be masked 1 The interrupt will be enabled (default)
Reserved	This field is reserved. Should always be written as 0.

47.5.40 Clock recovery separated interrupt status (USB*x*_CLK_RECOVER_INT_STATUS)

A Write operation with value high at 1'b1 on any combination of individual bits will clear those bits.

Address: 4007_2000h base + 15Ch offset = 4007_215Ch

Bit	7	6	5	4	3	2	1	0
Read	Reserved			OVF_ ERROR	Reserved			
Write	w1c			w1c	w1c			
Reset	0	0	0	0	0	0	0	0

USB*x*_CLK_RECOVER_INT_STATUS field descriptions

Field	Description
7–5 Reserved	This field is reserved. Should always be written as 0.
4 OVF_ERROR	Indicates that the USB clock recovery algorithm has detected that the frequency trim adjustment needed for the IRC48M output clock is outside the available TRIM_FINE adjustment range for the IRC48M module. 0 No interrupt is reported 1 Unmasked interrupt has been generated
Reserved	This field is reserved. Should always be written as 0.

47.6 Keep Alive mode

This mode is for chips with device-only USB implementations.

NOTE

The address range 0x4010_0000 to 0x4010_07FF consists of 2 KB of on-chip USB RAM used for the BDT and small data packet accessed by the USB core in Keep Alive mode.

This USB RAM can also be used in Run mode.

Keep Alive mode keeps USB active in STOP/VLPS low-power modes so there is no need to re-enumerate exiting low power modes. When connected to the host and when there is no data to transfer, to save extra power, software can power down the system bus and the CPU, keeping only USB power on. In this mode, only the USB 48 MHz functional clock

is required by the USB controller, and the rest of the device enters Stop mode. The USB core wakes up when the SETUP token or a software-assigned token is received, and it responds NAK to all other packets. Because the USB core must access the BDT in Keep Alive mode, the 2 KB on-chip USB RAM located from 0x4010_0000 to 0x4010_07FF is used for the BDT and small data packet. Software puts the BDT to this RAM in Keep Alive mode and changes all OWN bits to 0 except the SETUP endpoint. This can be done by setting USB_KEEP_ALIVE_CTRL[OWN_OVERRD_EN].

47.6.1 Entering Keep Alive mode

The following are the steps to enter Keep Alive mode:

1. Enter either VLPS mode or Stop mode, ensuring that the USB 48 MHz functional clock source remains enabled in either mode:
   - To enter VLPS mode, set KEEP_ALIVE_CTRL[KEEP_ALIVE_EN]=1, CLK_RECOVER_IRC_EN[IRC_EN]=1, PMC_REGSC[BGEN]=1, and PMC_REGSC[VLPO]=1.
   - To enter Stop mode, set only KEEP_ALIVE_CTRL[KEEP_ALIVE_EN] = 1 and CLOCK_RECOVER_IRC_EN[IRC_EN]=1.
2. To initialize the BDT in USB RAM, within each BDT set OWN=0 for every endpoint except EP0. The buffer address for EP0 must be configured to an address within the USB SRAM, ensuring received data is placed within the USB SRAM. The USB cannot access other memory locations within USB Keep Alive mode.
3. Enable the wakeup interrupt by setting KEEP_ALIVE_CTRL[WAKE_INT_EN]=1.
4. Set KEEP_ALIVE_WKCTRL[WAKE_ON_THIS] appropriately to configure what received token will generate the wakeup from Keep Alive mode. Software then enters stop mode and the USB is in Keep Alive mode.

47.6.2 Exiting Keep Alive mode

Any enabled interrupt generates a wakeup from STOP/VLPS mode; this causes an exit from USB Keep Alive mode. The USB can also be configured to generate an interrupt when it receives a particular token (for example, SETUP), as configured by KEEP_ALIVE[WAKE_ON_THIS]. Other USB interrupts can also remain enabled and can be generated in USB Keep Alive mode.

47.7 OTG and Host mode operation

The Host mode logic allows portable, mobile, and other devices using this SOC to function as a USB Embedded Host (EH) 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 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. 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 Done 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.

47.8 Host Mode Operation Examples

The following sections illustrate the steps required to perform USB host functions using the USB-FS core. For more information about these procedures, see the *Universal Serial Bus Specification, Revision 2.0*, "Chapter 9 USB Device Framework."

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 (INTEN[ATTACHEN]=1).
3. Wait for ATTACH interrupt (ISTAT[ATTACH]). Signaled by USB peripheral device 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. Enumerate the attached device by sending the appropriate commands to the default control pipe of the connected device.

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.
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 peripheral 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 peripheral 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.
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 an IN or OUT token to Endpoint 0, the peripheral 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.
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 peripheral 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.

To send a full speed bulk data transfer to a peripheral 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 peripheral 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 peripheral 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 peripheral 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 peripheral device cleared. If a Reset interrupt occurs (SE0 for more than 2.5 μs), the peripheral device 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.

47.9 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 driver software to implement the Host Negotiation Protocol (HNP) and Session Request Protocol (SRP) and give access to the OTG protocol control signals. The following state machines show the OTG operations involved with HNP and SRP protocols from either end of the USB cable.

47.9.1 OTG dual role A device operation

A device is considered the A device because of the type of cable attached. If the USB Type Standard-A or Micro-A plug 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.

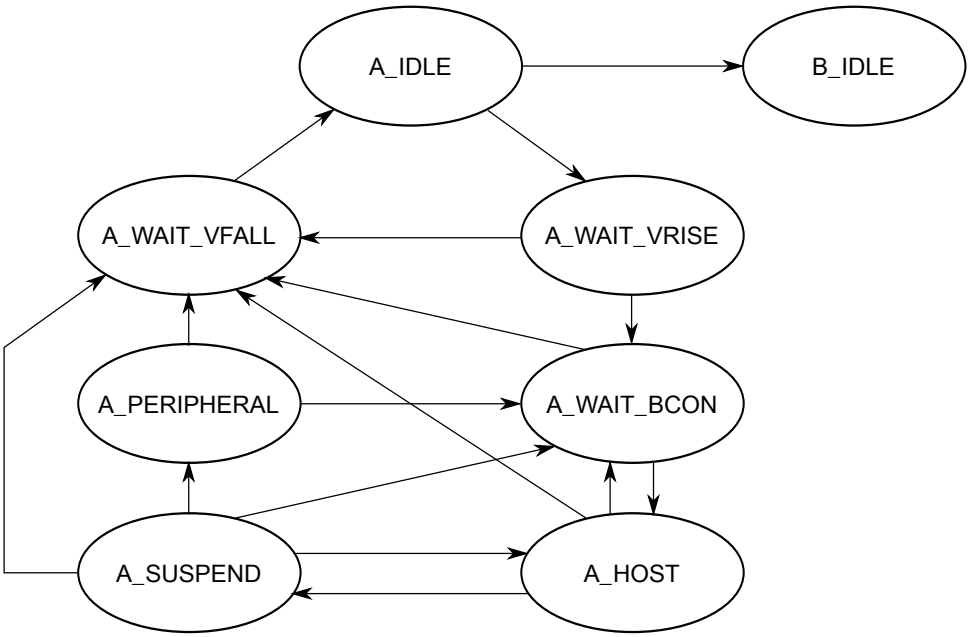


Figure 47-12. Dual role A device flow diagram

Table 47-8. State descriptions for the dual role A device flow

State	Action	Response
A_IDLE	If ID Interrupt. The cable has been unplugged 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 Turn on DRV_VBUS
A_WAIT_VRISE	If ID Interrupt or if A_VBUS_VLD is false after 100 msec The cable has been changed or the A device cannot support the current required from the B device.	Go to A_WAIT_VFALL Turn off DRV_VBUS
	If A_VBUS_VLD interrupt	Go to A_WAIT_BCON
A_WAIT_BCON	After 200 ms without Attach or ID Interrupt. (This could wait forever if desired.)	Go to A_WAIT_FALL Turn off DRV_VBUS
	A_VBUS_VLD Interrupt and B device attaches	Go to A_HOST Turn on Host mode
A_HOST	Enumerate Device determine OTG Support.	
	If A_VBUS_VLD/ Interrupt or A device is done and does not think it wants to do something soon or the B device disconnects	Go to A_WAIT_VFALL Turn off Host mode 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 Turn off DRV_VBUS

Table continues on the next page...

Table 47-8. State descriptions for the dual role A device flow (continued)

State	Action	Response
	If HNP enabled, and B disconnects in 150 ms then B device is becoming the host.	Go to A_PERIPHERAL 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 Turn off DRV_VBUS.
	If 3 –200 ms of Bus Idle	Go to A_WAIT_BCON Turn on Host mode
A_WAIT_VFALL	If ID Interrupt or (A_SESS_VLD/ & b_conn/)	Go to A_IDLE

47.9.2 OTG dual role B device operation

A device is considered a B device if it is connected to the bus with a USB Type Standard-B, Mini-B, or Micro-B plug inserted into the local USB receptacle. The Type Mini-B plug and receptacle are now only allowed for dedicated peripheral devices, not dual role/OTG devices.

A dual role B device operates as the following flow diagram and state description table illustrates.

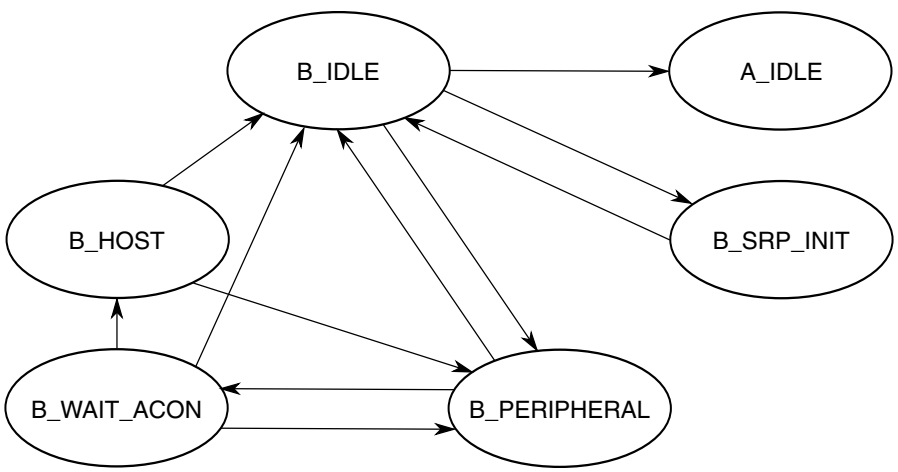


Figure 47-13. Dual role B device flow diagram

Table 47-9. State descriptions for the dual role B device flow

State	Action	Response
B_IDLE	If ID\ Interrupt. A Type A cable has been plugged in and the device should now respond as a Type A device.	Go to A_IDLE

Table continues on the next page...

Table 47-9. State descriptions for the dual role B device flow (continued)

State	Action	Response
	If B_SESS_VLD Interrupt. The A device has turned on VBUS and begins a session.	Go to B_PERIPHERAL 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 Pulse CHRG_VBUS 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 Turn off DP_HIGH
B_WAIT_ACON	If A connects, an attach interrupt is received	Go to B_HOST Turn on Host Mode
	If ID\ Interrupt or B_SESS_VLD/ Interrupt If the cable changes or if VBUS goes away, the host doesn't support us. 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 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

47.10 Device mode IRC48M operation

The following are the IRC48M initialization code steps:

1. Enable the IRC48M clock: USB_CLK_RECOVER_IRC_EN[IRC_EN] = 1b
2. Enable the USB clock recovery tuning: USB_CLK_RECOVER_CTRL[CLOCK_RECOVER_EN] = 1b
3. Choose the clock source of USB by configuring the muxes and dividers. The IRC48M is muxed by setting SIM_SOPT2[MCGPLLFLL]=11b for USB usage.
4. The selected mux output clock can be divided by the USB clock divider, so set these fields so no clock division is enabled. This is the equation for the divider:

   Divider output clock = Divider input clock × [(USBFRAC+1) / (USBDIV+1)].
   So set SIM_CLKDIV2[USBDIV] = 000b and SIM_CLKDIV2[USBFRAC] = 0b
5. The USB clock source must choose the output of the divided clock.

For chip-specific details, see the USB FS OTG controller clocking information in the "Clock Distribution" chapter.

Chapter 48
Serial Peripheral Interface (SPI)

48.1 Chip-specific SPI information

48.1.1 SPI Modules Configuration

The device contains two SPI modules.

48.1.2 SPI clocks

The SPI0/SPI1 is clocked by system clock.

The SPI can run at a maximum frequency of 24 MHz serial clocks on PORTE and 18 MHz on the other PORT interfaces.

48.1.3 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.

48.1.4 TX FIFO size

Table 48-1. SPI transmit FIFO size

SPI Module	Transmit FIFO size
SPI0	4
SPI1	1

48.1.5 RX FIFO Size

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

Table 48-2. SPI receive FIFO size

SPI Module	Receive FIFO size
SPI0	4
SPI1	1

48.1.6 Number of PCS signals

The following table shows the maximum number of peripheral chip select signals available per SPI module. Note that not all chip select signals are available in all device packages. Refer to KL82 signal multiplexing and pin assignments for additional information.

Table 48-3. SPI PCS signals

SPI Module	PCS Signals
SPI0	SPI0_PCS5
	SPI0_PCS4
	SPI0_PCS3
	SPI0_PCS2
	SPI0_PCS1
	SPI0_PCS0
SPI1	SPI1_PCS3
	SPI1_PCS2
	SPI1_PCS1
	SPI1_PCS0

48.1.7 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 4 MHz.

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.

48.1.8 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.

48.1.9 SPI Doze Mode

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

48.1.10 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.

48.1.11 Writing SPI Transmit FIFO

The SPI supports 8-bit or 16-bit writes to the PUSH TX FIFO, allowing a single write to the command word followed by multiple writes to the transmit word. The TX FIFO will save the last command word written, and convert a 8-bit/16-bit write to the transmit word into a 32-bit write that pushes both the command word and transmit word into the TX FIFO (PUSH TX FIFO Register In Master Mode (SPI_PUSHR))

A 32-bit write to the SPI_PUSH register will push all 32-bits to the TX FIFO. An 8-bit or 16-bit write to the 16-bit transmit data field will push the data together with the last written command word. An 8-bit or 16-bit write to the command word does not push data onto the FIFO, but that command word is pushed to the TX FIFO on all subsequent 8-bit or 16-bit writes to the transmit data field. This allows a single 16-bit write to the command word to be used for all subsequent 8-bit or 16-bit writes to the transmit data word. Writing a different 16-bit command word will cause all subsequent 8-bit or 16-bit writes to the transmit data word to be pushed to the TX FIFO with the new command word.

48.2 Introduction

The serial peripheral interface (SPI) module provides a synchronous serial bus for communication between a chip and an external peripheral device.

48.2.1 Block Diagram

The block diagram of this module is as follows:

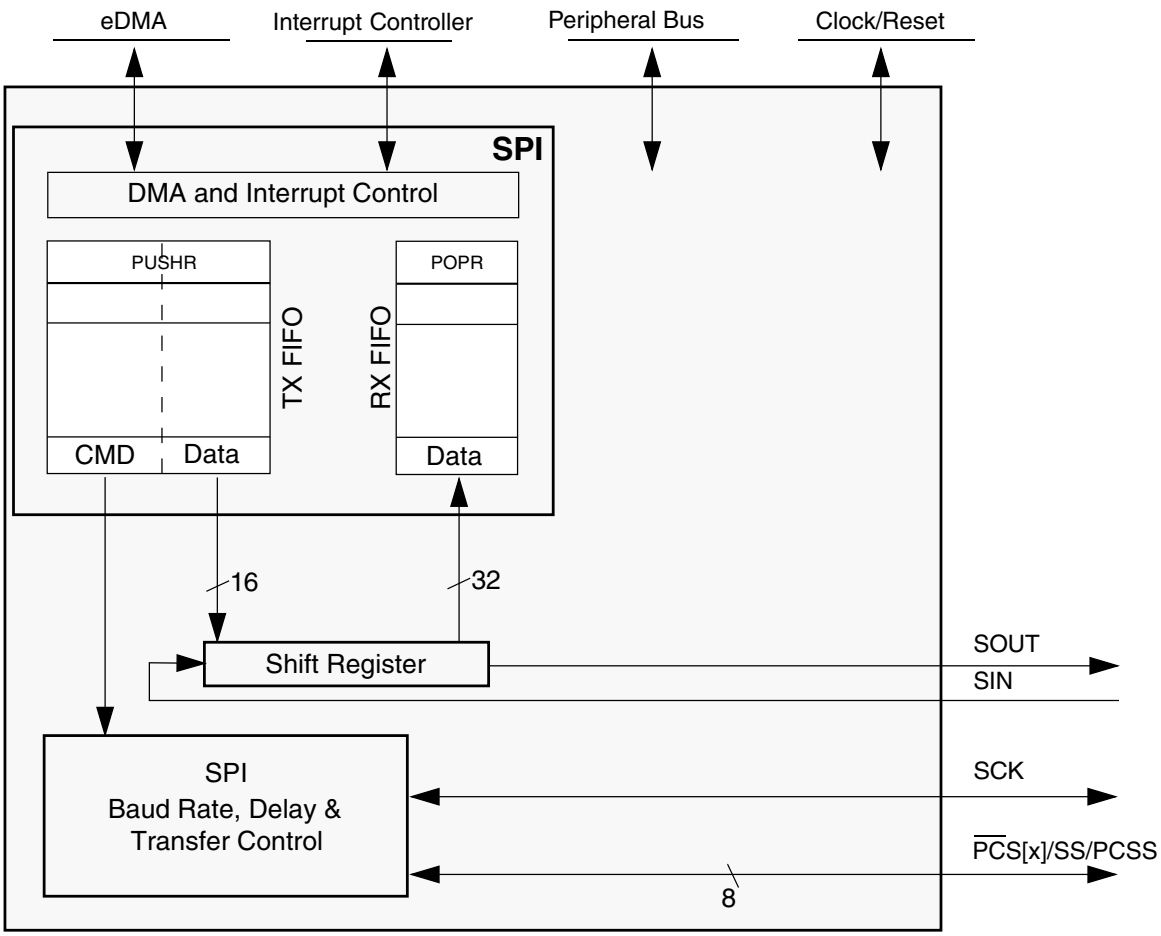


Figure 48-1. SPI Block Diagram

48.2.2 Features

The module supports the following features:

- Full-duplex, three-wire synchronous transfers
- Master mode
- Slave mode
- 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 4 entries
- Buffered receive operation using the receive FIFO (RX FIFO) with depth of 4 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: 4 to 16 bits
    - SPI frames longer than 16 bits can be supported using the continuous selection format.
  - Continuously held chip select capability
- 6 peripheral chip selects (PCSes), expandable to 64 with external demultiplexer
- Deglitching support for up to 32 peripheral chip selects (PCSes) 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)
- Modified SPI transfer formats for communication with slower peripheral devices
- Power-saving architectural features:
  - Support for Stop mode
  - Support for Doze mode

48.2.3 Interface configurations

48.2.3.1 SPI configuration

The Serial Peripheral Interface (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.

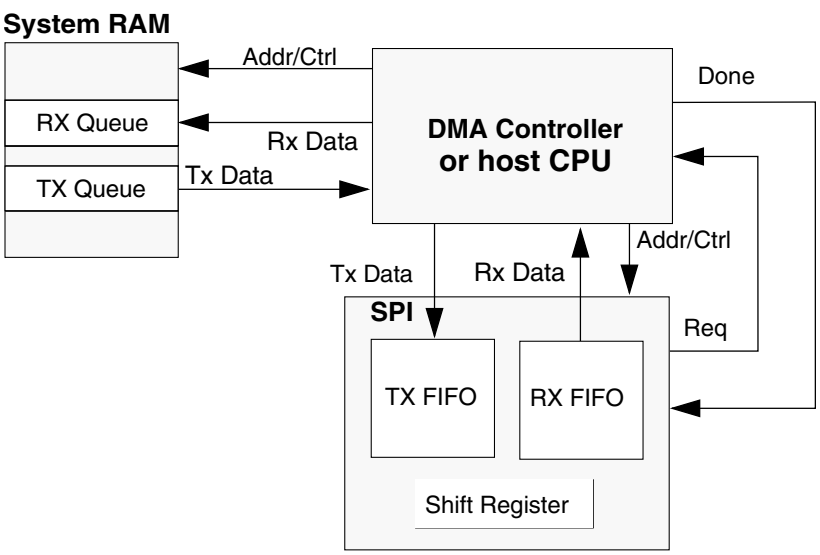


Figure 48-2. SPI with queues and DMA

48.2.4 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

- Chip-specific modes:
  - External Stop mode
  - Debug mode

The module enters module-specific modes when the host writes a module register. The chip-specific modes are controlled by signals external to the module. The chip-specific modes are modes that a chip may enter in parallel to the block-specific modes.

48.2.4.1 Master Mode

Master mode allows the module to initiate and control serial communication. In this mode, these signals are controlled by the module and configured as outputs:

- SCK
- SOUT
- PCS[*x*]

48.2.4.2 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]/$\overline{\text{SS}}$ signals are configured as inputs and driven by an SPI bus master.

48.2.4.3 Module Disable Mode

The Module Disable mode can be used for chip power management. The clock to the non-memory mapped logic in the module can be stopped while in the Module Disable mode.

48.2.4.4 External Stop Mode

External Stop mode is used for chip 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 protocol clock to the module may be shut off.

48.2.4.5 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 chip is in debug mode.
- If the bit is cleared, the chip debug mode has no effect on the module.

48.3 Module signal descriptions

This table describes the signals on the boundary of the module that may connect off chip (in alphabetical order).

Table 48-4. Module signal descriptions

Signal	Master mode	Slave mode	I/O
PCS0/$\overline{SS}$	Peripheral Chip Select 0 (O)	Slave Select (I)	I/O
PCS[1:3]	Peripheral Chip Selects 1–3	(Unused)	O
PCS4	Peripheral Chip Select 4	(Unused)	O
PCS5/ $\overline{PCSS}$	Peripheral Chip Select 5 /Peripheral Chip Select Strobe	(Unused)	O
SCK	Serial Clock (O)	Serial Clock (I)	I/O
SIN	Serial Data In	Serial Data In	I
SOUT	Serial Data Out	Serial Data Out	O

48.3.1 PCS0/$\overline{SS}$—Peripheral Chip Select/Slave Select

Master mode: Peripheral Chip Select 0 (O)—Selects an SPI slave to receive data transmitted from the module.

Slave mode: Slave Select (I)—Selects the module to receive data transmitted from an SPI master.

NOTE

Do not tie the SPI slave select pin to ground. Otherwise, SPI cannot function properly.

48.3.2 PCS1–PCS3—Peripheral Chip Selects 1–3

Master mode: Peripheral Chip Selects 1–3 (O)—Select an SPI slave to receive data transmitted by the module.

Slave mode: Unused

48.3.3 PCS4—Peripheral Chip Select 4

Master mode: Peripheral Chip Select 4 (O)—Selects an SPI slave to receive data transmitted by the module.

Slave mode: Unused

48.3.4 PCS5/$\overline{\text{PCSS}}$—Peripheral Chip Select 5/Peripheral Chip Select Strobe

Master mode:

- Peripheral Chip Select 5 (O)—Used only when the peripheral-chip-select strobe is disabled (MCR[PCSSE]). Selects an SPI slave to receive data transmitted by the module.
- Peripheral Chip Select Strobe (O)—Used only when the peripheral-chip-select strobe is enabled (MCR[PCSSE]). Strobes an off-module peripheral-chip-select demultiplexer, which decodes the module's PCS signals other than PCS5, preventing glitches on the demultiplexer outputs.

Slave mode: Unused

48.3.5 SCK—Serial Clock

Master mode: Serial Clock (O)—Supplies a clock signal from the module to SPI slaves.

Slave mode: Serial Clock (I)—Supplies a clock signal to the module from an SPI master.

48.3.6 SIN—Serial Input

Master mode: Serial Input (I)—Receives serial data.

Slave mode: Serial Input (I)—Receives serial data.

48.3.7 SOUT—Serial Output

Master mode: Serial Output (O)—Transmits serial data.

Slave mode: Serial Output (O)—Transmits serial data.

NOTE

Serial Data Out output buffers are controlled through SIU (or SIUL) and cannot be controlled through the module.

48.4 Memory Map/Register Definition

Register accesses to memory addresses that are reserved or undefined result in a transfer error. Write access to the POPR and RXFRn 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	48.4.1/1389
4002_C008	Transfer Count Register (SPI0_TCR)	32	R/W	0000_0000h	48.4.2/1392
4002_C00C	Clock and Transfer Attributes Register (In Master Mode) (SPI0_CTAR0)	32	R/W	7800_0000h	48.4.3/1393
4002_C00C	Clock and Transfer Attributes Register (In Slave Mode) (SPI0_CTAR0_SLAVE)	32	R/W	7800_0000h	48.4.4/1397
4002_C010	Clock and Transfer Attributes Register (In Master Mode) (SPI0_CTAR1)	32	R/W	7800_0000h	48.4.3/1393
4002_C02C	Status Register (SPI0_SR)	32	R/W	0200_0000h	48.4.5/1399
4002_C030	DMA/Interrupt Request Select and Enable Register (SPI0_RSER)	32	R/W	0000_0000h	48.4.6/1402
4002_C034	PUSH TX FIFO Register In Master Mode (SPI0_PUSHR)	32	R/W	0000_0000h	48.4.7/1404
4002_C034	PUSH TX FIFO Register In Slave Mode (SPI0_PUSHR_SLAVE)	32	R/W	0000_0000h	48.4.8/1406
4002_C038	POP RX FIFO Register (SPI0_POPR)	32	R	0000_0000h	48.4.9/1406
4002_C03C	Transmit FIFO Registers (SPI0_TXFR0)	32	R	0000_0000h	48.4.10/ 1407

Table continues on the next page...

SPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_C040	Transmit FIFO Registers (SPI0_TXFR1)	32	R	0000_0000h	48.4.10/ 1407
4002_C044	Transmit FIFO Registers (SPI0_TXFR2)	32	R	0000_0000h	48.4.10/ 1407
4002_C048	Transmit FIFO Registers (SPI0_TXFR3)	32	R	0000_0000h	48.4.10/ 1407
4002_C07C	Receive FIFO Registers (SPI0_RXFR0)	32	R	0000_0000h	48.4.11/ 1407
4002_C080	Receive FIFO Registers (SPI0_RXFR1)	32	R	0000_0000h	48.4.11/ 1407
4002_C084	Receive FIFO Registers (SPI0_RXFR2)	32	R	0000_0000h	48.4.11/ 1407
4002_C088	Receive FIFO Registers (SPI0_RXFR3)	32	R	0000_0000h	48.4.11/ 1407
4002_D000	Module Configuration Register (SPI1_MCR)	32	R/W	0000_4001h	48.4.1/1389
4002_D008	Transfer Count Register (SPI1_TCR)	32	R/W	0000_0000h	48.4.2/1392
4002_D00C	Clock and Transfer Attributes Register (In Master Mode) (SPI1_CTAR0)	32	R/W	7800_0000h	48.4.3/1393
4002_D00C	Clock and Transfer Attributes Register (In Slave Mode) (SPI1_CTAR0_SLAVE)	32	R/W	7800_0000h	48.4.4/1397
4002_D010	Clock and Transfer Attributes Register (In Master Mode) (SPI1_CTAR1)	32	R/W	7800_0000h	48.4.3/1393
4002_D02C	Status Register (SPI1_SR)	32	R/W	0200_0000h	48.4.5/1399
4002_D030	DMA/Interrupt Request Select and Enable Register (SPI1_RSER)	32	R/W	0000_0000h	48.4.6/1402
4002_D034	PUSH TX FIFO Register In Master Mode (SPI1_PUSHR)	32	R/W	0000_0000h	48.4.7/1404
4002_D034	PUSH TX FIFO Register In Slave Mode (SPI1_PUSHR_SLAVE)	32	R/W	0000_0000h	48.4.8/1406
4002_D038	POP RX FIFO Register (SPI1_POPR)	32	R	0000_0000h	48.4.9/1406
4002_D03C	Transmit FIFO Registers (SPI1_TXFR0)	32	R	0000_0000h	48.4.10/ 1407
4002_D040	Transmit FIFO Registers (SPI1_TXFR1)	32	R	0000_0000h	48.4.10/ 1407
4002_D044	Transmit FIFO Registers (SPI1_TXFR2)	32	R	0000_0000h	48.4.10/ 1407
4002_D048	Transmit FIFO Registers (SPI1_TXFR3)	32	R	0000_0000h	48.4.10/ 1407
4002_D07C	Receive FIFO Registers (SPI1_RXFR0)	32	R	0000_0000h	48.4.11/ 1407
4002_D080	Receive FIFO Registers (SPI1_RXFR1)	32	R	0000_0000h	48.4.11/ 1407
4002_D084	Receive FIFO Registers (SPI1_RXFR2)	32	R	0000_0000h	48.4.11/ 1407

Table continues on the next page...

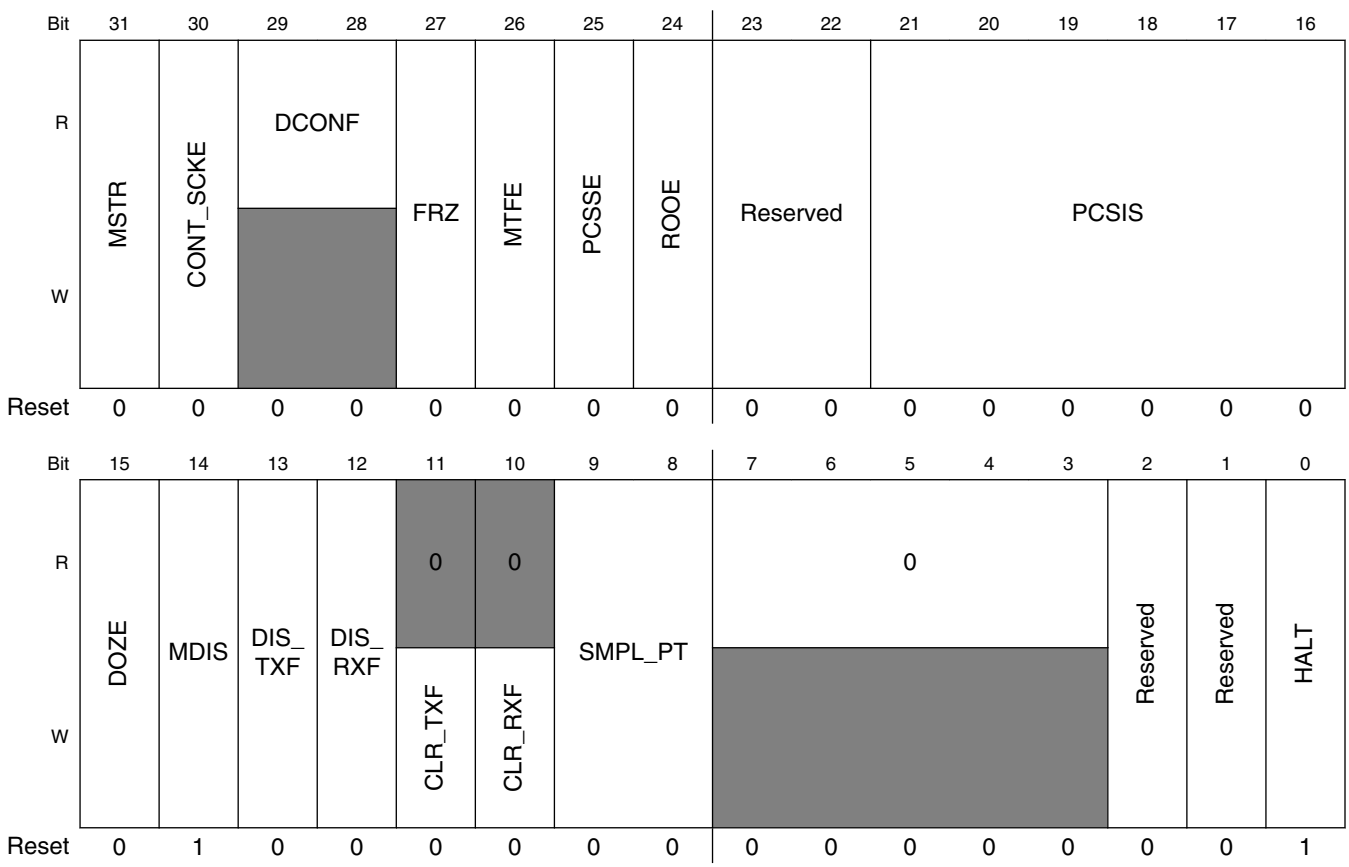
SPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_D088	Receive FIFO Registers (SPI1_RXFR3)	32	R	0000_0000h	48.4.11/ 1407

48.4.1 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 MSTR	Master/Slave Mode Select

Table continues on the next page...

SPIx_MCR field descriptions (continued)

Field	Description
	Enables either Master mode (if supported) or Slave mode (if supported) operation. 0 Enables Slave mode 1 Enables Master mode
30 CONT_SCKE	Continuous SCK Enable Enables the Serial Communication Clock (SCK) to run continuously. 0 Continuous SCK disabled. 1 Continuous SCK enabled.
29–28 DCONF	SPI Configuration. Selects among the different configurations of the module. 00 SPI 01 Reserved 10 Reserved 11 Reserved
27 FRZ	Freeze Enables transfers to be stopped on the next frame boundary when the device enters Debug mode. 0 Do not halt serial transfers in Debug mode. 1 Halt serial transfers in Debug mode.
26 MTFE	Modified Transfer Format Enable Enables a modified transfer format to be used. 0 Modified SPI transfer format disabled. 1 Modified SPI transfer format enabled.
25 PCSSE	Peripheral Chip Select Strobe Enable Enables the PCS5/ $\overline{\text{PCSS}}$ to operate as a PCS Strobe output signal. 0 PCS5/ $\overline{\text{PCSS}}$ is used as the Peripheral Chip Select[5] signal. 1 PCS5/ $\overline{\text{PCSS}}$ is used as an active-low PCS Strobe signal.
24 ROOE	Receive FIFO Overflow Overwrite Enable 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. 0 Incoming data is ignored. 1 Incoming data is shifted into the shift register.
23–22 Reserved	Always write the reset value to this field. This field is reserved.
21–16 PCSIS	Peripheral Chip Select x Inactive State Determines the inactive state of PCSx. Refer to the chip-specific SPI information for the number of PCS signals used in this chip.

Table continues on the next page...

SPIx_MCR field descriptions (continued)

Field	Description
	NOTE: The effect of this bit only takes place when module is enabled. Ensure that this bit is configured correctly before enabling the DSPI interface. 0 The inactive state of PCSx is low. 1 The inactive state of PCSx is high.
15 DOZE	Doze Enable Provides support for an externally controlled Doze mode power-saving mechanism. 0 Doze mode has no effect on the module. 1 Doze mode disables the module.
14 MDIS	Module Disable 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 1. When the module is used in Slave Mode, it is recommended to leave this bit 0, because a slave doesn't have control over master transactions. 0 Enables the module clocks. 1 Allows external logic to disable the module clocks.
13 DIS_TXF	Disable Transmit FIFO 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. 0 TX FIFO is enabled. 1 TX FIFO is disabled.
12 DIS_RXF	Disable Receive FIFO 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. 0 RX FIFO is enabled. 1 RX FIFO is disabled.
11 CLR_TXF	Clear TX FIFO Flushes the TX FIFO. Writing a 1 to CLR_TXF clears the TX FIFO Counter. The CLR_TXF bit is always read as zero. 0 Do not clear the TX FIFO counter. 1 Clear the TX FIFO counter.
10 CLR_RXF	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. 0 Do not clear the RX FIFO counter. 1 Clear the RX FIFO counter.
9–8 SMPL_PT	Sample Point Controls when the module master samples SIN in Modified Transfer Format. This field is valid only when CPHA bit in CTARn[CPHA] is 0.

Table continues on the next page...

SPIx_MCR field descriptions (continued)

Field	Description
	00 0 protocol clock cycles between SCK edge and SIN sample 01 1 protocol clock cycle between SCK edge and SIN sample 10 2 protocol clock cycles between SCK edge and SIN sample 11 Reserved
7–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 Reserved	This field is reserved.
1 Reserved	This field is reserved.
0 HALT	Halt The HALT bit starts and stops frame transfers. See Start and Stop of Module transfers 0 Start transfers. 1 Stop transfers.

48.4.2 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 SPI_TCNT	SPI Transfer Counter 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.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.3 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, it uses the CTAR0 register.

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 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 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 DBR	Double Baud Rate 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 48-5. 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 48-5. SPI SCK Duty Cycle (continued) DBR / CPHA / PBR / SCK Duty Cycle 1 / 0 / 11 / 43/57 1 / 1 / 00 / 50/50 1 / 1 / 01 / 66/33 1 / 1 / 10 / 60/40 1 / 1 / 11 / 57/43 0 The baud rate is computed normally with a 50/50 duty cycle. 1 The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler.
30–27 FMSZ	Frame Size The number of bits transferred per frame is equal to the FMSZ value plus 1. Regardless of the transmission mode, the minimum valid frame size value is 4.
26 CPOL	Clock Polarity 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. **NOTE:** In case of Continuous SCK mode, when the module goes in low power mode(disabled), inactive state of SCK is not guaranted. 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
25 CPHA	Clock Phase 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. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
24 LSBFE	LSB First Specifies whether the LSB or MSB of the frame is transferred first. 0 Data is transferred MSB first. 1 Data is transferred LSB first.
23–22 PCSSCK	PCS to SCK Delay Prescaler 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.

Table continues on the next page...

SPIx_CTAR*n* field descriptions (continued)

Field	Description
	00 PCS to SCK Prescaler value is 1. 01 PCS to SCK Prescaler value is 3. 10 PCS to SCK Prescaler value is 5. 11 PCS to SCK Prescaler value is 7.
21–20 PASC	After SCK Delay Prescaler 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. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.
19–18 PDT	Delay after Transfer Prescaler 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. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.
17–16 PBR	Baud Rate Prescaler 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 protocol 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. 00 Baud Rate Prescaler value is 2. 01 Baud Rate Prescaler value is 3. 10 Baud Rate Prescaler value is 5. 11 Baud Rate Prescaler value is 7.
15–12 CSSCK	PCS to SCK Delay Scaler 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 protocol clock period, and it is computed according to the following equation: $t_{CSC} = (1/f_P) \times PCSSCK \times CSSCK$. The following table lists the delay scaler values.

Table 48-6. Delay Scaler Encoding

Field Value	Delay Scaler Value
0000	2
0001	4
0010	8

Table continues on the next page...

SPIx_CTAR*n* field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
Table 48-6. Delay Scaler Encoding (continued)
<table>
<tr><th>Field Value</th><th>Delay Scaler Value</th></tr>
<tr><td>0011</td><td>16</td></tr>
<tr><td>0100</td><td>32</td></tr>
<tr><td>0101</td><td>64</td></tr>
<tr><td>0110</td><td>128</td></tr>
<tr><td>0111</td><td>256</td></tr>
<tr><td>1000</td><td>512</td></tr>
<tr><td>1001</td><td>1024</td></tr>
<tr><td>1010</td><td>2048</td></tr>
<tr><td>1011</td><td>4096</td></tr>
<tr><td>1100</td><td>8192</td></tr>
<tr><td>1101</td><td>16384</td></tr>
<tr><td>1110</td><td>32768</td></tr>
<tr><td>1111</td><td>65536</td></tr>
</table>
Refer PCS to SCK Delay (t_{CSC}) for more details.
</td></tr>
<tr><td>11–8
ASC</td><td>After SCK Delay Scaler

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 protocol clock period, and it is computed according to the following equation:

$t_{ASC} = (1/f_P) \times PASC \times ASC$

See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. Refer After SCK Delay (t_{ASC}) for more details.</td></tr>
<tr><td>7–4
DT</td><td>Delay After Transfer Scaler

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.

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 protocol clock period, and it is computed according to the following equation:

$t_{DT} = (1/f_P) \times PDT \times DT$

See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values.</td></tr>
<tr><td>BR</td><td>Baud Rate Scaler

Selects the scaler value for the baud rate. This field is used only in master mode. The prescaled protocol 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:

SCK baud rate = (f_P /PBR) x [(1+DBR)/BR]

The following table lists the baud rate scaler values.</td></tr>
</table>

Table continues on the next page...

SPIx_CTAR*n* field descriptions (continued)

Field	Description

Table 48-7. 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

48.4.4 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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	FMSZ				CPOL	CPHA	0		Reserved	Reserved					
W																
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	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPIx_CTAR*n*_SLAVE field descriptions

Field	Description
31 Reserved	Always write the reset value to this field. This field is reserved.
30–27 FMSZ	Frame Size 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 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). **NOTE:** In case of Continuous SCK mode, when the module goes in low power mode(disabled), inactive state of SCK is not guaranted. 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
25 CPHA	Clock Phase 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. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
24–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 Reserved	This field is reserved.
Reserved	This field is reserved.

48.4.5 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 TCF	Transfer Complete Flag Indicates that all bits in a frame have been shifted out. TCF remains set until it is cleared by writing a 1 to it. 0 Transfer not complete. 1 Transfer complete.
30 TXRXS	TX and RX Status Reflects the run status of the module. 0 Transmit and receive operations are disabled (The module is in Stopped state). 1 Transmit and receive operations are enabled (The module is in Running state).
29 Reserved	This field is reserved. 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 EOQF	End of Queue Flag 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. 0 EOQ is not set in the executing command. 1 EOQ is set in the executing SPI command.
27 TFUF	Transmit FIFO Underflow Flag 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. 0 No TX FIFO underflow. 1 TX FIFO underflow has occurred.
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 TFFF	Transmit FIFO Fill Flag 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. **NOTE:** The reset value of this bit is 0 when the module is disabled,(MCR[MDIS]=1). 0 TX FIFO is full. 1 TX FIFO is not full.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 RFOF	Receive FIFO Overflow Flag 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. 0 No Rx FIFO overflow. 1 Rx FIFO overflow has occurred.
18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 RFDF	Receive FIFO Drain Flag

Table continues on the next page...

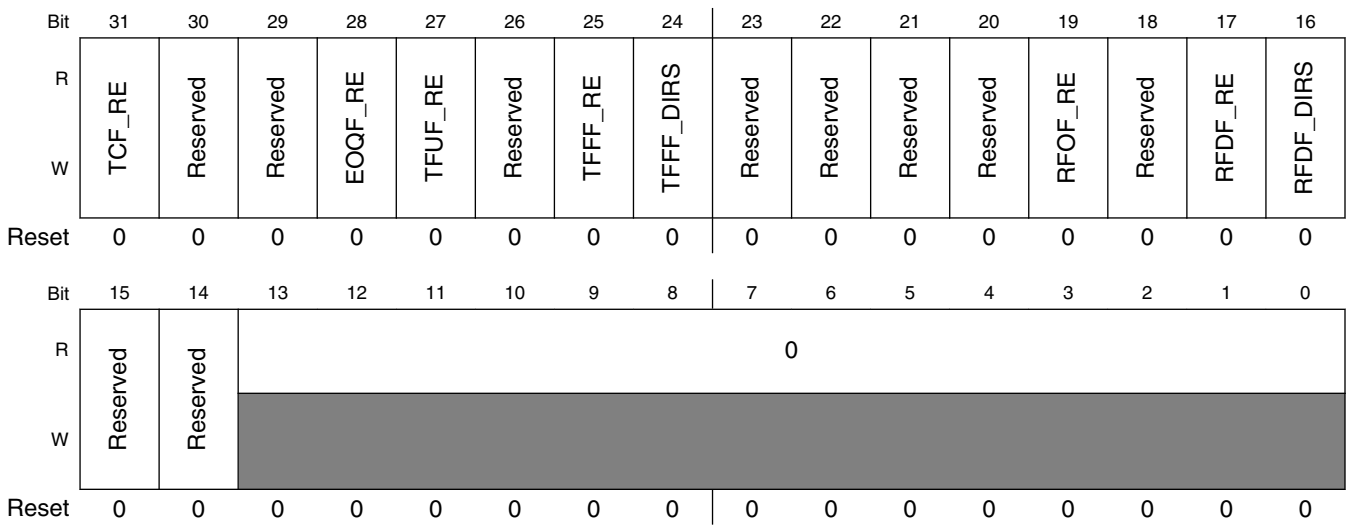
SPIx_SR field descriptions (continued)

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. 0 RX FIFO is empty. 1 RX FIFO is not empty.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–12 TXCTR	TX FIFO Counter 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 TXNXTPTR	Transmit Next Pointer 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 RXCTR	RX FIFO Counter 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.
POPNXTPTR	Pop Next Pointer 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.

48.4.6 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 TCF_RE	Transmission Complete Request Enable Enables TCF flag in the SR to generate an interrupt request. 0 TCF interrupt requests are disabled. 1 TCF interrupt requests are enabled.
30 Reserved	Always write the reset value to this field. This field is reserved.
29 Reserved	Always write the reset value to this field. This field is reserved.
28 EOQF_RE	Finished Request Enable Enables the EOQF flag in the SR to generate an interrupt request. 0 EOQF interrupt requests are disabled. 1 EOQF interrupt requests are enabled.
27 TFUF_RE	Transmit FIFO Underflow Request Enable Enables the TFUF flag in the SR to generate an interrupt request.

Table continues on the next page...

SPIx_RSER field descriptions (continued)

Field	Description
	0 TFUF interrupt requests are disabled. 1 TFUF interrupt requests are enabled.
26 Reserved	Always write the reset value to this field. This field is reserved.
25 TFFF_RE	Transmit FIFO Fill Request Enable 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. 0 TFFF interrupts or DMA requests are disabled. 1 TFFF interrupts or DMA requests are enabled.
24 TFFF_DIRS	Transmit FIFO Fill DMA or Interrupt Request Select 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. 0 TFFF flag generates interrupt requests. 1 TFFF flag generates DMA requests.
23 Reserved	Always write the reset value to this field. This field is reserved.
22 Reserved	Always write the reset value to this field. This field is reserved.
21 Reserved	Always write the reset value to this field. This field is reserved.
20 Reserved	Always write the reset value to this field. This field is reserved.
19 RFOF_RE	Receive FIFO Overflow Request Enable Enables the RFOF flag in the SR to generate an interrupt request. 0 RFOF interrupt requests are disabled. 1 RFOF interrupt requests are enabled.
18 Reserved	Always write the reset value to this field. This field is reserved.
17 RFDF_RE	Receive FIFO Drain Request Enable 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. 0 RFDF interrupt or DMA requests are disabled. 1 RFDF interrupt or DMA requests are enabled.
16 RFDF_DIRS	Receive FIFO Drain DMA or Interrupt Request Select

Table continues on the next page...

SPIx_RSER field descriptions (continued)

Field	Description
	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. 0 Interrupt request. 1 DMA request.
15 Reserved	Always write the reset value to this field. This field is reserved.
14 Reserved	Always write the reset value to this field. This field is reserved.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.7 PUSH TX FIFO Register In Master Mode (SPIx_PUSHR)

Specifies data to be transferred to the TX FIFO. An 8- or 16-bit write access transfers all 32 bits to the TX FIFO. In Master mode, the register transfers 16 bits of data and 16 bits of command information. A read access of PUSHR returns the topmost TX FIFO entry.

When the module is disabled, writing to this register does not update the FIFO. Therefore, any reads performed while the module is disabled return the last PUSHR write performed while the module was still enabled.

Address: Base address + 34h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CONT	CTAS			EOQ	CTCNT	Reserved		Reserved		PCS					
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 W	TXDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPIx_PUSHR field descriptions

Field	Description
31 CONT	Continuous Peripheral Chip Select Enable 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. 0 Return PCSn signals to their inactive state between transfers. 1 Keep PCSn signals asserted between transfers.
30–28 CTAS	Clock and Transfer Attributes Select 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 chip specific section for details to determine how many CTARs this device has. You should not program a value in this field for a register that is not present. 000 CTAR0 001 CTAR1 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
27 EOQ	End Of Queue 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. 0 The SPI data is not the last data to transfer. 1 The SPI data is the last data to transfer.
26 CTCNT	Clear Transfer Counter Clears the TCNT field in the TCR register. The TCNT field is cleared before the module starts transmitting the current SPI frame. 0 Do not clear the TCR[TCNT] field. 1 Clear the TCR[TCNT] field.
25–24 Reserved	Always write the reset value to this field. This field is reserved.
23–22 Reserved	Always write the reset value to this field. This field is reserved.
21–16 PCS	Select which PCS signals are to be asserted for the transfer. Refer to the chip-specific SPI information for the number of PCS signals used in this chip. 0 Negate the PCS[x] signal. 1 Assert the PCS[x] signal.
TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command.

48.4.8 PUSH TX FIFO Register In Slave Mode (SPIx_PUSHR_SLAVE)

Specifies data to be transferred to the TX FIFO in slave mode. An 8- or 16-bit write access to PUSHR transfers the 16-bit TXDATA field to the TX FIFO.

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 W	Reserved																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
31–16 Reserved	This field is reserved.
TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command.

48.4.9 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
RXDATA	Received Data Contains the SPI data from the RX FIFO entry to which the Pop Next Data Pointer points.

48.4.10 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

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	TXCMD_TXDATA																TXDATA															
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_TXFR*n* field descriptions

Field	Description
31–16 TXCMD_ TXDATA	Transmit Command or Transmit Data In Master mode the TXCMD field contains the command that sets the transfer attributes for the SPI data. In Slave mode, this field is reserved.
TXDATA	Transmit Data Contains the SPI data to be shifted out.

48.4.11 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

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_RXFR*n* field descriptions

Field	Description
RXDATA	Receive Data Contains the received SPI data.

SPIx_RXFR*n* field descriptions (continued)

Field	Description

48.5 Functional description

The module supports full-duplex, synchronous serial communications between chips and peripheral devices. The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes.

The module has the following configuration

- 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.

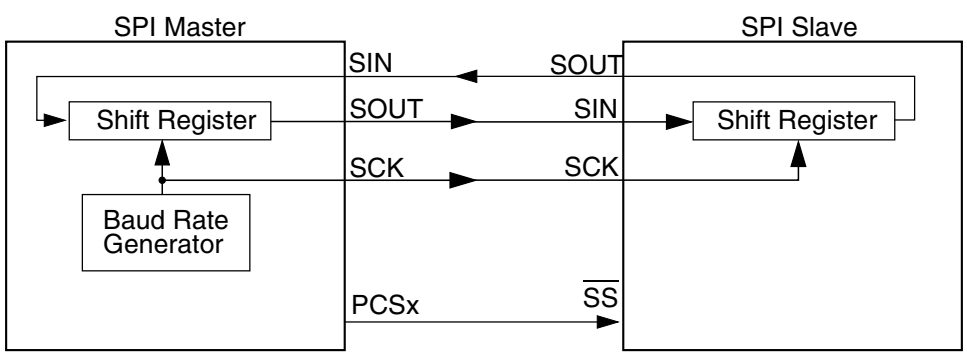


Figure 48-3. Serial protocol overview

Generally, more than one slave device can be connected to the module master. 6 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 specific section for details on the number of PCS signals used in this chip.

The SPI configuration shares 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.

48.5.1 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
- Chip 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
- Chip 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.

48.5.2 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 the module 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 the following sections:

- Transmit First In First Out (TX FIFO) buffering mechanism
- Transmit First In First Out (TX FIFO) buffering mechanism
- 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.

48.5.2.1 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.

48.5.2.2 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.

48.5.2.3 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:
- SR[TFFF], SR[TFUF] and SR[TXCTR] behave as if there is a one-entry FIFO
- 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.

48.5.2.4 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 4 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 PUSHR[TXDATA] 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.

48.5.2.4.1 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.

48.5.2.4.2 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.

48.5.2.5 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 4 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.

48.5.2.5.1 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.

48.5.2.5.2 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.

48.5.3 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.

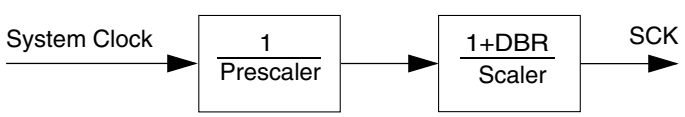


Figure 48-4. Communications clock prescalers and scalers

48.5.3.1 Baud rate generator

The baud rate is the frequency of the SCK. The protocol 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 48-8. Baud rate computation example

f_P	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.

48.5.3.2 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 48-6 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 48-9. 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.

48.5.3.3 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 48-6 and Figure 48-7 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 48-10. After SCK Delay computation example

f_P	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.

48.5.3.4 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 48-6 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 48-11. Delay after Transfer computation example

f_P	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.

48.5.3.5 Peripheral Chip Select Strobe Enable ($\overline{\text{PCSS}}$)

The $\overline{\text{PCSS}}$ 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, $\overline{\text{PCSS}}$ provides a signal for an external demultiplexer to decode peripheral chip selects other than PCS5 into glitch-free PCS signals. The following figure shows the timing of the $\overline{\text{PCSS}}$ signal relative to PCS signals.

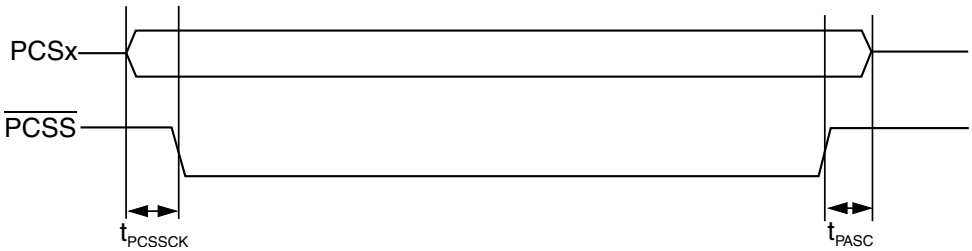


Figure 48-5. Peripheral Chip Select Strobe timing

The delay between the assertion of the PCS signals and the assertion of $\overline{\text{PCSS}}$ is selected by the PCSSCK field in the CTAR based on the following formula:

$$t_{\text{PCSSCK}} = \frac{1}{f_{\text{P}}} \times \text{PCSSCK}$$

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:

$$t_{\text{PASC}} = \frac{1}{f_{\text{P}}} \times \text{PASC}$$

The following table shows an example of how to compute the t_{pcssck} delay.

Table 48-12. Peripheral Chip Select Strobe Assert computation example

f_P	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 48-13. Peripheral Chip Select Strobe Negate computation example

f_P	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.

48.5.4 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 interface configurations, the module provides the option of keeping the PCS signals asserted between frames. See Continuous Selection Format for details.

48.5.4.1 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.

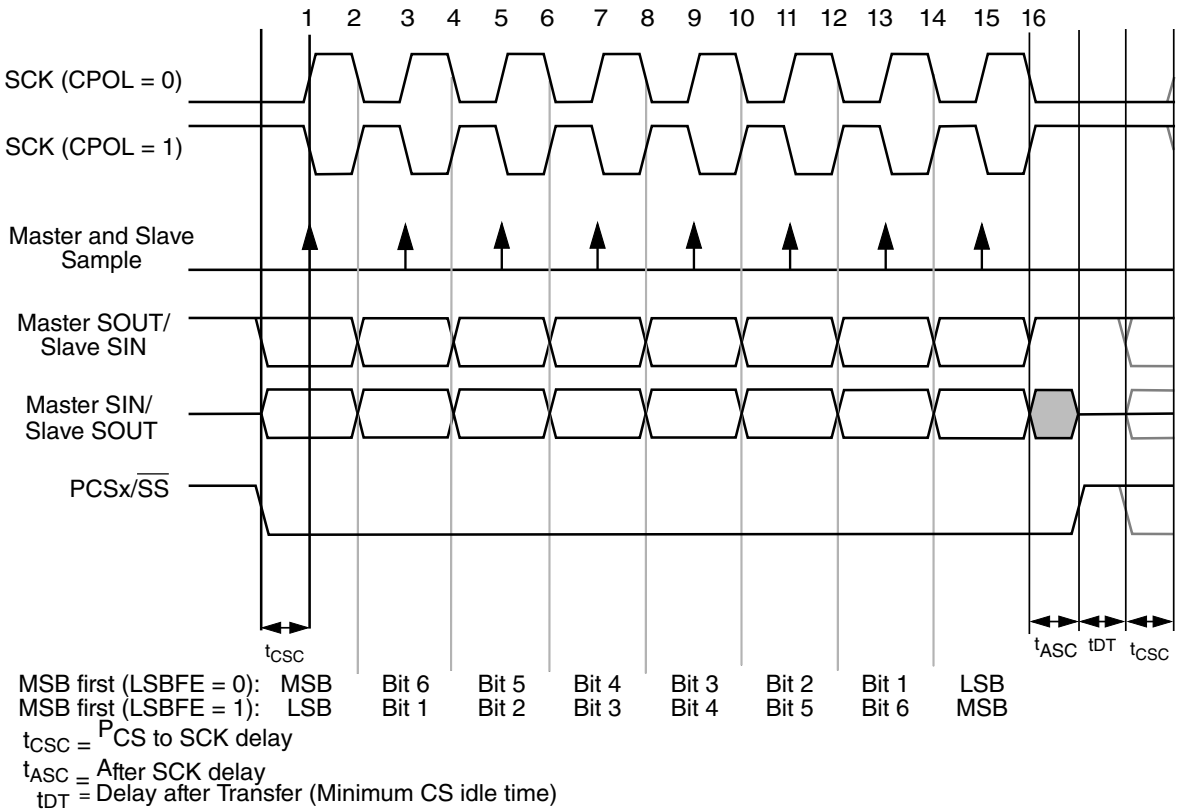


Figure 48-6. Module transfer timing diagram (MTFE=0, CPHA=0, FMSZ=8)

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_{CSC} 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.

48.5.4.2 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.

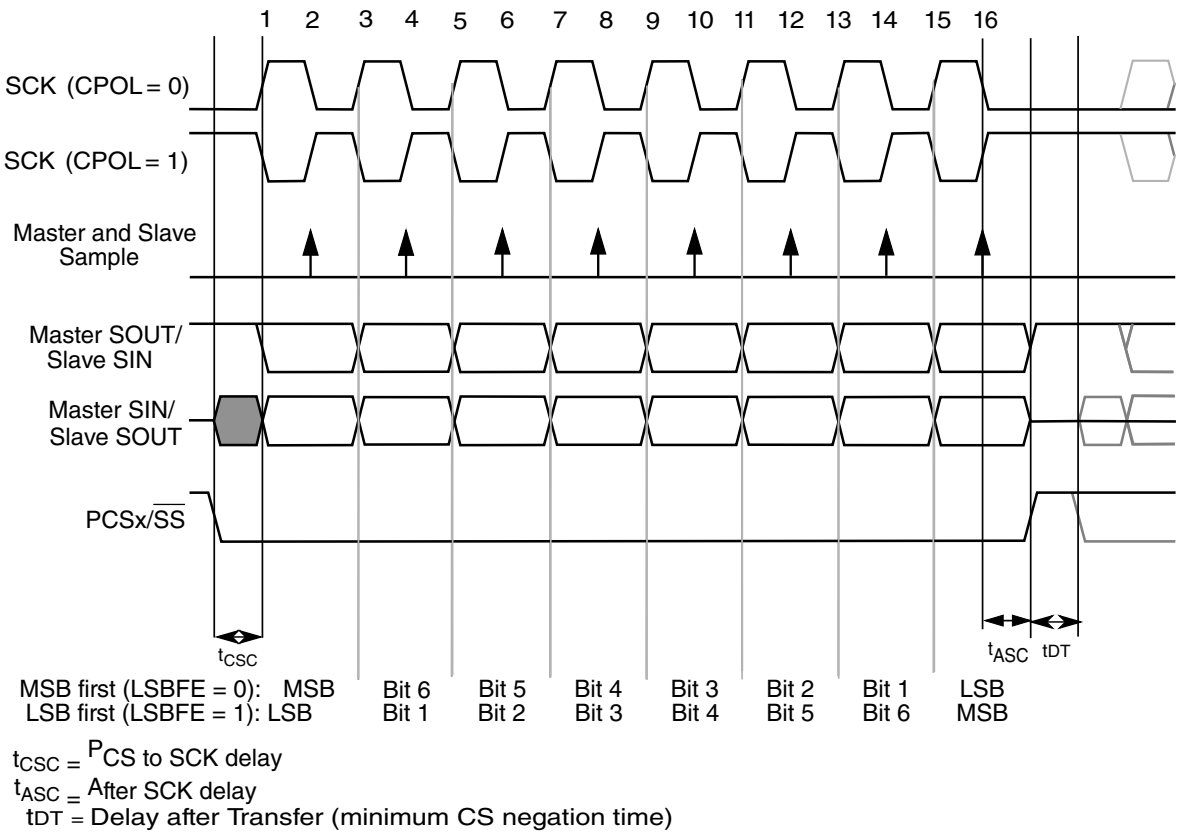


Figure 48-7. Module transfer timing diagram (MTFE=0, CPHA=1, FMSZ=8)

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.

48.5.4.3 Modified SPI Transfer Format (MTFE = 1, CPHA = 0)

In this Modified Transfer Format both the master and the slave sample later in the SCK period than in Classic SPI mode to allow the logic to tolerate more delays in device pads and board traces. These delays become a more significant fraction of the SCK period as the SCK period decreases with increasing baud rates.

The master and the slave place data on the SOUT pins at the assertion of the PCS signal. After the PCS to SCK delay has elapsed the first SCK edge is generated. The slave samples the master SOUT signal on every odd numbered SCK edge. The DSPI in the slave mode when the MTFE bit is set also places new data on the slave SOUT on every odd numbered clock edge. Regular external slave, configured with CPHA=0 format drives its SOUT output at every even numbered SCK clock edge.

The DSPI master places its second data bit on the SOUT line one protocol clock after odd numbered SCK edge if the protocol clock frequency to SCK frequency ratio is higher than three. If this ratio is below four the master changes SOUT at odd numbered SCK edge. The point where the master samples the SIN is selected by the DSPI_MCR[SMPL_PT] field. The master sample point can be delayed by one or two protocol clock cycles. The SMPL_PT field should be set to 0 if the protocol to SCK frequency ratio is less than 4. However if this ratio is less than 4, the actual sample point is delayed by one protocol clock cycle automatically by the design.

The following timing diagrams illustrate the DSPI operation with MTFE=1. Timing delays shown are:

- T_{csc} - PCS to SCK assertion delay
- T_{acs} - After SCK PCS negation delay
- T_{su_ms} - master SIN setup time
- T_{hd_ms} - master SIN hold time
- T_{vd_sl} - slave data output valid time, time between slave data output SCK driving edge and data becomes valid.
- T_{su_sl} - data setup time on slave data input
- T_{hd_sl} - data hold time on slave data input
- T_{sys} - protocol clock period.

The following figure shows the modified transfer format for CPHA = 0 and Fsys/Fsck = 4. Only the condition where CPOL = 0 is illustrated. Solid triangles show the data sampling clock edges. The two possible slave behavior are shown.

- Signal, marked "SOUT of Ext Slave", presents regular SPI slave serial output.
- Signal, marked "SOUT of DSPI Slave", presents DSPI in the slave mode with MTFE bit set.

Other MTFE = 1 diagrams show DSPI SIN input as being driven by a regular external SPI slave, configured according DSPI master CPHA programming.

Note

In the following diagrams, f_{sys} represents the protocol clock frequency from which the Baud frequency f_{sck} is derived.

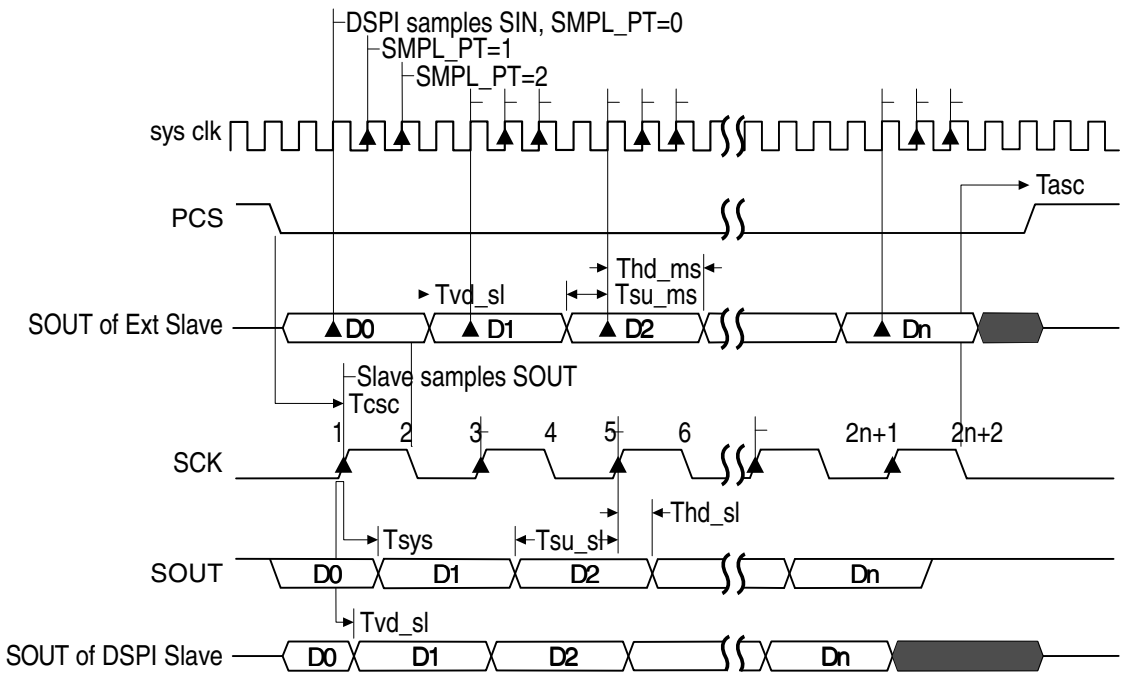


Figure 48-8. DSPI Modified Transfer Format (MTFE=1, CPHA=0, $f_{sck} = f_{sys}/4$)

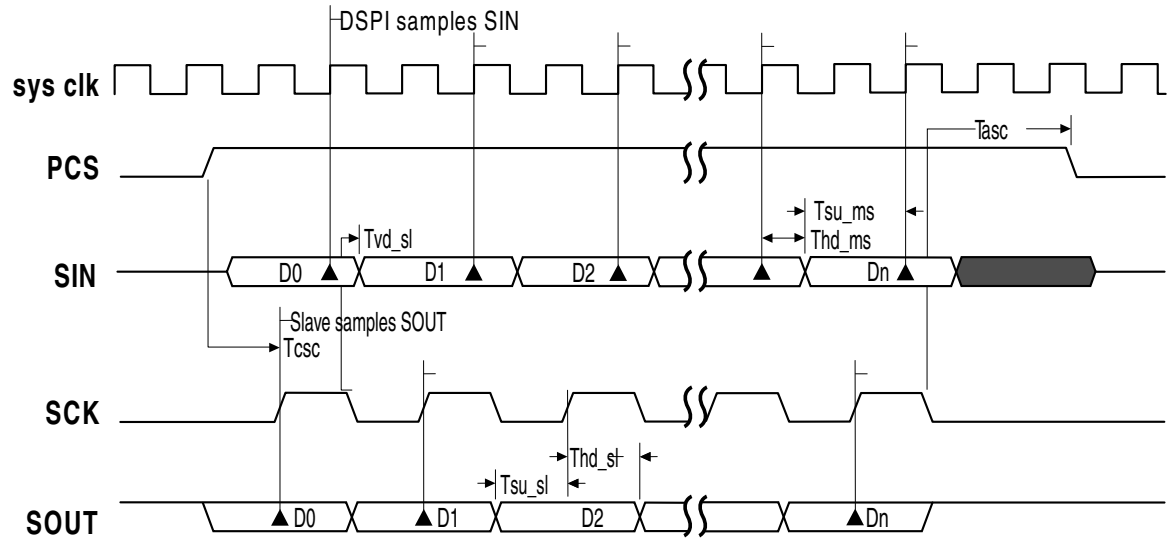


Figure 48-9. DSPI Modified Transfer Format (MTFE=1, CPHA=0, $f_{sck} = f_{sys}/2$)

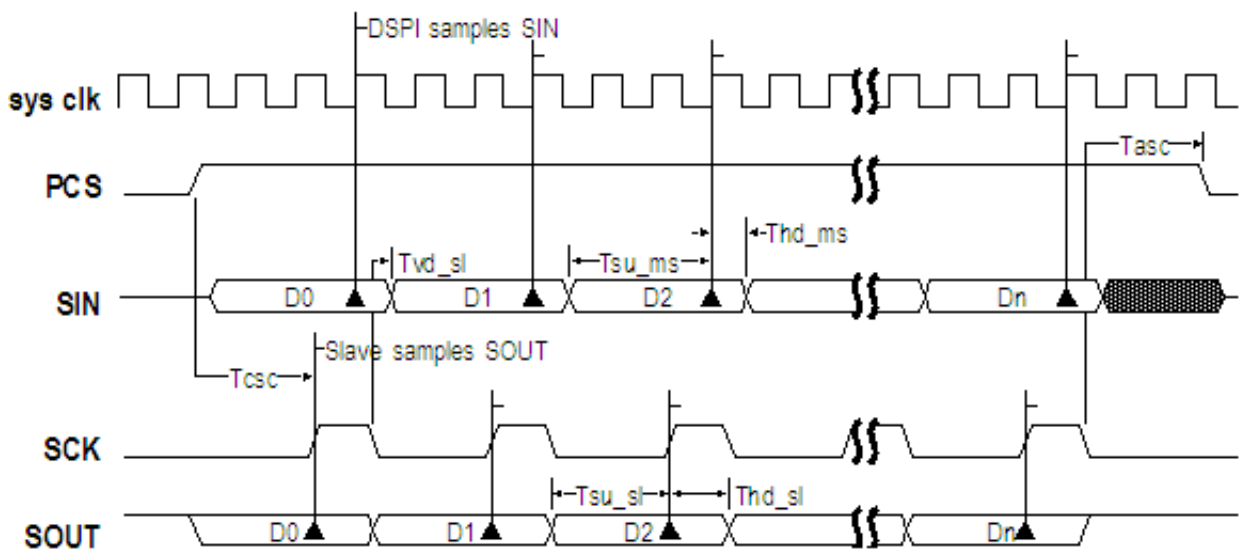


Figure 48-10. DSPI Modified Transfer Format (MTFE=1, CPHA=0, $f_{sck} = f_{sys}/3$)

48.5.4.4 Modified SPI Transfer Format (MTFE = 1, CPHA = 1)

The following figures show the Modified Transfer Format for CPHA = 1. Only the condition, where CPOL = 0 is shown. At the start of a transfer the DSPI asserts the PCS signal to the slave device. After the PCS to SCK delay has elapsed the master and the slave put data on their SOUT pins at the first edge of SCK . The slave samples the master SOUT signal on the even numbered edges of SCK. The master samples the slave SOUT

signal on the odd numbered SCK edges starting with the third SCK edge. The slave samples the last bit on the last edge of the SCK. The master samples the last slave SOUT bit one half SCK cycle after the last edge of SCK. No clock edge will be visible on the master SCK pin during the sampling of the last bit. **The SCK to PCS delay and the After SCK delay must be greater or equal to half of the SCK period.**

NOTE

When using MTFE=1 and CPHA=1 in master mode and receiving back to back frames by configuring MCR [CONT_SCKE] =1 and PUSHR [CONT] =1, configure the frame size of the first frame as less than or equal to the frame size of the next frame for correct operation. Also ensure that for all received frames, read bits equal to their respective frame sizes and mask any extra bits during POP operation.

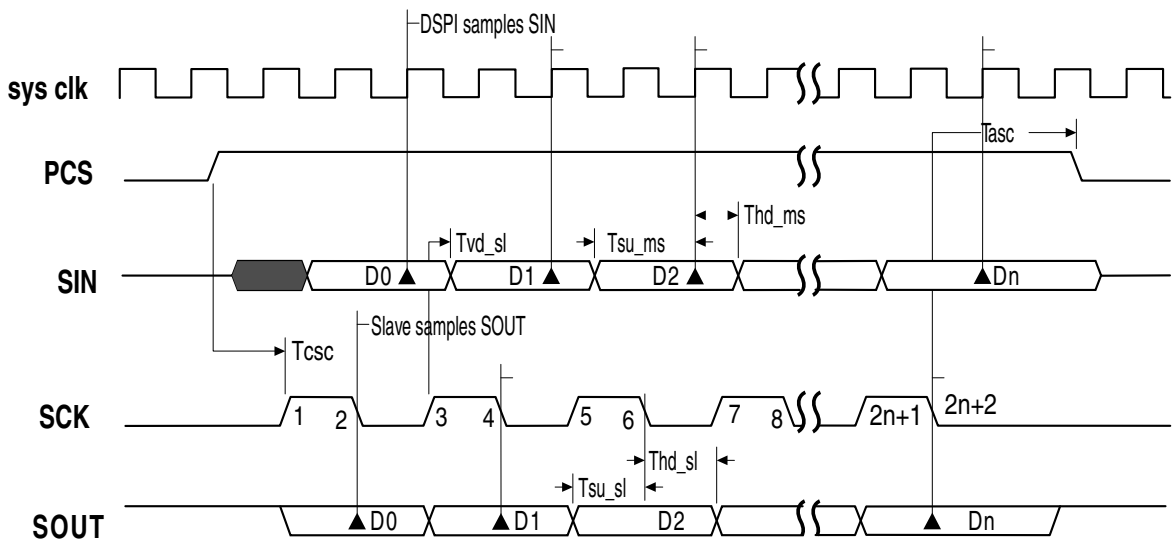


Figure 48-11. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/2$)

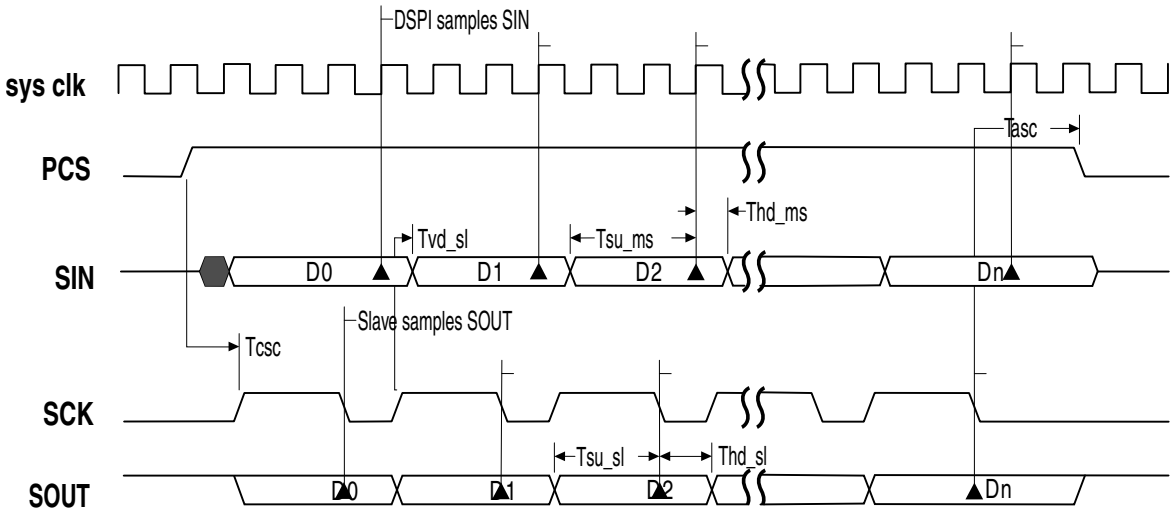


Figure 48-12. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/3$)

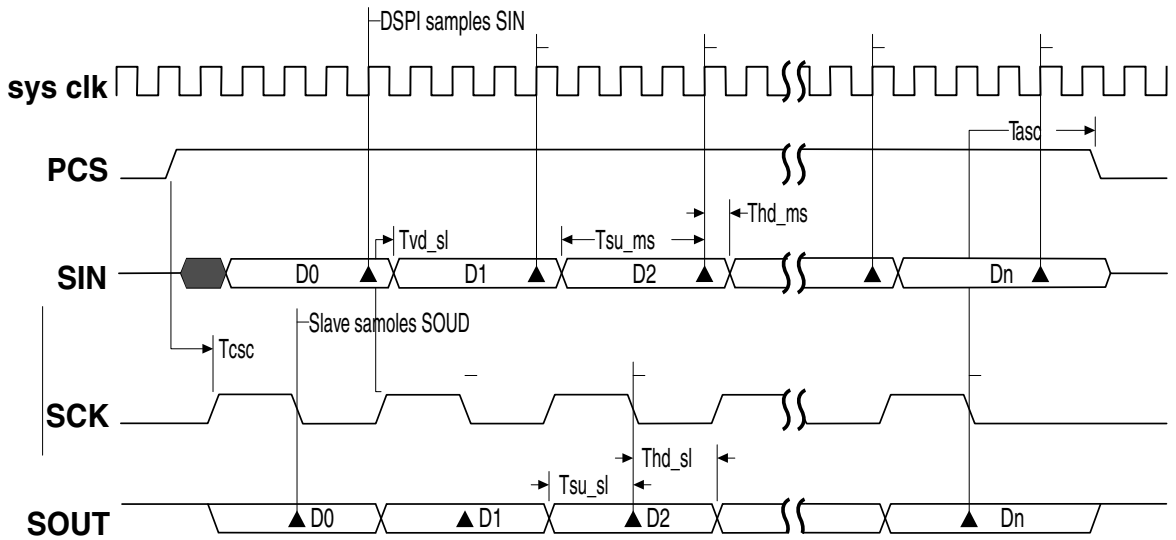


Figure 48-13. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/4$)

48.5.4.5 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.

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.

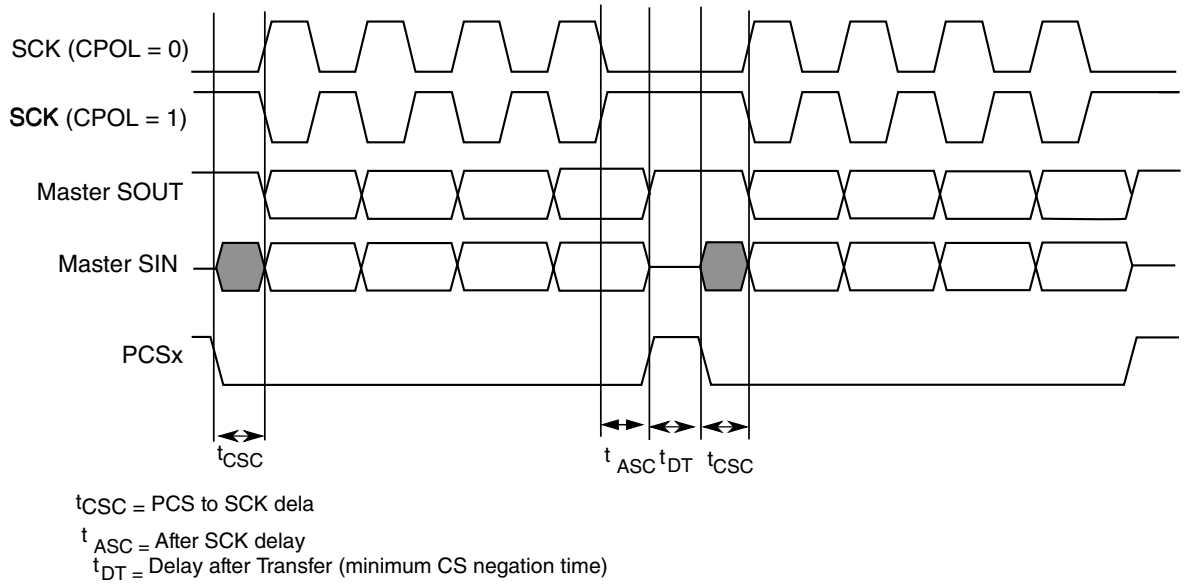


Figure 48-14. Example of non-continuous format (CPHA=1, CONT=0)

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.

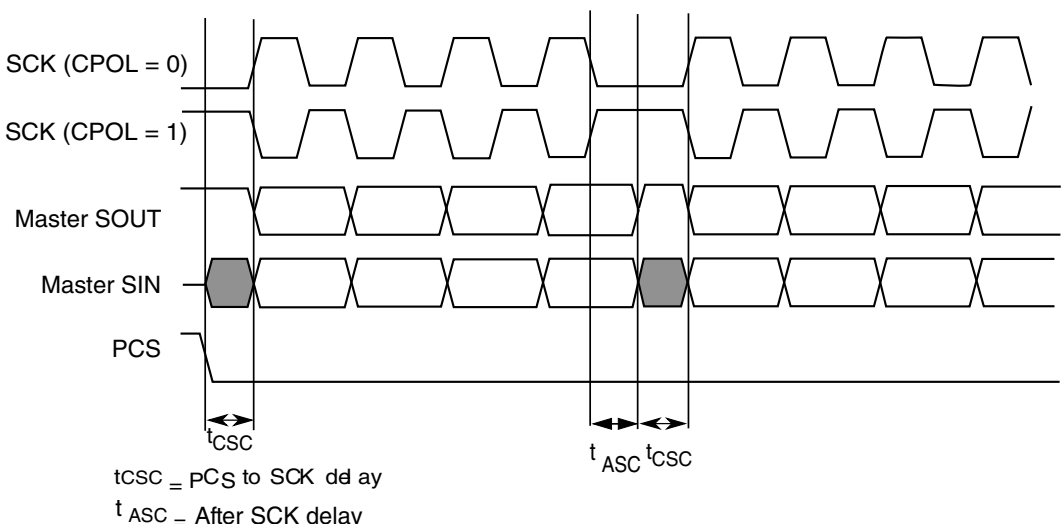


Figure 48-15. Example of continuous transfer (CPHA=1, CONT=1)

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.
- PUSHR[CONT] must be deasserted before asserting MCR[HALT] in master mode. This will make sure that the PCSn signals are deasserted. Asserting MCR[HALT] 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.

48.5.5 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 only if MCR[HALT] bit is low. 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.

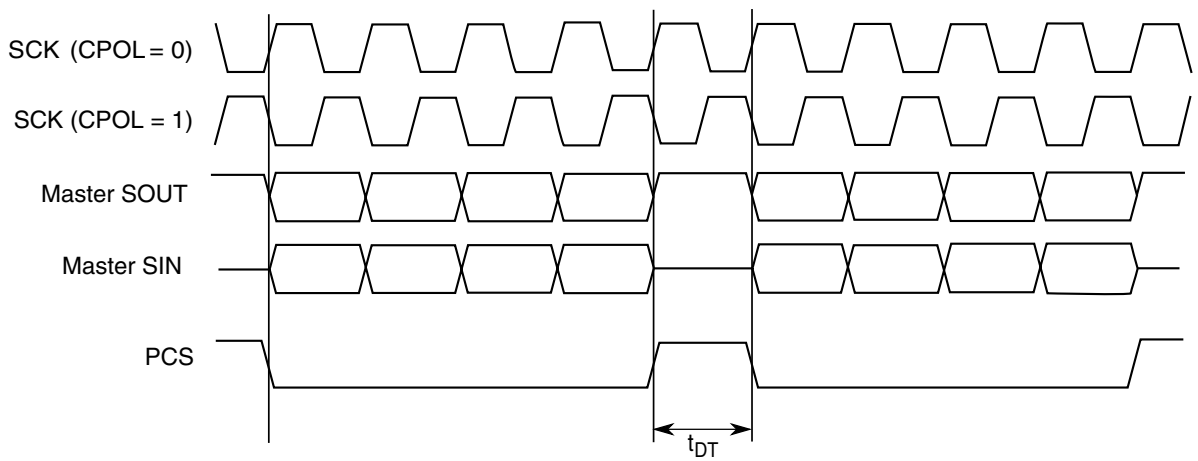


Figure 48-16. Continuous SCK Timing Diagram (CONT=0)

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.

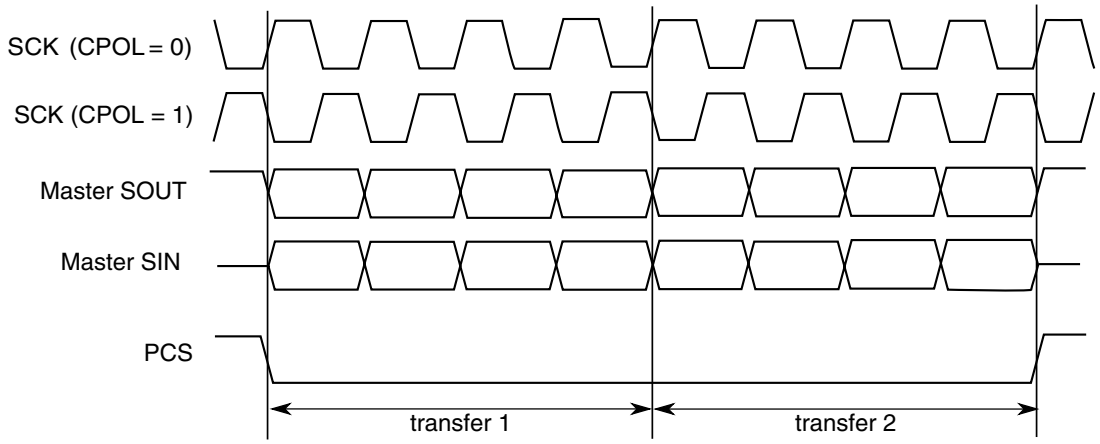


Figure 48-17. Continuous SCK timing diagram (CONT=1)

48.5.6 Slave Mode Operation Constraints

Slave mode logic shift register is buffered. This allows data streaming operation, when the module 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.

48.5.7 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 48-14. 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	-

Table continues on the next page...

Table 48-14. Interrupt and DMA request conditions (continued)

Condition	Flag	Interrupt	DMA
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.

48.5.7.1 End Of Queue interrupt request

The End Of Queue (EOQ) interrupt request indicates that the end of a transmit queue is reached. The module generates the interrupt request when EOQ interrupt requests are enabled (RSER[EOQF_RE]) and the EOQ bit in the executing SPI command is 1.

The module generates the interrupt request when the last bit of the SPI frame with EOQ bit set is transmitted.

48.5.7.2 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.

48.5.7.3 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.

48.5.7.4 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 the module 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.

48.5.7.5 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.

48.5.7.6 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.

48.5.8 Power saving features

The module supports following power-saving strategies:

- External Stop mode
- Module Disable mode – Clock gating of non-memory mapped logic

48.5.8.1 Stop mode (External Stop mode)

This module supports the Stop mode protocol. When a request is made to enter External Stop mode, the module acknowledges the request . If a serial transfer is in progress, then this module waits until it reaches the frame boundary before it is ready to have its clocks shut off . While the clocks are shut off, this module's memory-mapped logic is not accessible. This also puts the module 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.

48.5.8.2 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.

When the MDIS bit is set, the module negates the 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.

48.6 Initialization/application information

This section describes how to initialize the module.

48.6.1 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.

48.6.2 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].

48.6.3 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.

48.6.4 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 protocol 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 48-15. 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

Table continues on the next page...

Table 48-15. Baud rate values (bps) (continued)

		Baud rate divider prescaler values			
		2	3	5	7
	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

48.6.5 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 protocol 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 48-16. 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

48.6.6 Calculation of FIFO pointer addresses

Complete visibility of the 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. 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.

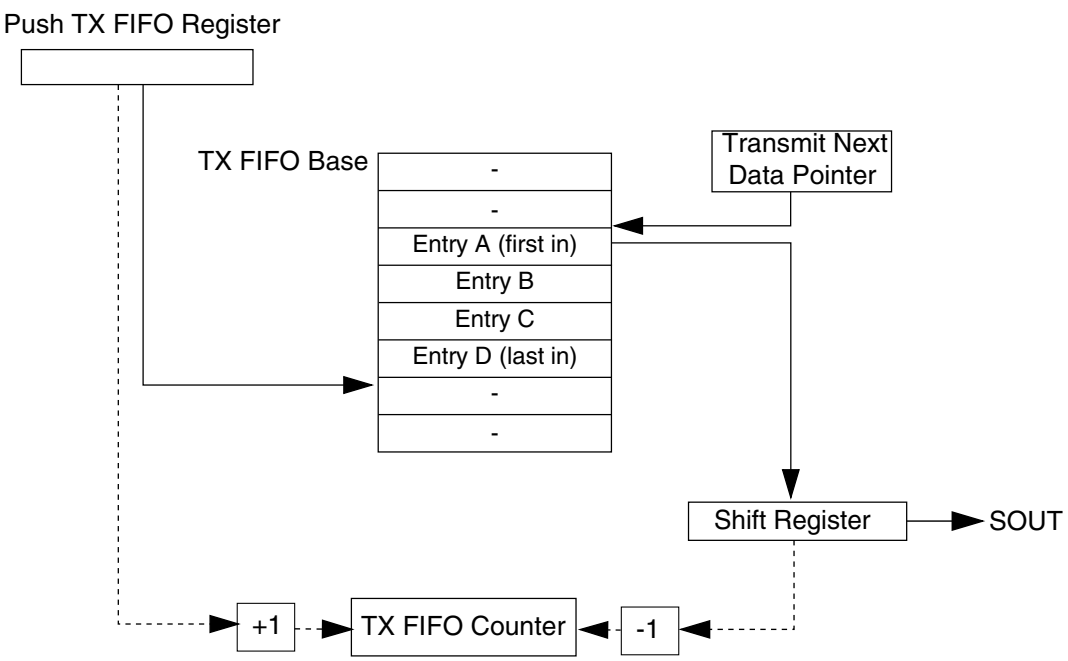


Figure 48-18. TX FIFO pointers and counter

48.6.6.1 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:

$$\text{First-in EntryAddress} = \text{TXFIFOBase} + \left(4 \times \text{TXNXTPTR}\right)$$

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

$$\text{Last-in Entry address} = \text{TXFIFO Base} + 4 \times (\text{TXCTR} + \text{TXNXTPTR} - 1) \bmod (\text{TXFIFO depth})$$

TX FIFO Base - Base address of TX FIFO
TXCTR - TX FIFO Counter
TXNXTPTR - Transmit Next Pointer
TX FIFO Depth - Transmit FIFO depth, implementation specific

48.6.6.2 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:

$$\text{First-in Entry Address} = \text{RX FIFO Base} + (4 \times \text{POPNXTPTR})$$

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

$$\text{Last-in Entry address} = \text{RX FIFO Base} + 4 \times (\text{RXCTR} + \text{POPNXTPTR} - 1) \bmod (\text{RXFIFO depth})$$

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 49
Inter-Integrated Circuit (I2C)

49.1 Chip-specific I2C information

49.1.1 I2C Instantiation

This device has 2 I^2C module(s).

The I2C module includes SMBus support and DMA support. It also has optional addressmatch wakeup in Stop/VLPS mode. The digital glitch filter implemented in the IICmodule, controlled by the I2Cx_FLT[FLT] registers, is clocked from the bus clock andthus has filter granularity in bus clock cycle counts.

49.2 Introduction

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 at least 400 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*.

49.2.1 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 input glitch filter
- Low power mode wakeup on slave address match
- Range slave address support
- DMA support
- Double buffering support to achieve higher baud rate

49.2.2 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.

49.2.3 Block diagram

The following figure is a functional block diagram of the I2C module.

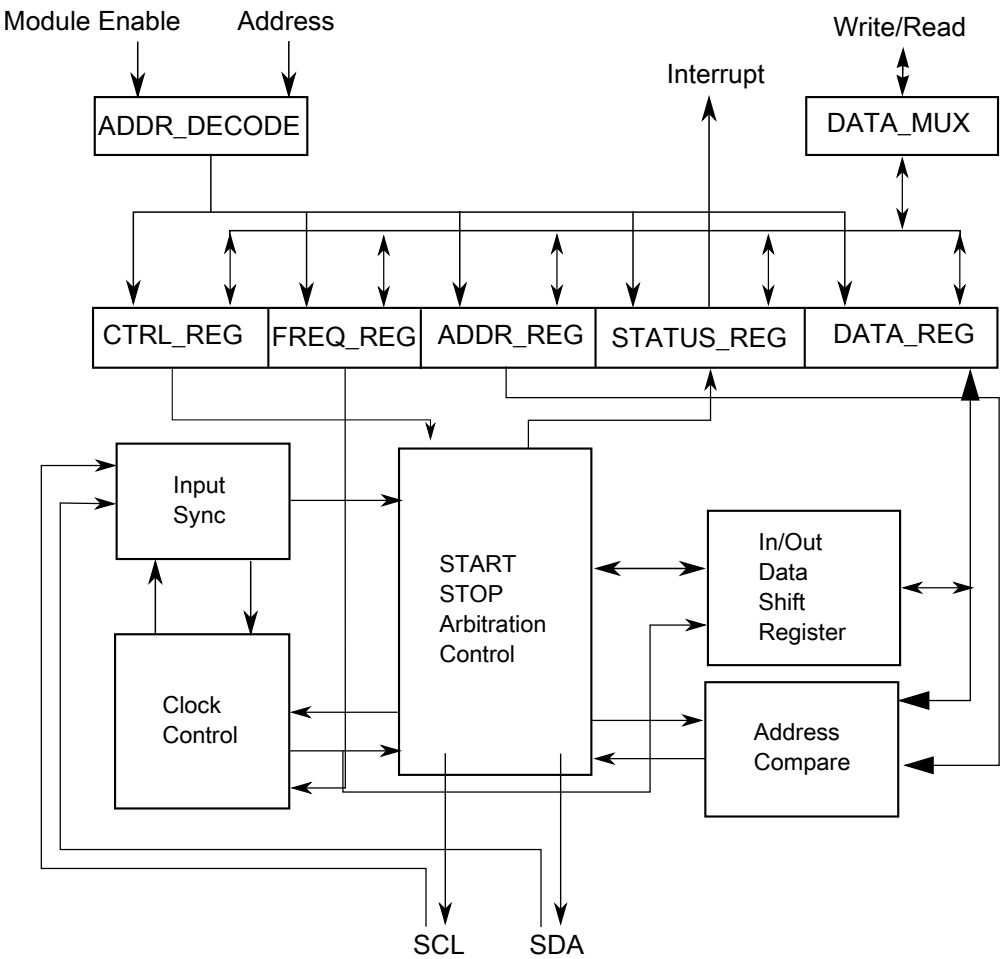


Figure 49-1. I2C Functional block diagram

49.3 I²C signal descriptions

The signal properties of I²C are shown in the table found here.

Table 49-1. 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

49.4 Memory map/register definition

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	49.4.1/1441
4006_6001	I2C Frequency Divider register (I2C0_F)	8	R/W	00h	49.4.2/1441
4006_6002	I2C Control Register 1 (I2C0_C1)	8	R/W	00h	49.4.3/1442
4006_6003	I2C Status register (I2C0_S)	8	R/W	80h	49.4.4/1444
4006_6004	I2C Data I/O register (I2C0_D)	8	R/W	00h	49.4.5/1446
4006_6005	I2C Control Register 2 (I2C0_C2)	8	R/W	00h	49.4.6/1446
4006_6006	I2C Programmable Input Glitch Filter Register (I2C0_FLT)	8	R/W	00h	49.4.7/1447
4006_6007	I2C Range Address register (I2C0_RA)	8	R/W	00h	49.4.8/1449
4006_6008	I2C SMBus Control and Status register (I2C0_SMB)	8	R/W	00h	49.4.9/1449
4006_6009	I2C Address Register 2 (I2C0_A2)	8	R/W	C2h	49.4.10/ 1451
4006_600A	I2C SCL Low Timeout Register High (I2C0_SLTH)	8	R/W	00h	49.4.11/ 1451
4006_600B	I2C SCL Low Timeout Register Low (I2C0_SLTL)	8	R/W	00h	49.4.12/ 1452
4006_600C	I2C Status register 2 (I2C0_S2)	8	R/W	01h	49.4.13/ 1452
4006_7000	I2C Address Register 1 (I2C1_A1)	8	R/W	00h	49.4.1/1441
4006_7001	I2C Frequency Divider register (I2C1_F)	8	R/W	00h	49.4.2/1441
4006_7002	I2C Control Register 1 (I2C1_C1)	8	R/W	00h	49.4.3/1442
4006_7003	I2C Status register (I2C1_S)	8	R/W	80h	49.4.4/1444
4006_7004	I2C Data I/O register (I2C1_D)	8	R/W	00h	49.4.5/1446
4006_7005	I2C Control Register 2 (I2C1_C2)	8	R/W	00h	49.4.6/1446
4006_7006	I2C Programmable Input Glitch Filter Register (I2C1_FLT)	8	R/W	00h	49.4.7/1447
4006_7007	I2C Range Address register (I2C1_RA)	8	R/W	00h	49.4.8/1449
4006_7008	I2C SMBus Control and Status register (I2C1_SMB)	8	R/W	00h	49.4.9/1449
4006_7009	I2C Address Register 2 (I2C1_A2)	8	R/W	C2h	49.4.10/ 1451
4006_700A	I2C SCL Low Timeout Register High (I2C1_SLTH)	8	R/W	00h	49.4.11/ 1451
4006_700B	I2C SCL Low Timeout Register Low (I2C1_SLTL)	8	R/W	00h	49.4.12/ 1452
4006_700C	I2C Status register 2 (I2C1_S2)	8	R/W	01h	49.4.13/ 1452

49.4.1 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 Write	AD[7:1]							0
Reset	0	0	0	0	0	0	0	0

I2Cx_A1 field descriptions

Field	Description
7–1 AD[7:1]	Address 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

49.4.2 I2C Frequency Divider register (I2Cx_F)

Address: Base address + 1h offset

Bit	7	6	5	4	3	2	1	0
Read Write	MULT		ICR					
Reset	0	0	0	0	0	0	0	0

I2Cx_F field descriptions

Field	Description
7–6 MULT	Multiplier Factor Defines the multiplier factor (mul). This factor is used along with the SCL divider to generate the I2C baud rate. 00 mul = 1 01 mul = 2 10 mul = 4 11 Reserved
ICR	ClockRate Prescales the I2C module 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. The SCL divider multiplied by multiplier factor (mul) determines the I2C baud rate. `I2C baud rate = I2C module clock speed (Hz)/(mul × SCL divider)`

Table continues on the next page...

I2Cx_F field descriptions (continued)

Field	Description

	The SDA hold time is the delay from the falling edge of SCL (I2C clock) to the changing of SDA (I2C data).
	
	`SDA hold time = I2C module clock period (s) × mul × SDA hold value`
	
	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).
	
	`SCL start hold time = I2C module clock period (s) × mul × SCL start hold value`
	
	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).
	
	`SCL stop hold time = I2C module clock period (s) × mul × SCL stop hold value`
	
	For example, if the I2C module clock 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

49.4.3 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 IICEN	I2C Enable Enables I2C module operation. 0 Disabled 1 Enabled
6 IICIE	I2C Interrupt Enable Enables I2C interrupt requests.

Table continues on the next page...

I2Cx_C1 field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
5 MST	Master Mode Select When MST is changed from 0 to 1, a START signal is generated on the bus and master mode is selected. When this bit changes from 1 to 0, a STOP signal is generated and the mode of operation changes from master to slave. 0 Slave mode 1 Master mode
4 TX	Transmit Mode Select 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. 0 Receive 1 Transmit
3 TXAK	Transmit Acknowledge Enable Specifies the value driven onto the SDA during data acknowledge cycles for both master and slave receivers. The value of SMB[FACK] affects NACK/ACK generation. **NOTE:** SCL is held low until TXAK is written. 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). 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 RSTA	Repeat START Writing 1 to this bit generates a repeated START condition provided it is the current master. This bit will always be read as 0. Attempting a repeat at the wrong time results in loss of arbitration.
1 WUEN	Wakeup Enable The I2C module can wake the MCU from low power mode with no peripheral bus running when slave address matching occurs. 0 Normal operation. No interrupt generated when address matching in low power mode. 1 Enables the wakeup function in low power mode.
0 DMAEN	DMA Enable Enables or disables the DMA function. 0 All DMA signalling disabled. 1 DMA transfer is enabled. While SMB[FACK] = 0, the following conditions trigger the DMA request: • a data byte is received, and either address or data is transmitted. (ACK/NACK is automatic) • the first byte received matches the A1 register or is a general call address.

Table continues on the next page...

I2Cx_C1 field descriptions (continued)

Field	Description
	If any address matching occurs, S[IAAS] and S[TCF] are set. If the direction of transfer is known from master to slave, then it is not required to check S[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. When FACK = 1, an address or a data byte is transmitted.

49.4.4 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 TCF	Transfer Complete Flag Acknowledges a byte transfer; TCF is set on the completion of a byte transfer. This bit is valid only during or immediately following a transfer to or from the I2C module. TCF is cleared by reading the I2C data register in receive mode or by writing to the I2C data register in transmit mode. **NOTE:** In the buffer mode, TCF is cleared automatically by internal reading or writing the data register I2C_D, with no need waiting for manually reading/writing the I2C data register in Rx/Tx mode. 0 Transfer in progress 1 Transfer complete
6 IAAS	Addressed As A Slave This bit is set by one of the following conditions: • The calling address matches the programmed primary slave address in the A1 register, or matches the range address in the RA register (which must be set to a nonzero value and under the condition I2C_C2[RMEN] = 1). • C2[GCAEN] is set and a general call is received. • SMB[SIICAEN] is set and the calling address matches the second programmed slave address. • ALERTEN is set and an SMBus alert response address is received • RMEN is set and an address is received that is within the range between the values of the A1 and RA registers. IAAS 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. 0 Not addressed 1 Addressed as a slave
5 BUSY	Bus Busy

Table continues on the next page...

I2C*x*_S field descriptions (continued)

Field	Description
	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. 0 Bus is idle 1 Bus is busy
4 ARBL	Arbitration Lost This bit is set by hardware when the arbitration procedure is lost. The ARBL bit must be cleared by software, by writing 1 to it. 0 Standard bus operation. 1 Loss of arbitration.
3 RAM	Range Address Match This bit is set to 1 by any of the following conditions, if I2C_C2[RMEN] = 1: • Any nonzero calling address is received that matches the address in the RA register. • The calling address is within the range of values of the A1 and RA registers. **NOTE:** For the RAM bit to be set to 1 correctly, C1[IICIE] must be set to 1. Writing the C1 register with any value clears this bit to 0. 0 Not addressed 1 Addressed as a slave
2 SRW	Slave Read/Write When addressed as a slave, SRW indicates the value of the R/$\overline{W}$ command bit of the calling address sent to the master. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave
1 IICIF	Interrupt Flag This bit sets when an interrupt is pending. This bit must be cleared by software by writing 1 to it, such as in the interrupt routine. One of the following events can set this bit: • 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. • One byte transfer, excluding ACK/NACK bit, completes if FACK is 1. • Match of slave address to calling address including primary slave address, range slave address, alert response address, second slave address, or general call address. • Arbitration lost • In SMBus mode, any timeouts except SCL and SDA high timeouts • I2C bus stop or start detection if the SSIE bit in the Input Glitch Filter register is 1 **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. 0 No interrupt pending 1 Interrupt pending
0 RXAK	Receive Acknowledge

Table continues on the next page...

I2Cx_S field descriptions (continued)

Field	Description
	0 Acknowledge signal was received after the completion of one byte of data transmission on the bus 1 No acknowledge signal detected

49.4.5 I2C Data I/O register (I2Cx_D)

Address: Base address + 4h offset

Bit	7	6	5	4	3	2	1	0
Read Write	DATA							
Reset	0	0	0	0	0	0	0	0

I2Cx_D field descriptions

Field	Description
DATA	Data 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. **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. In slave mode, the same functions are available after an address match occurs. 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. 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. 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).

49.4.6 I2C Control Register 2 (I2Cx_C2)

Address: Base address + 5h offset

Bit	7	6	5	4	3	2	1	0
Read 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 GCAEN	General Call Address Enable Enables general call address. 0 Disabled 1 Enabled
6 ADEXT	Address Extension Controls the number of bits used for the slave address. 0 7-bit address scheme 1 10-bit address scheme
5 HDRS	High Drive Select Controls the drive capability of the I2C pads. 0 Normal drive mode 1 High drive mode
4 SBRC	Slave Baud Rate Control 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. 0 The slave baud rate follows the master baud rate and clock stretching may occur 1 Slave baud rate is independent of the master baud rate
3 RMEN	Range Address Matching Enable This bit controls the slave address matching for addresses between the values of the A1 and RA registers. When this bit is set, a slave address matching occurs for any address greater than the value of the A1 register and less than or equal to the value of the RA register. 0 Range mode disabled. No address matching occurs for an address within the range of values of the A1 and RA registers. 1 Range mode enabled. Address matching occurs when a slave receives an address within the range of values of the A1 and RA registers.
AD[10:8]	Slave Address 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.

49.4.7 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 SHEN	Stop Hold Enable Set this bit to hold off entry to stop mode when any data transmission or reception is occurring. The following scenario explains the holdoff functionality: 1. The I2C module is configured for a basic transfer, and the SHEN bit is set to 1. 2. A transfer begins. 3. The MCU signals the I2C module to enter stop mode. 4. The byte currently being transferred, including both address and data, completes its transfer. 5. The I2C slave or master acknowledges that the in-transfer byte completed its transfer and acknowledges the request to enter stop mode. 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. 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. 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. 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. 0 Stop holdoff is disabled. The MCU's entry to stop mode is not gated. 1 Stop holdoff is enabled.
6 STOPF	I2C Bus Stop Detect Flag Hardware sets this bit when the I2C bus's stop status is detected. The STOPF bit must be cleared by writing 1 to it. **NOTE:** The stop flag is only for the matched slave devices, therefore the master will not respond for it. 0 No stop happens on I2C bus 1 Stop detected on I2C bus
5 SSIE	I2C Bus Stop or Start Interrupt Enable This bit enables the interrupt for I2C bus stop or start detection. **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. 0 Stop or start detection interrupt is disabled 1 Stop or start detection interrupt is enabled
4 STARTF	I2C Bus Start Detect Flag Hardware sets this bit when the I2C bus's start status is detected. The STARTF bit must be cleared by writing 1 to it. 0 No start happens on I2C bus 1 Start detected on I2C bus
FLT	I2C Programmable Filter Factor

Table continues on the next page...

I2Cx_FLT field descriptions (continued)

Field	Description
	Controls the width of the glitch, in terms of I2C module 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. 0h No filter/bypass 1-Fh Filter glitches up to width of *n* I2C module clock cycles, where *n*=1-15d

49.4.8 I2C Range Address register (I2Cx_RA)

Address: Base address + 7h offset

Bit	7	6	5	4	3	2	1	0
Read	RAD							0
Write								
Reset	0	0	0	0	0	0	0	0

I2Cx_RA field descriptions

Field	Description
7–1 RAD	Range Slave Address This field contains the slave address to be used by the I2C module. The field is used in the 7-bit address scheme. If I2C_C2[RMEN] is set to 1, any nonzero value write enables this register. This register value can be considered as a maximum boundary in the range matching mode.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

49.4.9 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 is set to 1 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 FACK	Fast NACK/ACK Enable For SMBus packet error checking, the CPU must be able to issue an ACK or NACK according to the result of receiving data byte. 0 An ACK or NACK is sent on the following receiving data byte 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. **NOTE:** Enable I2C_S2[DFEN] in the master receive mode.
6 ALERTEN	SMBus Alert Response Address Enable Enables or disables SMBus alert response address matching. **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. 0 SMBus alert response address matching is disabled 1 SMBus alert response address matching is enabled
5 SIICAEN	Second I2C Address Enable Enables or disables SMBus device default address. 0 I2C address register 2 matching is disabled 1 I2C address register 2 matching is enabled
4 TCKSEL	Timeout Counter Clock Select Selects the clock source of the timeout counter. 0 Timeout counter counts at the frequency of the I2C module clock / 64 1 Timeout counter counts at the frequency of the I2C module clock
3 SLTF	SCL Low Timeout Flag 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. **NOTE:** The low timeout function is disabled when the SLT register's value is 0. 0 No low timeout occurs 1 Low timeout occurs
2 SHTF1	SCL High Timeout Flag 1 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.

Table continues on the next page...

I2Cx_SMB field descriptions (continued)

Field	Description
	0 No SCL high and SDA high timeout occurs 1 SCL high and SDA high timeout occurs
1 SHTF2	SCL High Timeout Flag 2 This bit sets when SCL is held high and SDA is held low more than clock × LoValue / 512. Software clears this bit by writing 1 to it. 0 No SCL high and SDA low timeout occurs 1 SCL high and SDA low timeout occurs
0 SHTF2IE	SHTF2 Interrupt Enable Enables SCL high and SDA low timeout interrupt. 0 SHTF2 interrupt is disabled 1 SHTF2 interrupt is enabled

49.4.10 I2C Address Register 2 (I2Cx_A2)

Address: Base address + 9h offset

Bit	7	6	5	4	3	2	1	0
Read	SAD							0
Write								
Reset	1	1	0	0	0	0	1	0

I2Cx_A2 field descriptions

Field	Description
7–1 SAD	SMBus Address Contains the slave address used by the SMBus. This field is used on the device default address or other related addresses.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

49.4.11 I2C SCL Low Timeout Register High (I2Cx_SLTH)

Address: Base address + Ah offset

Bit	7	6	5	4	3	2	1	0
Read Write	SSLT[15:8]							
Reset	0	0	0	0	0	0	0	0

I2Cx_SLTH field descriptions

Field	Description
SSLT[15:8]	SSLT[15:8] Most significant byte of SCL low timeout value that determines the timeout period of SCL low.

49.4.12 I2C SCL Low Timeout Register Low (I2Cx_SLTL)

Address: Base address + Bh offset

Bit	7	6	5	4	3	2	1	0
Read Write	SSLT[7:0]							
Reset	0	0	0	0	0	0	0	0

I2Cx_SLTL field descriptions

Field	Description
SSLT[7:0]	SSLT[7:0] Least significant byte of SCL low timeout value that determines the timeout period of SCL low.

49.4.13 I2C Status register 2 (I2Cx_S2)

Address: Base address + Ch offset

Bit	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	DFEN	ERROR	EMPTY
Write							w1c	
Reset	0	0	0	0	0	0	0	1

I2Cx_S2 field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 DFEN	Double Buffer Enable Enables or disables the double buffer mode. In the double buffer mode, the clock stretch is disabled.

Table continues on the next page...

I2Cx_S2 field descriptions (continued)

Field	Description
	0 Disables the double buffer mode; clock stretch is enabled. 1 Enables the double buffer mode; clock stretch is disabled. In the slave mode, the I2C will not hold bus between data transfers.
1 ERROR	Error flag Indicates if there are read or write errors with the Tx and Rx buffers. 0 The buffer is not full and all write/read operations have no errors. 1 There are 3 or more write/read errors during the data transfer phase (when the Empty flag is not set and the buffer is busy).
0 EMPTY	Empty flag Indicates if the Tx or Rx buffer is empty. 0 Tx or Rx buffer is not empty and cannot be written to, that is new data cannot be loaded into the buffer. 1 Tx or Rx buffer is empty and can be written to, that is new data can be loaded into the buffer.

49.5 Functional description

This section provides a comprehensive functional description of the I2C module.

49.5.1 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.

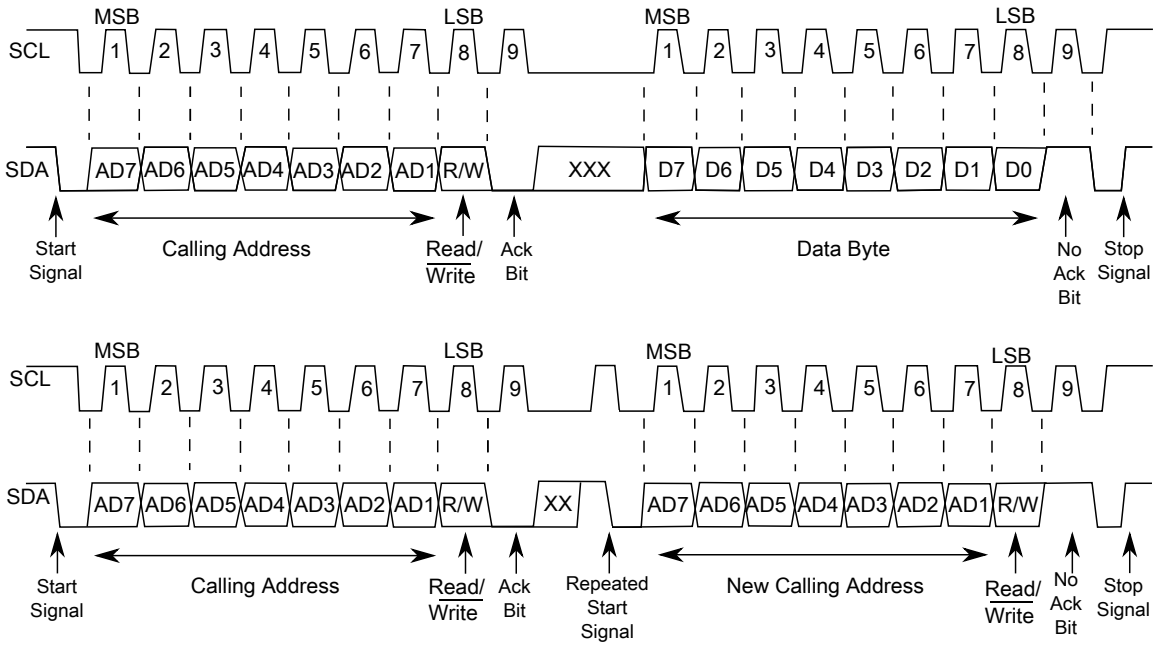


Figure 49-2. I2C bus transmission signals

49.5.1.1 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.

49.5.1.2 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.

49.5.1.3 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.

49.5.1.4 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.

49.5.1.5 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.The master needs to send a NACK signal before sending repeated-START in the buffering mode.

49.5.1.6 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.

49.5.1.7 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.

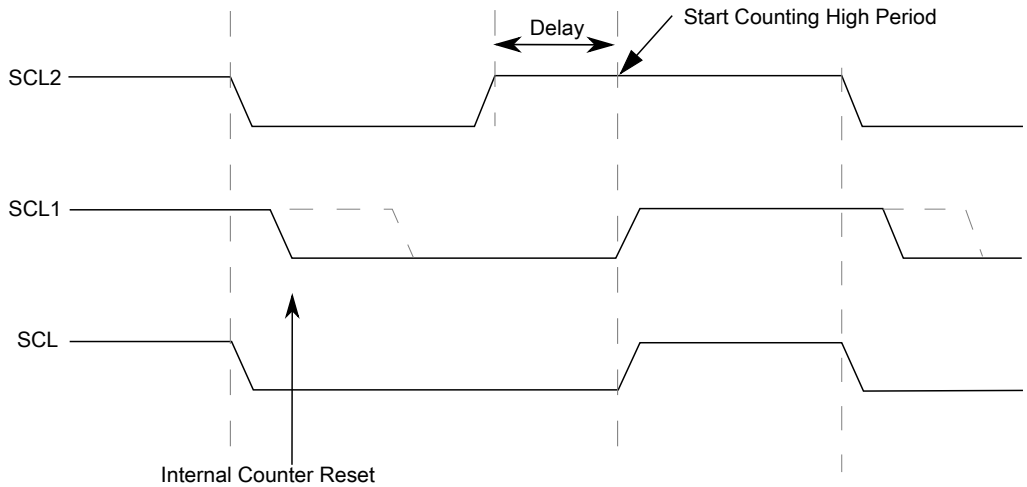


Figure 49-3. I2C clock synchronization

49.5.1.8 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.

49.5.1.9 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.

49.5.1.10 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 49-2. 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

49.5.2 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.

49.5.2.1 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 49-3. 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.

49.5.2.2 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 49-4. 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.

49.5.3 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 RMEN bit is set, when the Range Address register is programmed to a nonzero value, any address within the range of values of Address Register 1 (excluded) and the Range Address register (included) 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.

49.5.4 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.

49.5.4.1 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.

49.5.4.1.1 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.

49.5.4.1.2 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.

49.5.4.1.3 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.

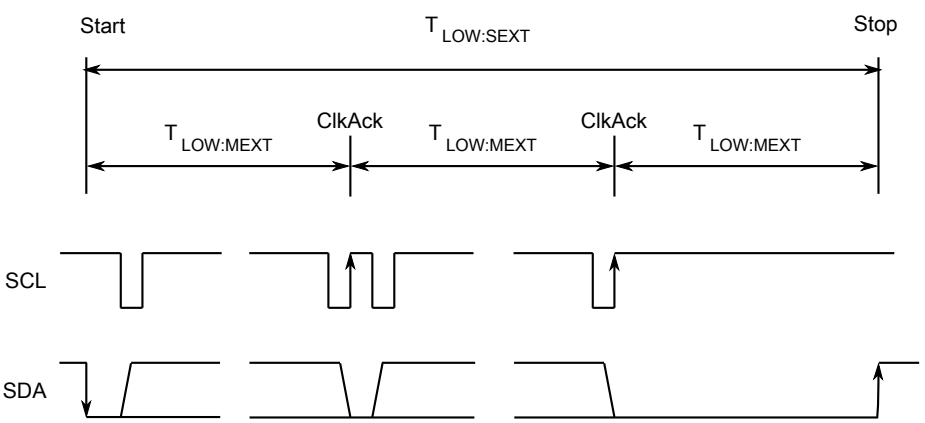


Figure 49-4. Timeout measurement intervals

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.

49.5.4.2 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.

49.5.5 Resets

The I2C module is disabled after a reset. The I2C module cannot cause a core reset.

49.5.6 Interrupts

The I2C module generates an interrupt when any of the events in the table found here 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 49-5. 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

49.5.6.1 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.

49.5.6.2 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.

49.5.6.3 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 SSIE bits are both set to 1.

49.5.6.4 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.

49.5.6.5 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.

49.5.6.6 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.

49.5.7 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) I2C module 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 I2C module clock cycles) for the filter to absorb and not pass.

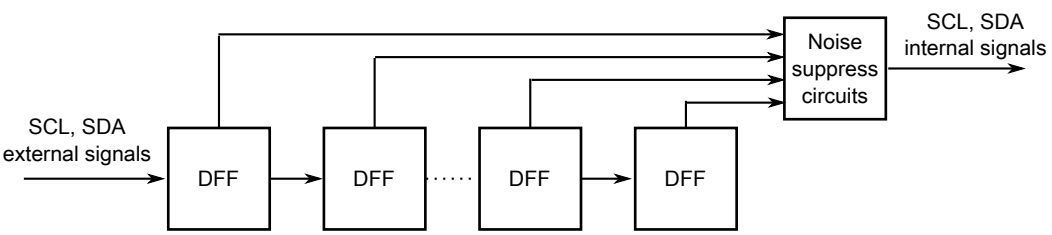


Figure 49-5. Programmable input glitch filter diagram

49.5.8 Address matching wake-up

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.

Data sent on the bus that is the same as a target device address might also wake the target MCU.

After the address matching IAAS bit is set, an interrupt is sent at the end of address matching to wake the core. The IAAS bit must be cleared after the clock recovery.

NOTE

After the system recovers and is in Run mode, restart the I2C module if it is needed to transfer packets. To avoid I2C transfer problems resulting from the situation, firmware should prevent the MCU execution of a STOP instruction when the I2C module is in the middle of a transfer unless the Stop mode holdoff feature is used during this period (set FLT[SHEN] to 1).

NOTE

After I2C address matching wake-up, the master must wait a time long enough for the slave ISR to finish running and resend start or repeat start signals.

For the SRW bit to function properly, it only supports Address +Write to wake up by I2C address matching. Before entering the next low power mode, Address+Write must be sent to change the SRW status.

49.5.9 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, only the TCF initiates a DMA request. All other events generate CPU interrupts.

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.

49.5.10 Double buffering mode

In the double buffering mode, the data transfer is processed byte by byte. However, the data can be transferred without waiting for the interrupt or the polling to finish. This means the write/read I2C_D operation will not block the data transfer, as the hardware has already finished the internal write or read. The benefit is that the baud rate is able to achieve higher speed.

There are several items to consider as follows:

- When initiating a double buffering transfer at Tx side, the user can write 2 values to the I2C_D buffer before transfer. However, that is allowed only at one time per package frame (due to the buffer depth, and because two-times writes in each ISR are not allowed). The second write to the I2C_D buffer must wait for the Empty flag. On the other hand, at Rx side the user can read twice in a one-byte transfer (if needed).

NOTE

Check Empty flag before write to I2C_D.

Write twice to the I2C_D buffer ONLY after the address matching byte. Do not write twice (Address+Data) before START or at the beginning of I2C transfer, especially when the baud rate is very slow.

- To write twice in one frame, during the next-to-last ISR, do a dummy read from the I2C_D buffer at Tx side (or the TCF will stay high, because the TCF is cleared by write/read operation). In the next-to-last ISR, do not send data again (the buffer data will be under running).
- To keep new ISRs software-compatible with previous ISRs, the write/read I2C_D operation will not block the internal-hardware-released SCL/SDA signals. At the ACK phase, the bus is released to accept the next byte if the master can send the clock immediately.
- On the slave side, two-times writes to the I2C_D buffer may be limited by the master's clock and START/repeated-START signal. This is not currently supported, and the master's START/repeated-START signal will break data transfers. To release the bus, do a dummy read or write to the I2C_D buffer again. It is suggested to send repeated-START/START during intervals as before.
- The master receive should send a NACK in the next-to-last ISR, if it wants to do the STOP or the repeated-START work. The transmitting slave which receives the NACK, will switch to receive mode, and do a dummy read to release SCL and SDA signals.

49.6 Initialization/application information

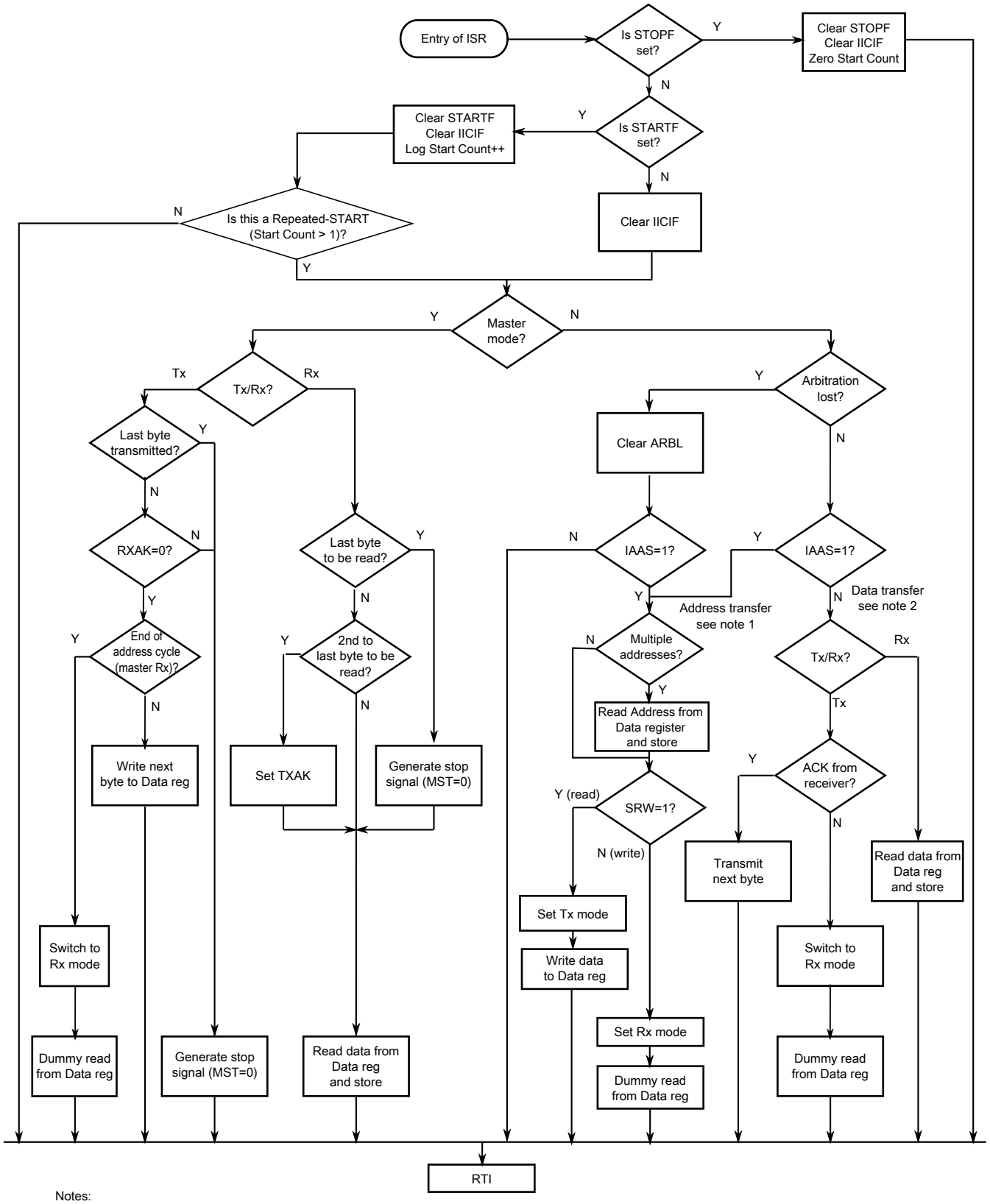
Module Initialization (Slave)

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)

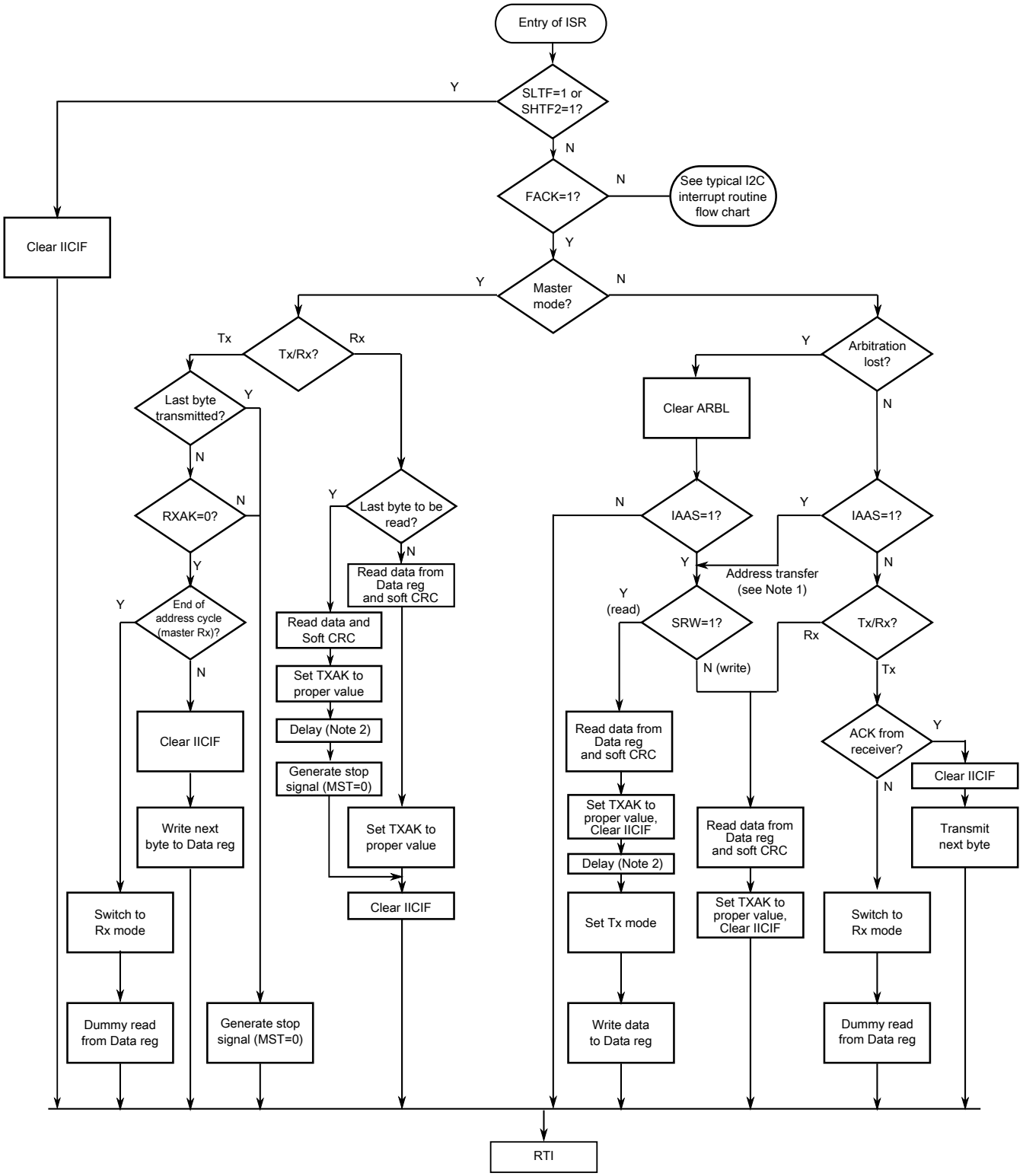
1. Write: Frequency Divider register to set the I2C baud rate (see example in description of ICR)
2. Write: Control Register 1 to enable the I2C module and interrupts
3. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
4. Initialize RAM variables used to achieve the routine shown 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. An example of an I2C driver which implements many of the steps described here is available in AN4342: Using the Inter-Integrated Circuit on ColdFire+ and Kinetis .



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 49-6. 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 stop signal generation, to wait for the possible longest time period (in worst case) of the 9th SCL cycle.

Figure 49-7. Typical I2C SMBus interrupt routine

Chapter 50
Low Power Universal asynchronous receiver/ transmitter (LPUART)

50.1 Chip-specific LPUART information

50.1.1 LPUART overview

The LPUART module supports basic UART with DMA interface function and x4 to x32 oversampling of baud-rate.

The module can remain functional in Stop and VLPS mode provided the clock it is using remains enabled.

This module supports LIN slave operation.

50.1.2 LPUART Instantiation

This device contains 3 LPUART modules. This section describes how each module is configured on this device.

Table 50-1. LPUART instantiations

Feature	Receive Buffer Size
LPUARTs instantiated	3
LPUART0 FIFO (RX & TX)	8 Bytes
LPUART1 FIFO (RX & TX)	8 Bytes
LPUART2 FIFO (RX & TX)	1 Byte

50.1.3 USB and VREGIN pin status detection and wakeup interrupt features

This device does not have a dedicated VBUS detect pin. For VBUS detection, use either VREGIN for bus-powered USB cases or a GPIO pin for both bus-powered and self-powered USB cases. Because the GPIO pins on this device do not directly support a 5V input, use an external resistive voltage divider to keep the input voltage within the valid range if a GPIO pin is used for VBUS detection.

This device does not have a dedicated OTG ID detect pin. For OTG ID pin detection, if needed, use a GPIO configured as an input pin with pullup enabled.

When the USB detects that there is no activity on the USB bus for more than 3 ms, the USBx_ISTAT[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 USBx_USBTRC0[USBRESMEN] bit enables this function.

The following wakeup features are also supported:

- In Stop/VLPS, the USB controller can generate an interrupt on VREGIN detection using the USBx_MISCCTRL[VFEDG_EN, VREDG_EN] and USBx_USBTRC0[VFEDG_DET, VREDG_DET] fields.
- In LLS/VLLS except VLLS1/0, VREGIN is an input pin to the LLWU and a transition on it can generate a wakeup.
- In LLS/VLLS, USB0_DP and USB0_DM are input pins to the LLWU and a transition on those pins can generate a wakeup provided the USB is powered and in host mode.
- In LLS/VLLS, the GPIO pins chosen to detect VBUS and OTG ID are also input to the LLWU, so transitions on them can generate a wakeup.

50.2 Introduction

50.2.1 Features

Features of the LPUART module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Programmable baud rates (13-bit modulo divider) with configurable oversampling ratio from 4x to 32x
- Transmit and receive baud rate can operate asynchronous to the bus clock:

    - Baud rate can be configured independently of the bus clock frequency
    - Supports operation in Stop modes
- Interrupt, DMA or polled operation:
    - Transmit data register empty and transmission complete
    - Receive data register full
    - Receive overrun, parity error, framing error, and noise error
    - Idle receiver detect
    - Active edge on receive pin
    - Break detect supporting LIN
    - Receive data match
- Hardware parity generation and checking
- Programmable 8-bit, 9-bit or 10-bit character length
- Programmable 1-bit or 2-bit stop bits
- Three receiver wakeup methods:
    - Idle line wakeup
    - Address mark wakeup
    - Receive data match
- Automatic address matching to reduce ISR overhead:
    - Address mark matching
    - Idle line address matching
    - Address match start, address match end
- Optional 13-bit break character generation / 11-bit break character detection
- Configurable idle length detection supporting 1, 2, 4, 8, 16, 32, 64 or 128 idle characters
- Selectable transmitter output and receiver input polarity
- Hardware flow control support for request to send (RTS) and clear to send (CTS) signals
- Selectable IrDA 1.4 return-to-zero-inverted (RZI) format with programmable pulse width
- Independent FIFO structure for transmit and receive
    - Separate configurable watermark for receive and transmit requests
    - Option for receiver to assert request after a configurable number of idle characters if receive FIFO is not empty

50.2.2 Modes of operation

50.2.2.1 Stop mode

The LPUART will remain functional during Stop mode, provided the asynchronous transmit and receive clock remains enabled. The LPUART can generate an interrupt or DMA request to cause a wakeup from Stop mode.

50.2.2.2 Wait mode

The LPUART can be configured to Stop in Wait modes, when the DOZEEN bit is set. The transmitter and receiver will finish transmitting/receiving the current word.

50.2.2.3 Debug mode

The LPUART remains functional in debug mode.

50.2.3 Signal Descriptions

Signal	Description	I/O
LPUART_TX	Transmit data. This pin is normally an output, but is an input (tristated) in single wire mode whenever the transmitter is disabled or transmit direction is configured for receive data.	I/O
LPUART_RX	Receive data.	I
LPUART_CTS	Clear to send.	I
LPUART_RTS	Request to send.	O

50.2.4 Block diagram

The following figure shows the transmitter portion of the LPUART.

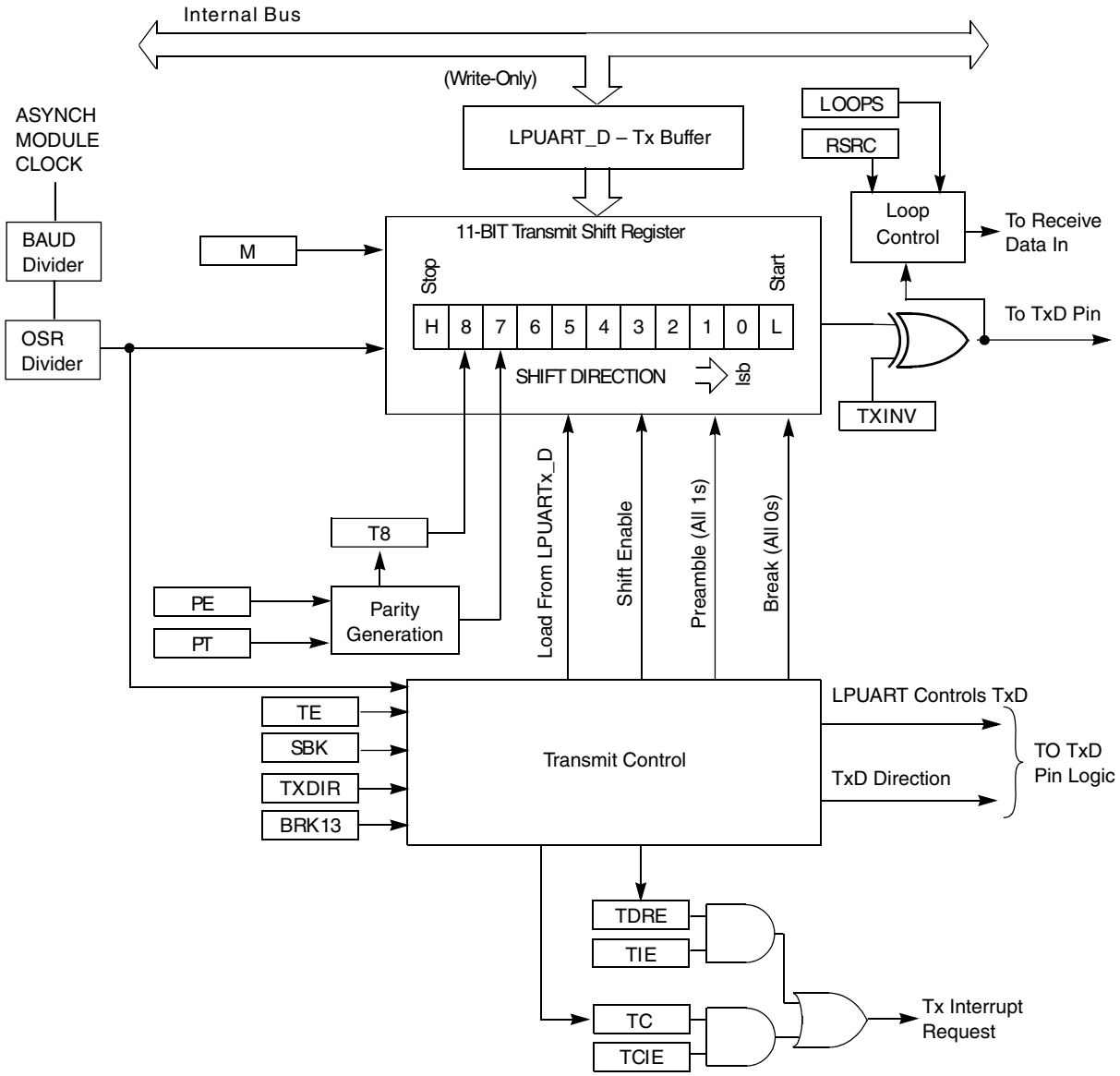


Figure 50-1. LPUART transmitter block diagram

The following figure shows the receiver portion of the LPUART.

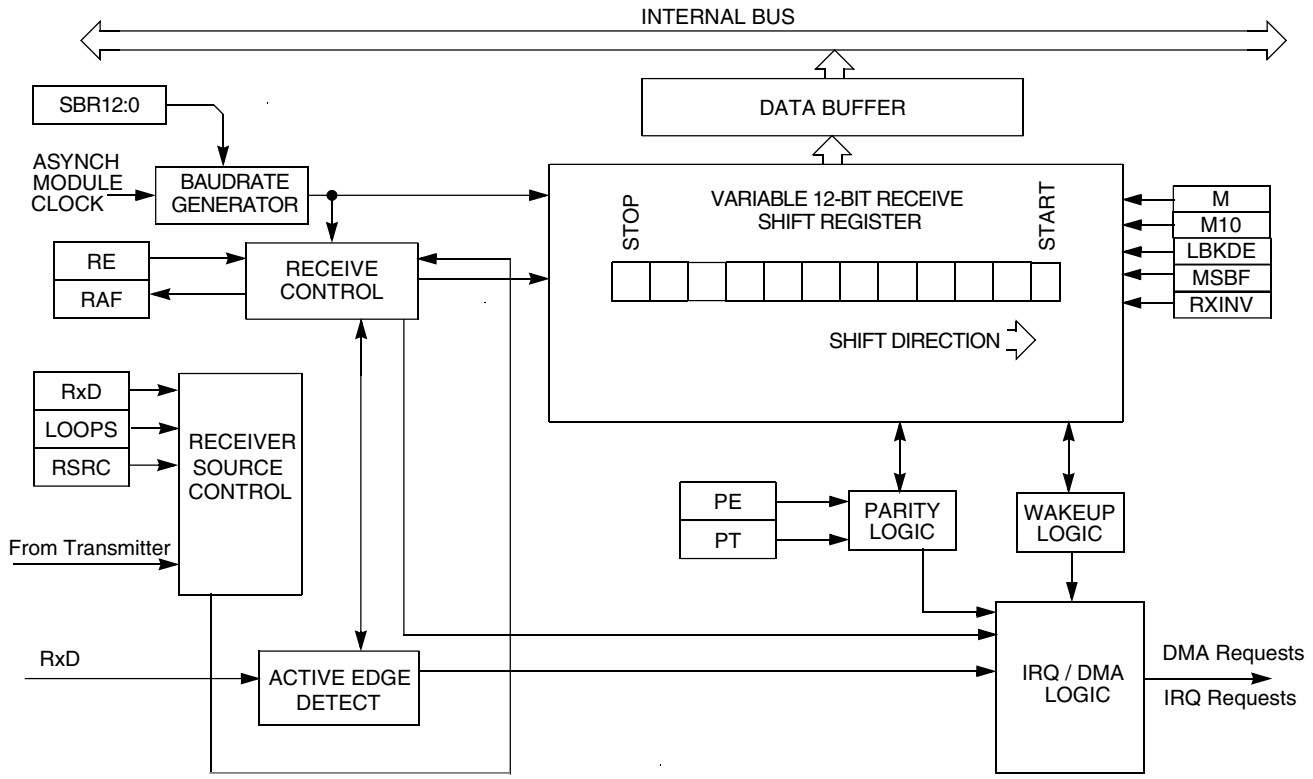


Figure 50-2. LPUART receiver block diagram

50.3 Register definition

The LPUART includes registers to control baud rate, select LPUART options, report LPUART status, and for transmit/receive data. Access to an address outside the valid memory map will generate a bus error.

LPUART memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4005_4000	LPUART Baud Rate Register (LPUART0_BAUD)	32	R/W	0F00_0004h	50.3.1/1479
4005_4004	LPUART Status Register (LPUART0_STAT)	32	R/W	00C0_0000h	50.3.2/1482
4005_4008	LPUART Control Register (LPUART0_CTRL)	32	R/W	0000_0000h	50.3.3/1486
4005_400C	LPUART Data Register (LPUART0_DATA)	32	R/W	0000_1000h	50.3.4/1491
4005_4010	LPUART Match Address Register (LPUART0_MATCH)	32	R/W	0000_0000h	50.3.5/1493
4005_4014	LPUART Modem IrDA Register (LPUART0_MODIR)	32	R/W	0000_0000h	50.3.6/1493
4005_4018	LPUART FIFO Register (LPUART0_FIFO)	32	R/W	See section	50.3.7/1496
4005_401C	LPUART Watermark Register (LPUART0_WATER)	32	R/W	0000_0000h	50.3.8/1499
4005_5000	LPUART Baud Rate Register (LPUART1_BAUD)	32	R/W	0F00_0004h	50.3.1/1479

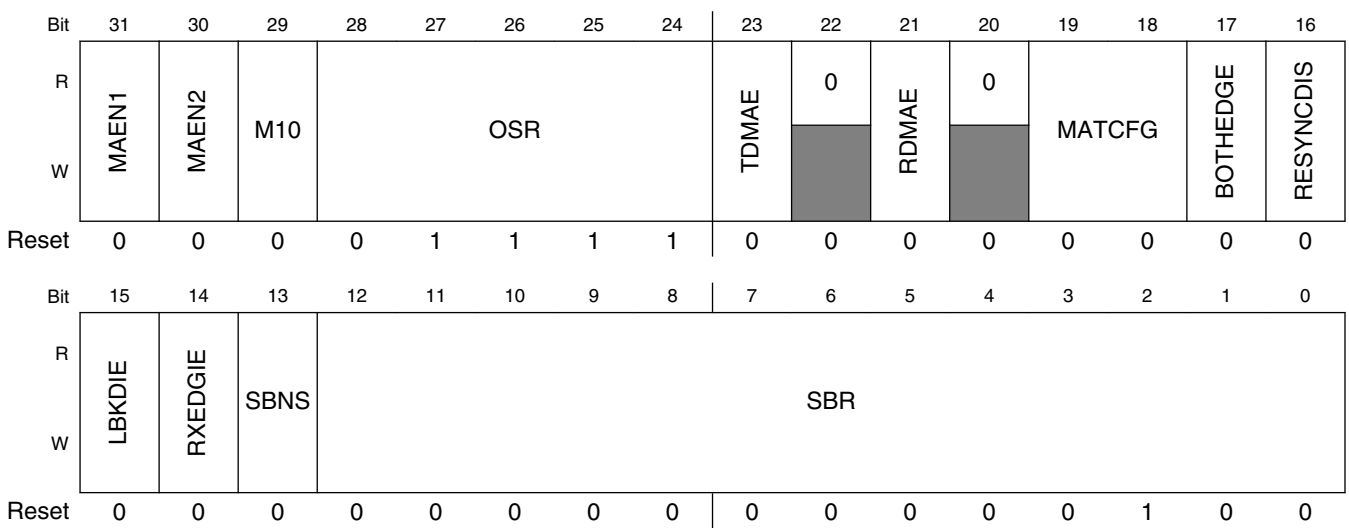
Table continues on the next page...

LPUART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4005_5004	LPUART Status Register (LPUART1_STAT)	32	R/W	00C0_0000h	50.3.2/1482
4005_5008	LPUART Control Register (LPUART1_CTRL)	32	R/W	0000_0000h	50.3.3/1486
4005_500C	LPUART Data Register (LPUART1_DATA)	32	R/W	0000_1000h	50.3.4/1491
4005_5010	LPUART Match Address Register (LPUART1_MATCH)	32	R/W	0000_0000h	50.3.5/1493
4005_5014	LPUART Modem IrDA Register (LPUART1_MODIR)	32	R/W	0000_0000h	50.3.6/1493
4005_5018	LPUART FIFO Register (LPUART1_FIFO)	32	R/W	See section	50.3.7/1496
4005_501C	LPUART Watermark Register (LPUART1_WATER)	32	R/W	0000_0000h	50.3.8/1499
4005_6000	LPUART Baud Rate Register (LPUART2_BAUD)	32	R/W	0F00_0004h	50.3.1/1479
4005_6004	LPUART Status Register (LPUART2_STAT)	32	R/W	00C0_0000h	50.3.2/1482
4005_6008	LPUART Control Register (LPUART2_CTRL)	32	R/W	0000_0000h	50.3.3/1486
4005_600C	LPUART Data Register (LPUART2_DATA)	32	R/W	0000_1000h	50.3.4/1491
4005_6010	LPUART Match Address Register (LPUART2_MATCH)	32	R/W	0000_0000h	50.3.5/1493
4005_6014	LPUART Modem IrDA Register (LPUART2_MODIR)	32	R/W	0000_0000h	50.3.6/1493
4005_6018	LPUART FIFO Register (LPUART2_FIFO)	32	R/W	See section	50.3.7/1496
4005_601C	LPUART Watermark Register (LPUART2_WATER)	32	R/W	0000_0000h	50.3.8/1499

50.3.1 LPUART Baud Rate Register (LPUARTx_BAUD)

Address: Base address + 0h offset



LPUARTx_BAUD field descriptions

Field	Description
31 MAEN1	Match Address Mode Enable 1

Table continues on the next page...

LPUARTx_BAUD field descriptions (continued)

Field	Description
	0 Normal operation. 1 Enables automatic address matching or data matching mode for MATCH[MA1].
30 MAEN2	Match Address Mode Enable 2 0 Normal operation. 1 Enables automatic address matching or data matching mode for MATCH[MA2].
29 M10	10-bit Mode select The M10 bit causes a tenth bit to be part of the serial transmission. This bit should only be changed when the transmitter and receiver are both disabled. 0 Receiver and transmitter use 8-bit or 9-bit data characters. 1 Receiver and transmitter use 10-bit data characters.
28–24 OSR	Oversampling Ratio This field configures the oversampling ratio for the receiver between 4x (00011) and 32x (11111). Writing an invalid oversampling ratio (for example, a value not between 4x and 32x) will default to an oversampling ratio of 16 (01111). The OSR field should only be changed when the transmitter and receiver are both disabled. Note that the oversampling ratio = OSR + 1.
23 TDMAE	Transmitter DMA Enable TDMAE configures the transmit data register empty flag, LPUART_STAT[TDRE], to generate a DMA request. 0 DMA request disabled. 1 DMA request enabled.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 RDMAE	Receiver Full DMA Enable RDMAE configures the receiver data register full flag, LPUART_STAT[RDRF], to generate a DMA request. 0 DMA request disabled. 1 DMA request enabled.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–18 MATCFG	Match Configuration Configures the match addressing mode used. 00 Address Match Wakeup 01 Idle Match Wakeup 10 Match On and Match Off 11 Enables RWU on Data Match and Match On/Off for transmitter CTS input
17 BOTHEDGE	Both Edge Sampling Enables sampling of the received data on both edges of the baud rate clock, effectively doubling the number of times the receiver samples the input data for a given oversampling ratio. This bit must be set for oversampling ratios between x4 and x7 and is optional for higher oversampling ratios. This bit should only be changed when the receiver is disabled.

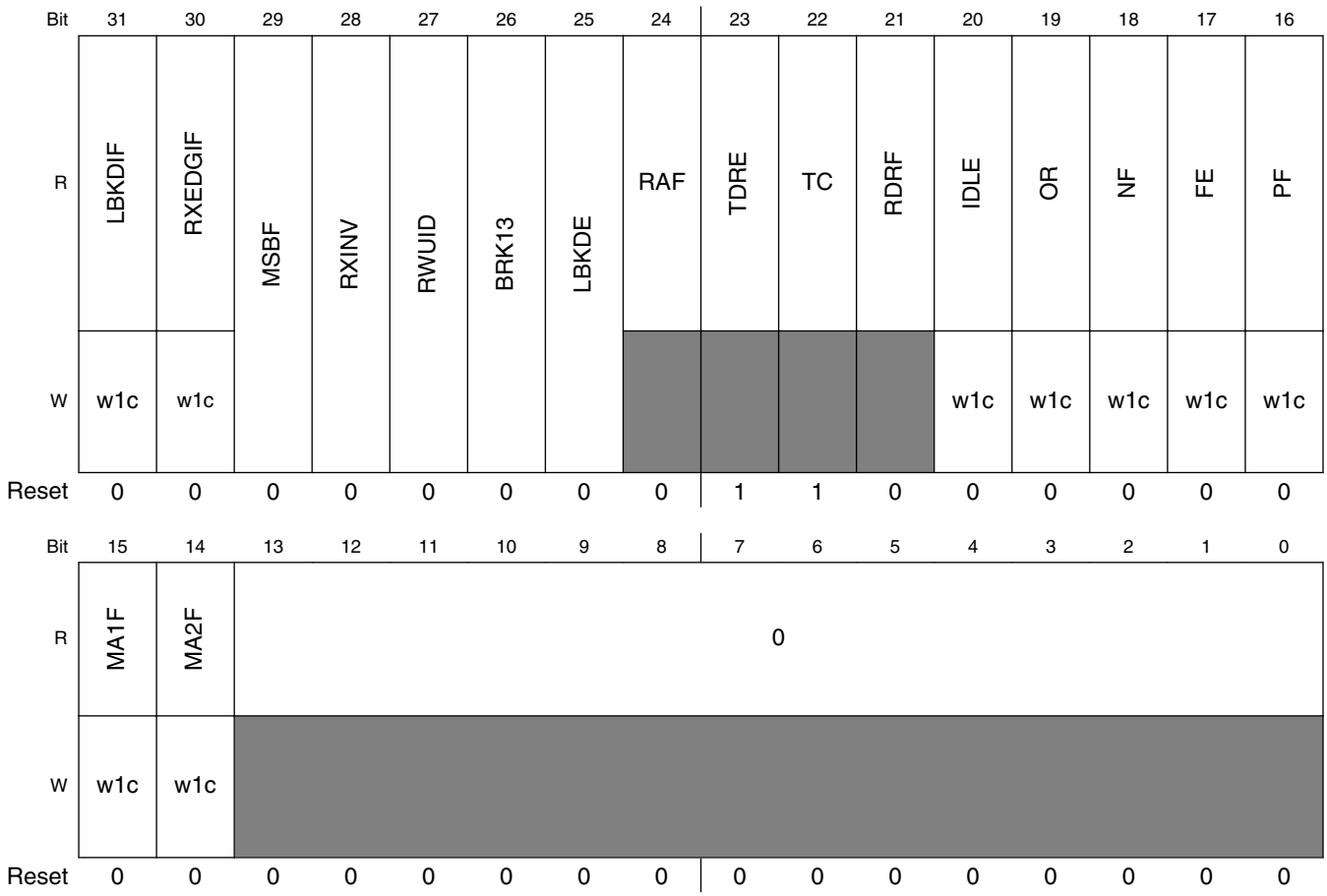
Table continues on the next page...

LPUARTx_BAUD field descriptions (continued)

Field	Description
	0 Receiver samples input data using the rising edge of the baud rate clock. 1 Receiver samples input data using the rising and falling edge of the baud rate clock.
16 RESYNCDIS	Resynchronization Disable When set, disables the resynchronization of the received data word when a data one followed by data zero transition is detected. This bit should only be changed when the receiver is disabled. 0 Resynchronization during received data word is supported 1 Resynchronization during received data word is disabled
15 LBKDIE	LIN Break Detect Interrupt Enable LBKDIE enables the LIN break detect flag, LBKDIF, to generate interrupt requests. 0 Hardware interrupts from LPUART_STAT[LBKDIF] disabled (use polling). 1 Hardware interrupt requested when LPUART_STAT[LBKDIF] flag is 1.
14 RXEDGIE	RX Input Active Edge Interrupt Enable Enables the receive input active edge, RXEDGIF, to generate interrupt requests. Changing CTRL[LOOP] or CTRL[RSRC] when RXEDGIE is set can cause the RXEDGIF to set. 0 Hardware interrupts from LPUART_STAT[RXEDGIF] disabled (use polling). 1 Hardware interrupt requested when LPUART_STAT[RXEDGIF] flag is 1.
13 SBNS	Stop Bit Number Select SBNS determines whether data characters are one or two stop bits. This bit should only be changed when the transmitter and receiver are both disabled. 0 One stop bit. 1 Two stop bits.
SBR	Baud Rate Modulo Divisor. The 13 bits in SBR[12:0] set the modulo divide rate for the baud rate generator. When SBR is 1 - 8191, the baud rate equals "baud clock / ((OSR+1) × SBR)". The 13-bit baud rate setting [SBR12:SBR0] must only be updated when the transmitter and receiver are both disabled (LPUART_CTRL[RE] and LPUART_CTRL[TE] are both 0).

50.3.2 LPUART Status Register (LPUARTx_STAT)

Address: Base address + 4h offset



LPUARTx_STAT field descriptions

Field	Description
31 LBKDIF	LIN Break Detect Interrupt Flag LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a 1 to it. 0 No LIN break character has been detected. 1 LIN break character has been detected.
30 RXEDGIF	LPUART_RX Pin Active Edge Interrupt Flag RXEDGIF is set when an active edge, falling if RXINV = 0, rising if RXINV=1, on the LPUART_RX pin occurs. RXEDGIF is cleared by writing a 1 to it. 0 No active edge on the receive pin has occurred. 1 An active edge on the receive pin has occurred.
29 MSBF	MSB First

Table continues on the next page...

LPUARTx_STAT field descriptions (continued)

Field	Description
	Setting this bit reverses the order of the bits that are transmitted and received on the wire. This bit does not affect the polarity of the bits, the location of the parity bit or the location of the start or stop bits. This bit should only be changed when the transmitter and receiver are both disabled. 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. 1 MSB (bit9, bit8, bit7 or bit6) is the first bit that is transmitted following the start bit depending on the setting of CTRL[M], CTRL[PE] and BAUD[M10]. Further, the first bit received after the start bit is identified as bit9, bit8, bit7 or bit6 depending on the setting of CTRL[M] and CTRL[PE].
28 RXINV	Receive Data Inversion Setting this bit reverses the polarity of the received data input. **NOTE:** Setting RXINV inverts the LPUART_RX input for all cases: data bits, start and stop bits, break, and idle. 0 Receive data not inverted. 1 Receive data inverted.
27 RWUID	Receive Wake Up Idle Detect For RWU on idle character, RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit. For address match wakeup, RWUID controls if the IDLE bit is set when the address does not match. This bit should only be changed when the receiver is disabled. 0 During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character. During address match wakeup, the IDLE bit does not get set when an address does not match. 1 During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character. During address match wakeup, the IDLE bit does get set when an address does not match.
26 BRK13	Break Character Generation Length BRK13 selects a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit. This bit should only be changed when the transmitter is disabled. 0 Break character is transmitted with length of 10 bit times (if M = 0, SBNS = 0) or 11 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 12 (if M = 1, SBNS = 1 or M10 = 1, SNBS = 0) or 13 (if M10 = 1, SNBS = 1). 1 Break character is transmitted with length of 13 bit times (if M = 0, SBNS = 0) or 14 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 15 (if M = 1, SBNS = 1 or M10 = 1, SNBS = 0) or 16 (if M10 = 1, SNBS = 1).
25 LBKDE	LIN Break Detection Enable LBKDE selects a longer break character detection length. While LBKDE is set, receive data is not stored in the receive data buffer. 0 Break character is detected at length 10 bit times (if M = 0, SBNS = 0) or 11 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 12 (if M = 1, SBNS = 1 or M10 = 1, SNBS = 0) or 13 (if M10 = 1, SNBS = 1). 1 Break character is detected at length of 11 bit times (if M = 0, SBNS = 0) or 12 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 14 (if M = 1, SBNS = 1 or M10 = 1, SNBS = 0) or 15 (if M10 = 1, SNBS = 1).
24 RAF	Receiver Active Flag RAF is set when the receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line.

Table continues on the next page...

LPUARTx_STAT field descriptions (continued)

Field	Description
	0 LPUART receiver idle waiting for a start bit. 1 LPUART receiver active (LPUART_RX input not idle).
23 TDRE	Transmit Data Register Empty Flag When the transmit FIFO is enabled, TDRE will set when the number of datawords in the transmit FIFO (LPUART_DATA) is equal to or less than the number indicated by LPUART_WATER[TXWATER]). To clear TDRE, write to the LPUART data register (LPUART_DATA) until the number of words in the transmit FIFO is greater than the number indicated by LPUART_WATER[TXWATER]. When the transmit FIFO is disabled,TDRE will set when the transmit data register (LPUART_DATA) is empty. To clear TDRE, write to the LPUART data register (LPUART_DATA). TDRE is not affected by a character that is in the process of being transmitted, it is updated at the start of each transmitted character. 0 Transmit data buffer full. 1 Transmit data buffer empty.
22 TC	Transmission Complete Flag TC is cleared when there is a transmission in progress or when a preamble or break character is loaded. 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 writing to LPUART_DATA to transmit new data, queuing a preamble by clearing and then setting LPUART_CTRL[TE], queuing a break character by writing 1 to LPUART_CTRL[SBK]. 0 Transmitter active (sending data, a preamble, or a break). 1 Transmitter idle (transmission activity complete).
21 RDRF	Receive Data Register Full Flag When the receive FIFO is enabled, RDRF is set when the number of datawords in the receive buffer is greater than the number indicated by LPUART_WATER[RXWATER]. To clear RDRF, read LPUART_DATA until the number of datawords in the receive data buffer is equal to or less than the number indicated by LPUART_WATER[RXWATER]. When the receive FIFO is disabled, RDRF is set when the receive buffer (LPUART_DATA) is full. To clear RDRF, read the LPUART_DATA register. A character that is in the process of being received does not cause a change in RDRF until the entire character is received. Even if RDRF is set, the character will continue to be received until an overrun condition occurs once the entire character is received. 0 Receive data buffer empty. 1 Receive data buffer full.
20 IDLE	Idle Line Flag IDLE is set when the LPUART receive line becomes idle for a full character time after a period of activity. When ILT is cleared, the receiver starts counting idle bit times after the start bit. If the receive character is all 1s, these bit times and the stop bits time count toward the full character time of logic high, 10 to 13 bit times, needed for the receiver to detect an idle line. When ILT is set, the receiver doesn't start counting idle bit times until after the stop bits. The stop bits and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line. To clear IDLE, write logic 1 to the IDLE flag. After IDLE has been cleared, it cannot become set again until after a new character has been stored in the receive buffer or a LIN break character has set the LBKDIF flag . IDLE is set only once even if the receive line remains idle for an extended period. 0 No idle line detected. 1 Idle line was detected.

Table continues on the next page...

LPUARTx_STAT field descriptions (continued)

Field	Description
19 OR	Receiver Overrun Flag 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 LPUART data registers is not affected. If LBKDE is enabled and a LIN Break is detected, the OR field asserts if LBKDIF is not cleared before the next data character is received. While the OR flag is set, no additional data is stored in the data buffer even if sufficient room exists. To clear OR, write logic 1 to the OR flag. 0 No overrun. 1 Receive overrun (new LPUART data lost).
18 NF	Noise Flag The advanced sampling technique used in the receiver takes three samples in each of the received bits. If any of these samples disagrees with the rest of the samples within any bit time in the frame then noise is detected for that character. NF is set whenever the next character to be read from LPUART_DATA was received with noise detected within the character. To clear NF, write logic one to the NF. 0 No noise detected. 1 Noise detected in the received character in LPUART_DATA.
17 FE	Framing Error Flag FE is set whenever the next character to be read from LPUART_DATA was received with logic 0 detected where a stop bit was expected. To clear FE, write logic one to the FE. 0 No framing error detected. This does not guarantee the framing is correct. 1 Framing error.
16 PF	Parity Error Flag PF is set whenever the next character to be read from LPUART_DATA was received when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, write a logic one to the PF. 0 No parity error. 1 Parity error.
15 MA1F	Match 1 Flag MA1F is set whenever the next character to be read from LPUART_DATA matches MA1. To clear MA1F, write a logic one to the MA1F. 0 Received data is not equal to MA1 1 Received data is equal to MA1
14 MA2F	Match 2 Flag MA2F is set whenever the next character to be read from LPUART_DATA matches MA2. To clear MA2F, write a logic one to the MA2F. 0 Received data is not equal to MA2 1 Received data is equal to MA2
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

50.3.3 LPUART Control Register (LPUARTx_CTRL)

This read/write register controls various optional features of the LPUART system. This register should only be altered when the transmitter and receiver are both disabled.

Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	R8T9	R9T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MA1IE	MA2IE	0			IDLECFG			LOOPS	DOZEEN	RSRC	M	WAKE	ILT	PE	PT
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LPUARTx_CTRL field descriptions

Field	Description
31 R8T9	Receive Bit 8 / Transmit Bit 9 R8 is the ninth data bit received when the LPUART is configured for 9-bit or 10-bit data formats. When reading 9-bit or 10-bit data, read R8 before reading LPUART_DATA. T9 is the tenth data bit received when the LPUART is configured for 10-bit data formats. When writing 10-bit data, write T9 before writing LPUART_DATA. If T9 does not need to change from its previous value, such as when it is used to generate address mark or parity, they it need not be written each time LPUART_DATA is written.
30 R9T8	Receive Bit 9 / Transmit Bit 8 R9 is the tenth data bit received when the LPUART is configured for 10-bit data formats. When reading 10-bit data, read R9 before reading LPUART_DATA T8 is the ninth data bit received when the LPUART is configured for 9-bit or 10-bit data formats. When writing 9-bit or 10-bit data, write T8 before writing LPUART_DATA. If T8 does not need to change from its previous value, such as when it is used to generate address mark or parity, they it need not be written each time LPUART_DATA is written.
29 TXDIR	LPUART_TX Pin Direction in Single-Wire Mode When the LPUART is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the LPUART_TX pin. When clearing TXDIR, the transmitter will finish receiving the current character (if any) before the receiver starts receiving data from the LPUART_TX pin. 0 LPUART_TX pin is an input in single-wire mode. 1 LPUART_TX pin is an output in single-wire mode.

Table continues on the next page...

LPUARTx_CTRL field descriptions (continued)

Field	Description
28 TXINV	Transmit Data Inversion Setting this bit reverses the polarity of the transmitted data output. **NOTE:** Setting TXINV inverts the LPUART_TX output for all cases: data bits, start and stop bits, break, and idle. 0 Transmit data not inverted. 1 Transmit data inverted.
27 ORIE	Overrun Interrupt Enable This bit enables the overrun flag (OR) to generate hardware interrupt requests. 0 OR interrupts disabled; use polling. 1 Hardware interrupt requested when OR is set.
26 NEIE	Noise Error Interrupt Enable This bit enables the noise flag (NF) to generate hardware interrupt requests. 0 NF interrupts disabled; use polling. 1 Hardware interrupt requested when NF is set.
25 FEIE	Framing Error Interrupt Enable This bit enables the framing error flag (FE) to generate hardware interrupt requests. 0 FE interrupts disabled; use polling. 1 Hardware interrupt requested when FE is set.
24 PEIE	Parity Error Interrupt Enable This bit enables the parity error flag (PF) to generate hardware interrupt requests. 0 PF interrupts disabled; use polling). 1 Hardware interrupt requested when PF is set.
23 TIE	Transmit Interrupt Enable Enables STAT[TDRE] to generate interrupt requests. 0 Hardware interrupts from TDRE disabled; use polling. 1 Hardware interrupt requested when TDRE flag is 1.
22 TCIE	Transmission Complete Interrupt Enable for TCIE enables the transmission complete flag, TC, to generate interrupt requests. 0 Hardware interrupts from TC disabled; use polling. 1 Hardware interrupt requested when TC flag is 1.
21 RIE	Receiver Interrupt Enable Enables STAT[RDRF] to generate interrupt requests. 0 Hardware interrupts from RDRF disabled; use polling. 1 Hardware interrupt requested when RDRF flag is 1.

Table continues on the next page...

LPUARTx_CTRL field descriptions (continued)

Field	Description
20 ILIE	Idle Line Interrupt Enable ILIE enables the idle line flag, STAT[IDLE], to generate interrupt requests. 0 Hardware interrupts from IDLE disabled; use polling. 1 Hardware interrupt requested when IDLE flag is 1.
19 TE	Transmitter Enable Enables the LPUART transmitter. TE can also be used to queue an idle preamble by clearing and then setting TE. When TE is cleared, this register bit will read as 1 until the transmitter has completed the current character and the LPUART_TX pin is tristated. 0 Transmitter disabled. 1 Transmitter enabled.
18 RE	Receiver Enable Enables the LPUART receiver. When RE is written to 0, this register bit will read as 1 until the receiver finishes receiving the current character (if any). 0 Receiver disabled. 1 Receiver enabled.
17 RWU	Receiver Wakeup Control This field can be set to place the LPUART receiver in a standby state. RWU automatically clears when an RWU event occurs, that is, an IDLE event when CTRL[WAKE] is clear or an address match when CTRL[WAKE] is set with STAT[RWUID] is clear. **NOTE:** RWU must be set only with CTRL[WAKE] = 0 (wakeup on idle) if the channel is currently not idle. This can be determined by STAT[RAF]. If the flag is set to wake up an IDLE event and the channel is already idle, it is possible that the LPUART 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 be reasserted. 0 Normal receiver operation. 1 LPUART receiver in standby waiting for wakeup condition.
16 SBK	Send Break Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 10 to 13, or 13 to 16 if LPUART_STATBRK13] is set, bit times of logic 0 are queued as long as SBK is set. Depending on the timing of the set and clear of SBK relative to the information currently being transmitted, a second break character may be queued before software clears SBK. 0 Normal transmitter operation. 1 Queue break character(s) to be sent.
15 MA1IE	Match 1 Interrupt Enable 0 MA1F interrupt disabled 1 MA1F interrupt enabled
14 MA2IE	Match 2 Interrupt Enable 0 MA2F interrupt disabled 1 MA2F interrupt enabled

Table continues on the next page...

LPUARTx_CTRL field descriptions (continued)

Field	Description
13–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–8 IDLECFG	Idle Configuration Configures the number of idle characters that must be received before the IDLE flag is set. 000 1 idle character 001 2 idle characters 010 4 idle characters 011 8 idle characters 100 16 idle characters 101 32 idle characters 110 64 idle characters 111 128 idle characters
7 LOOPS	Loop Mode Select When LOOPS is set, the LPUART_RX pin is disconnected from the LPUART and the transmitter output is internally connected to the receiver input. The transmitter and the receiver must be enabled to use the loop function. 0 Normal operation - LPUART_RX and LPUART_TX use separate pins. 1 Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input (see RSRC bit).
6 DOZEEN	Doze Enable 0 LPUART is enabled in Doze mode. 1 LPUART is disabled in Doze mode.
5 RSRC	Receiver Source Select 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. 0 Provided LOOPS is set, RSRC is cleared, selects internal loop back mode and the LPUART does not use the LPUART_RX pin. 1 Single-wire LPUART mode where the LPUART_TX pin is connected to the transmitter output and receiver input.
4 M	9-Bit or 8-Bit Mode Select 0 Receiver and transmitter use 8-bit data characters. 1 Receiver and transmitter use 9-bit data characters.
3 WAKE	Receiver Wakeup Method Select Determines which condition wakes the LPUART when RWU=1: • Address mark in the most significant bit position of a received data character, or • An idle condition on the receive pin input signal. 0 Configures RWU for idle-line wakeup. 1 Configures RWU with address-mark wakeup.
2 ILT	Idle Line Type Select

Table continues on the next page...

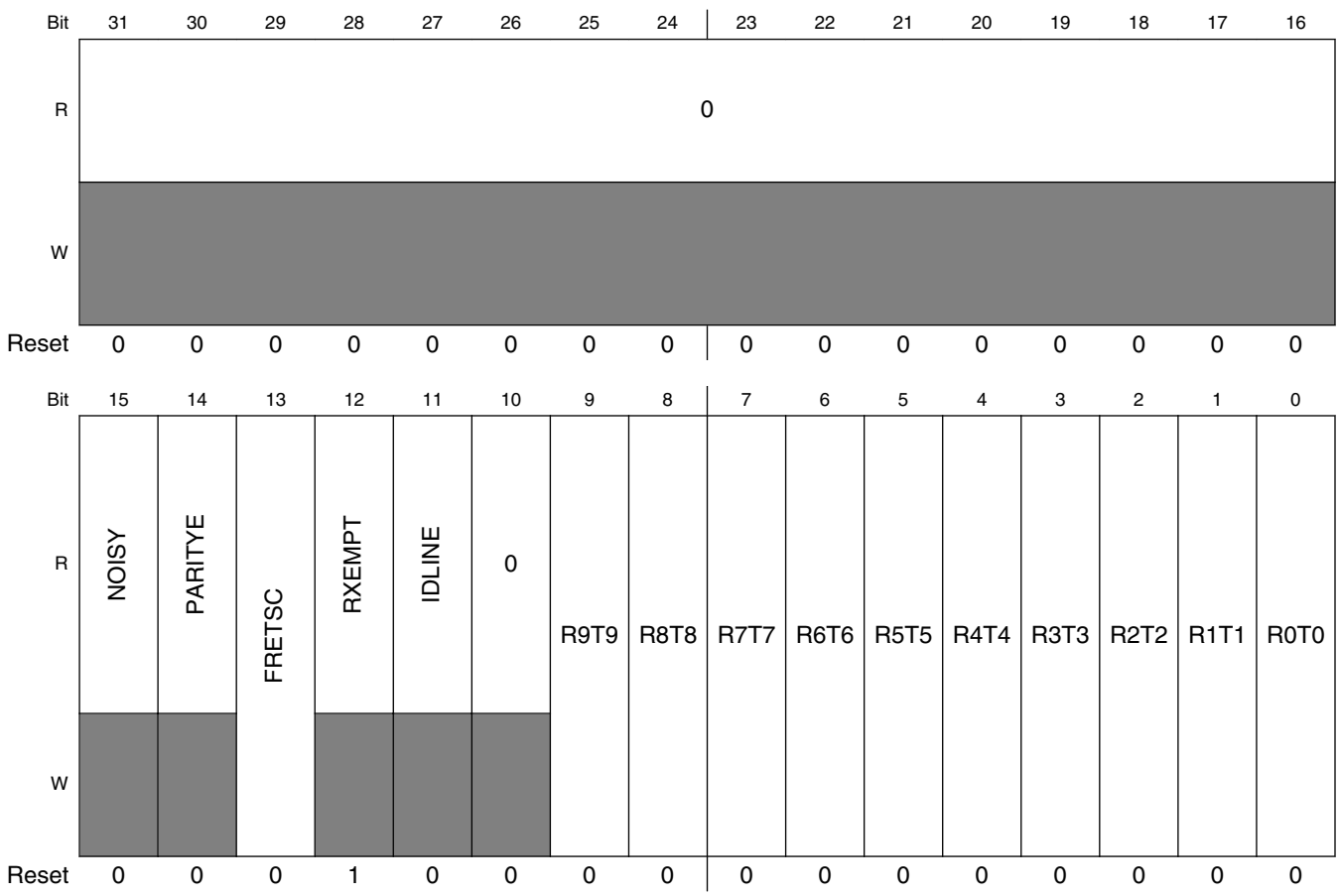
LPUARTx_CTRL field descriptions (continued)

Field	Description
	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. **NOTE:** In case the LPUART is programmed with ILT = 1, a logic 0 is automatically shifted after a received stop bit, therefore resetting the idle count. 0 Idle character bit count starts after start bit. 1 Idle character bit count starts after stop bit.
1 PE	Parity Enable Enables hardware parity generation and checking. When parity is enabled, the bit immediately before the stop bit is treated as the parity bit. 0 No hardware parity generation or checking. 1 Parity enabled.
0 PT	Parity Type Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even. 0 Even parity. 1 Odd parity.

50.3.4 LPUART Data Register (LPUARTx_DATA)

This register is actually two separate registers. Reads return the contents of the read-only receive data buffer and writes go to the write-only transmit data buffer. Reads and writes of this register are also involved in the automatic flag clearing mechanisms for some of the LPUART status flags.

Address: Base address + Ch offset



LPUARTx_DATA field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 NOISY	The current received dataword contained in DATA[R9:R0] was received with noise. 0 The dataword was received without noise. 1 The data was received with noise.
14 PARITYE	The current received dataword contained in DATA[R9:R0] was received with a parity error.

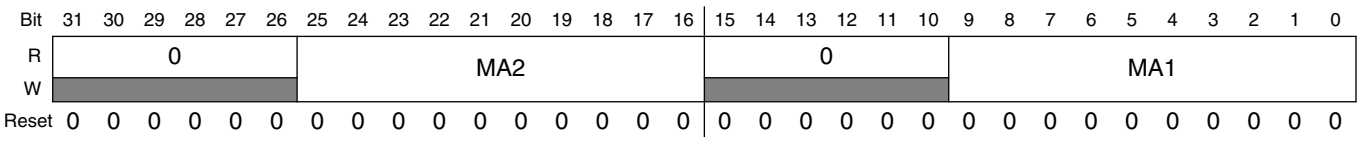
Table continues on the next page...

LPUARTx_DATA field descriptions (continued)

Field	Description
	0 The dataword was received without a parity error. 1 The dataword was received with a parity error.
13 FRETSC	Frame Error / Transmit Special Character For reads, indicates the current received dataword contained in DATA[R9:R0] was received with a frame error. For writes, indicates a break or idle character is to be transmitted instead of the contents in DATA[T9:T0]. T9 is used to indicate a break character when 0 and a idle character when 1, he contents of DATA[T8:T0] should be zero. 0 The dataword was received without a frame error on read, transmit a normal character on write. 1 The dataword was received with a frame error, transmit an idle or break character on transmit.
12 RXEMPT	Receive Buffer Empty Asserts when there is no data in the receive buffer. This field does not take into account data that is in the receive shift register. 0 Receive buffer contains valid data. 1 Receive buffer is empty, data returned on read is not valid.
11 IDLINE	Idle Line Indicates the receiver line was idle before receiving the character in DATA[9:0]. Unlike the IDLE flag, this bit can set for the first character received when the receiver is first enabled. 0 Receiver was not idle before receiving this character. 1 Receiver was idle before receiving this character.
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 R9T9	Read receive data buffer 9 or write transmit data buffer 9.
8 R8T8	Read receive data buffer 8 or write transmit data buffer 8.
7 R7T7	Read receive data buffer 7 or write transmit data buffer 7.
6 R6T6	Read receive data buffer 6 or write transmit data buffer 6.
5 R5T5	Read receive data buffer 5 or write transmit data buffer 5.
4 R4T4	Read receive data buffer 4 or write transmit data buffer 4.
3 R3T3	Read receive data buffer 3 or write transmit data buffer 3.
2 R2T2	Read receive data buffer 2 or write transmit data buffer 2.
1 R1T1	Read receive data buffer 1 or write transmit data buffer 1.
0 R0T0	Read receive data buffer 0 or write transmit data buffer 0.

50.3.5 LPUART Match Address Register (LPUARTx_MATCH)

Address: Base address + 10h offset



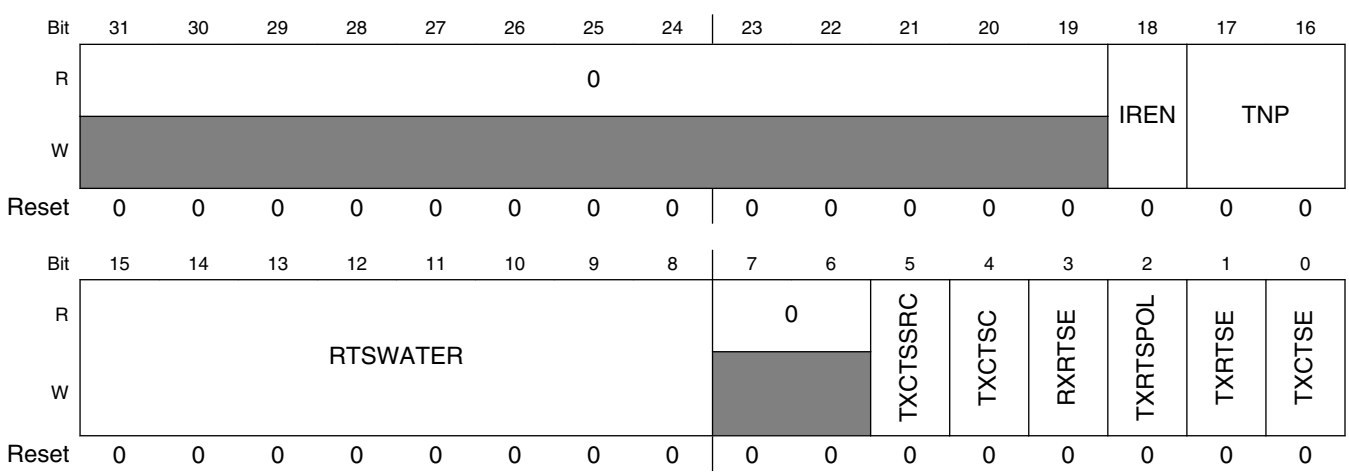
LPUARTx_MATCH field descriptions

Field	Description
31–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–16 MA2	Match Address 2 The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated BAUD[MAEN] bit is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded. Software should only write a MA register when the associated BAUD[MAEN] bit is clear.
15–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
MA1	Match Address 1 The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated BAUD[MAEN] bit is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded. Software should only write a MA register when the associated BAUD[MAEN] bit is clear.

50.3.6 LPUART Modem IrDA Register (LPUARTx_MODIR)

The MODEM register controls options for setting the modem configuration.

Address: Base address + 14h offset



LPUARTx_MODIR field descriptions

Field	Description
31–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 IREN	Infrared enable Enables/disables the infrared modulation/demodulation. 0 IR disabled. 1 IR enabled.
17–16 TNP	Transmitter narrow pulse Enables whether the LPUART transmits a 1/OSR, 2/OSR, 3/OSR or 4/OSR narrow pulse. 00 1/OSR. 01 2/OSR. 10 3/OSR. 11 4/OSR.
15–8 RTSWATER	Receive RTS Configuration Configures the point at which the RX RTS output negates based on the number of additional characters that can be stored in the Receive FIFO. When configured to 0, RTS negates when the the start bit is detetected for the character that will cause the FIFO to become full. 0 RTS asserts when the receiver FIFO is full or receiving a character that causes the FIFO to become full. 1 RTS asserts when the receive FIFO is less than or equal to the RXWATER configuration and negates when the receive FIFO is greater than the RXWATER configuration.
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 TXCTSSRC	Transmit CTS Source Configures the source of the CTS input. 0 CTS input is the LPUART_CTS pin. 1 CTS input is the inverted Receiver Match result.
4 TXCTSC	Transmit CTS Configuration Configures if the CTS state is checked at the start of each character or only when the transmitter is idle. 0 CTS input is sampled at the start of each character. 1 CTS input is sampled when the transmitter is idle.
3 RXRTSE	Receiver request-to-send enable Allows the RTS output to control the CTS input of the transmitting device to prevent receiver overrun. **NOTE:** Do not set both RXRTSE and TXRTSE. 0 The receiver has no effect on RTS. 1 RTS assertion is configured by the RTSWATER field
2 TXRTSPOL	Transmitter request-to-send polarity 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.

Table continues on the next page...

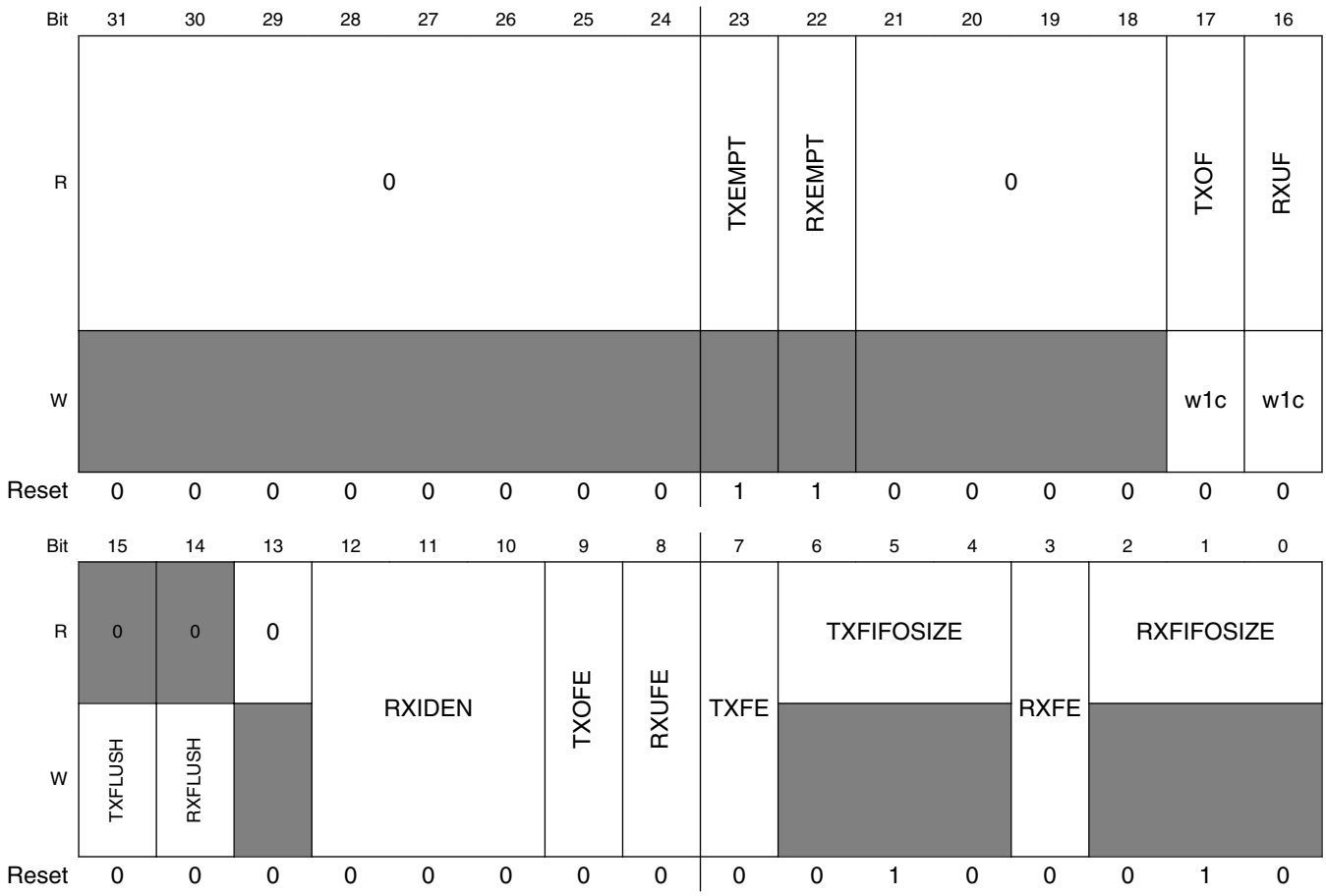
LPUARTx_MODIR field descriptions (continued)

Field	Description
	0 Transmitter RTS is active low. 1 Transmitter RTS is active high.
1 TXRTSE	Transmitter request-to-send enable Controls RTS before and after a transmission. 0 The transmitter has no effect on RTS. 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.
0 TXCTSE	Transmitter clear-to-send enable TXCTSE controls the operation of the transmitter. TXCTSE can be set independently from the state of TXRTSE and RXRTSE. 0 CTS has no effect on the transmitter. 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.

50.3.7 LPUART FIFO Register (LPUARTx_FIFO)

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 CTRL[RE] and CTRL[TE] are cleared/not set and when the data buffer/FIFO is empty.

Address: Base address + 18h offset



LPUARTx_FIFO field descriptions

Field	Description
31–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 TXEMPT	Transmit Buffer/FIFO Empty 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. 0 Transmit buffer is not empty. 1 Transmit buffer is empty.

Table continues on the next page...

LPUARTx_FIFO field descriptions (continued)

Field	Description
22 RXEMPT	Receive Buffer/FIFO Empty 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. 0 Receive buffer is not empty. 1 Receive buffer is empty.
21–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 TXOF	Transmitter Buffer Overflow Flag Indicates that more data has been written to the transmit buffer than it can hold. This field will assert regardless of the value of TXOFE. However, an interrupt will be issued to the host only if TXOFE is set. This flag is cleared by writing a 1. 0 No transmit buffer overflow has occurred since the last time the flag was cleared. 1 At least one transmit buffer overflow has occurred since the last time the flag was cleared.
16 RXUF	Receiver Buffer Underflow Flag Indicates that more data has been read from the receive buffer than was present. This field will assert regardless of the value of RXUFE. However, an interrupt will be issued to the host only if RXUFE is set. This flag is cleared by writing a 1. 0 No receive buffer underflow has occurred since the last time the flag was cleared. 1 At least one receive buffer underflow has occurred since the last time the flag was cleared.
15 TXFLUSH	Transmit FIFO/Buffer Flush 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. 0 No flush operation occurs. 1 All data in the transmit FIFO/Buffer is cleared out.
14 RXFLUSH	Receive FIFO/Buffer Flush 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. 0 No flush operation occurs. 1 All data in the receive FIFO/buffer is cleared out.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–10 RXIDEN	Receiver Idle Empty Enable When set, enables the assertion of RDRF when the receiver is idle for a number of idle characters and the FIFO is not empty. 000 Disable RDRF assertion due to partially filled FIFO when receiver is idle. 001 Enable RDRF assertion due to partially filled FIFO when receiver is idle for 1 character. 010 Enable RDRF assertion due to partially filled FIFO when receiver is idle for 2 characters. 011 Enable RDRF assertion due to partially filled FIFO when receiver is idle for 4 characters. 100 Enable RDRF assertion due to partially filled FIFO when receiver is idle for 8 characters. 101 Enable RDRF assertion due to partially filled FIFO when receiver is idle for 16 characters.

Table continues on the next page...

LPUARTx_FIFO field descriptions (continued)

Field	Description
	110 Enable RDRF assertion due to partially filled FIFO when receiver is idle for 32 characters. 111 Enable RDRF assertion due to partially filled FIFO when receiver is idle for 64 characters.
9 TXOFE	Transmit FIFO Overflow Interrupt Enable When this field is set, the TXOF flag generates an interrupt to the host. 0 TXOF flag does not generate an interrupt to the host. 1 TXOF flag generates an interrupt to the host.
8 RXUFE	Receive FIFO Underflow Interrupt Enable When this field is set, the RXUF flag generates an interrupt to the host. 0 RXUF flag does not generate an interrupt to the host. 1 RXUF flag generates an interrupt to the host.
7 TXFE	Transmit FIFO Enable 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 CTRL[TE] and CTRL[RE] must be cleared prior to changing this field. 0 Transmit FIFO is not enabled. Buffer is depth 1. (Legacy support). 1 Transmit FIFO is enabled. Buffer is depth indicated by TXFIFOSIZE.
6–4 TXFIFOSIZE	Transmit FIFO. Buffer Depth The maximum number of transmit datawords that can be stored in the transmit buffer. This field is read only. 000 Transmit FIFO/Buffer depth = 1 dataword. 001 Transmit FIFO/Buffer depth = 4 datawords. 010 Transmit FIFO/Buffer depth = 8 datawords. 011 Transmit FIFO/Buffer depth = 16 datawords. 100 Transmit FIFO/Buffer depth = 32 datawords. 101 Transmit FIFO/Buffer depth = 64 datawords. 110 Transmit FIFO/Buffer depth = 128 datawords. 111 Transmit FIFO/Buffer depth = 256 datawords
3 RXFE	Receive FIFO Enable 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 CTRL[TE] and CTRL[RE] must be cleared prior to changing this field. 0 Receive FIFO is not enabled. Buffer is depth 1. (Legacy support) 1 Receive FIFO is enabled. Buffer is depth indicted by RXFIFOSIZE.
RXFIFOSIZE	Receive FIFO. Buffer Depth The maximum number of receive datawords that can be stored in the receive buffer before an overrun occurs. This field is read only. 000 Receive FIFO/Buffer depth = 1 dataword.

Table continues on the next page...

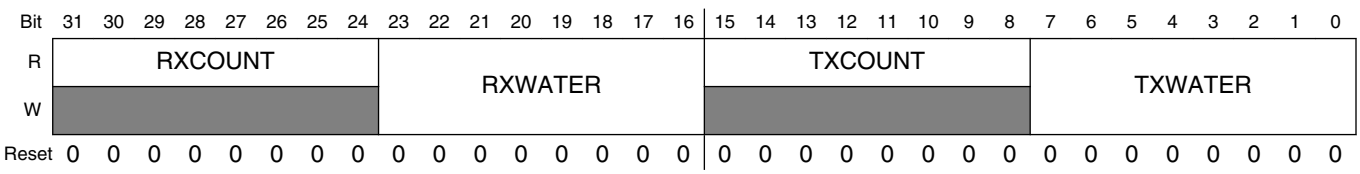
LPUARTx_FIFO field descriptions (continued)

Field	Description
	001 Receive FIFO/Buffer depth = 4 datawords. 010 Receive FIFO/Buffer depth = 8 datawords. 011 Receive FIFO/Buffer depth = 16 datawords. 100 Receive FIFO/Buffer depth = 32 datawords. 101 Receive FIFO/Buffer depth = 64 datawords. 110 Receive FIFO/Buffer depth = 128 datawords. 111 Receive FIFO/Buffer depth = 256 datawords.

50.3.8 LPUART Watermark Register (LPUARTx_WATER)

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 CTRL[TE] is not set.

Address: Base address + 1Ch offset



LPUARTx_WATER field descriptions

Field	Description
31–24 RXCOUNT	Receive Counter 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 FIFO[RXFIFOSIZE] to calculate how much room is left in the receive FIFO/buffer.
23–16 RXWATER	Receive Watermark When the number of datawords in the receive FIFO/buffer is greater than the value in this register field, an interrupt or a DMA request is generated. For proper operation, the value in RXWATER must be set to be less than the receive FIFO/buffer size as indicated by FIFO[RXFIFOSIZE] and FIFO[RXFE] and must be greater than 0.
15–8 TXCOUNT	Transmit Counter 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 FIFO[TXFIFOSIZE] to calculate how much room is left in the transmit FIFO/buffer.
TXWATER	Transmit Watermark When the number of datawords in the transmit FIFO/buffer is equal to or less than the value in this register field, an interrupt or a DMA request is generated. 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 FIFO[TXFIFOSIZE] and FIFO[TXFE].

LPUARTx_WATER field descriptions (continued)

Field	Description

50.4 Functional description

The LPUART supports full-duplex, asynchronous, NRZ serial communication and comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. The following describes each of the blocks of the LPUART.

50.4.1 Baud rate generation

A 13-bit modulus 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 baud clock divisor for the asynchronous LPUART baud clock. The SBR bits are in the LPUART baud rate registers, BDH and BDL. The baud rate clock drives the receiver, while the transmitter is driven by the baud rate clock divided by the over sampling ratio. Depending on the over sampling ratio, the receiver has an acquisition rate of 4 to 32 samples per bit time.

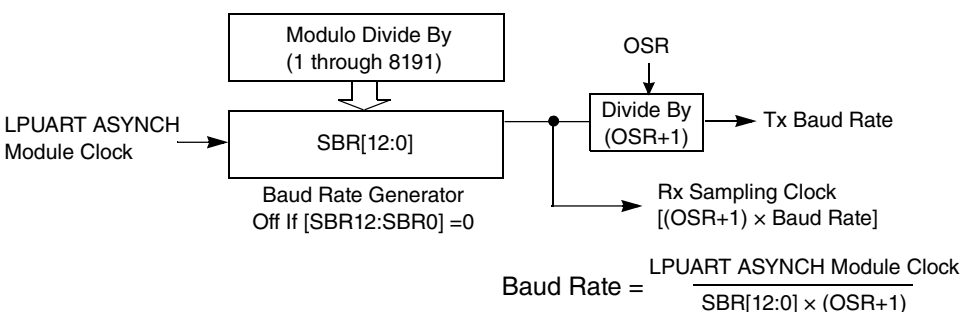


Figure 50-3. LPUART baud rate generation

Baud rate generation is subject to two sources of error:

- Integer division of the asynchronous LPUART baud clock may not give the exact target frequency.
- Synchronization with the asynchronous LPUART baud clock can cause phase shift.

50.4.2 Transmitter functional description

This section describes the overall block diagram for the LPUART transmitter, as well as specialized functions for sending break and idle characters.

The transmitter output (LPUART_TX) idle state defaults to logic high, CTRL[TXINV] is cleared following reset. The transmitter output is inverted by setting CTRL[TXINV]. The transmitter is enabled by setting the CTRL[TE] bit. This queues a preamble character that is one full character frame of the idle state. The transmitter then remains idle until data is available in the transmit data buffer. Programs store data into the transmit data buffer by writing to the LPUART data register.

The central element of the LPUART transmitter is the transmit shift register that is 10-bit to 13 bits long depending on the setting in the CTRL[M], BAUD[M10] and BAUD[SBNS] control bits. For the remainder of this section, assume CTRL[M], BAUD[M10] and BAUD[SBNS] are cleared, selecting the normal 8-bit data mode. In 8-bit data mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new character, the value waiting in the transmit data register is transferred to the shift register, synchronized with the baud rate clock, and the transmit data register empty (STAT[TDRE]) status flag is set to indicate another character may be written to the transmit data buffer at LPUART_DATA.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the LPUART_TX pin, the transmitter sets the transmit complete flag and enters an idle mode, with LPUART_TX high, waiting for more characters to transmit.

Writing 0 to CTRL[TE] does not immediately disable the transmitter. The current transmit activity in progress must first be completed (that could include a data character, idle character or break character), although the transmitter will not start transmitting another character.

50.4.2.1 Send break and queued idle

The LPUART_CTRL[SBK] bit sends break characters originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0, 10-bit to 12-bit times including the start and stop bits. A longer break of 13-bit times can be enabled by setting LPUART_STAT[BRK13]. Normally, a program would wait for LPUART_STAT[TDRE] to become set to indicate the last character of a message has moved to the transmit shifter, write 1, and then write 0 to the LPUART_CTRL[SBK] bit. This action queues a break character to be sent as soon as the shifter is available. If LPUART_CTRL[SBK] remains 1 when the queued break moves into the shifter,

synchronized to the baud rate clock, an additional break character is queued. If the receiving device is another LPUART, the break characters are received as 0s in all data bits and a framing error (LPUART_STAT[FE] = 1) occurs.

A break character can also be transmitted by writing to the LPUART_DATA register with bit 13 set and the data bits clear. This supports transmitting the break character as part of the normal data stream and also allows the DMA to transmit a break character.

When idle-line wakeup is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program would wait for LPUART_STAT[TDRE] to become set to indicate the last character of a message has moved to the transmit shifter, then write 0 and then write 1 to the LPUART_CTRL[TE] bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while LPUART_CTRL[TE] is cleared, the LPUART transmitter never actually releases control of the LPUART_TX pin.

An idle character can also be transmitted by writing to the LPUART_DATA register with bit 13 set and the data bits also set. This supports transmitting the idle character as part of the normal data stream and also allows the DMA to transmit a break character.

The length of the break character is affected by the LPUART_STAT[BRK13], LPUART_CTRL[M], LPUART_BAUD[M10] and LPUART_BAUD[SNBS] bits as shown below.

Table 50-2. Break character length

BRK13	M	M10	SBNS	Break character length
0	0	0	0	10 bit times
0	0	0	1	11 bit times
0	1	0	0	11 bit times
0	1	0	1	12 bit times
0	X	1	0	12 bit times
0	X	1	1	13 bit times
1	0	0	0	13 bit times
1	0	0	1	13 bit times
1	1	0	0	14 bit times
1	1	0	1	14 bit times
1	X	1	0	15 bit times
1	X	1	1	15 bit times

50.4.2.2 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 LPUART_TX remains in the mark state until CTS is reasserted.

If the clear-to-send operation is disabled, the transmitter ignores the state of CTS.

The transmitter's CTS signal can also be enabled even if the same LPUART receiver's RTS signal is disabled.

50.4.2.3 Transceiver driver enable

The transmitter can use LPUART_RTS as an enable signal for the driver of an external transceiver. See Transceiver driver enable using LPUART_RTS for details. If the request-to-send operation is enabled, when a character is placed into an empty transmitter data buffer, LPUART_RTS asserts one bit time before the start bit is transmitted. LPUART_RTS remains asserted for the whole time that the transmitter data buffer has any characters. LPUART_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 LPUART_RTS, with the same assertion and deassertion timing as having a character in the transmitter data buffer.

The transmitter's LPUART_RTS signal asserts only when the transmitter is enabled. However, the transmitter's LPUART_RTS signal is unaffected by its LPUART_CTS signal. LPUART_RTS will remain asserted until the transfer is completed, even if the transmitter is disabled mid-way through a data transfer.

50.4.3 Receiver functional description

In this section, the receiver block diagram is a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, different variations of the receiver wakeup function are explained.

The receiver input is inverted by setting LPUART_STAT[RXINV]. The receiver is enabled by setting the LPUART_CTRL[RE] bit. Character frames consist of a start bit of logic 0, eight to ten data bits (msb or lsb first), and one or two stop bits of logic 1. For

information about 9-bit or 10-bit data mode, refer to 8-bit, 9-bit and 10-bit data modes. For the remainder of this discussion, assume the LPUART is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (LPUART_STAT[RDRF]) status flag is set. If LPUART_STAT[RDRF] was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the LPUART receiver is double-buffered, the program has one full character time after LPUART_STAT[RDRF] is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full (LPUART_STAT[RDRF] = 1), it gets the data from the receive data register by reading LPUART_DATA. Refer to Interrupts and status flags for details about flag clearing.

50.4.3.1 Data sampling technique

The LPUART receiver supports a configurable oversampling rate of between 4× and 32× of the baud rate clock for sampling. The receiver starts by taking logic level samples at the oversampling rate times the baud rate to search for a falling edge on the LPUART_RX serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The oversampling baud rate clock divides the bit time into 4 to 32 segments from 1 to OSR (where OSR is the configured oversampling ratio). When a falling edge is located, three more samples are taken at (OSR/2), (OSR/2)+1, and (OSR/2)+2 to make sure this was a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes it is synchronized to a received character. If another falling edge is detected before the receiver is considered synchronized, the receiver restarts the sampling from the first segment.

The receiver then samples each bit time, including the start and stop bits, at (OSR/2), (OSR/2)+1, and (OSR/2)+2 to determine the logic level for that bit. The logic level is interpreted to be that of the majority of the samples taken during the bit time. If any sample in any bit time, including the start and stop bits, in a character frame fails to agree with the logic level for that bit, the noise flag (LPUART_STAT[NF]) is set when the received character is transferred to the receive data buffer.

When the LPUART receiver is configured to sample on both edges of the baud rate clock, the number of segments in each received bit is effectively doubled (from 1 to OSR×2). The start and data bits are then sampled at OSR, OSR+1 and OSR+2. Sampling on both edges of the clock must be enabled for oversampling rates of 4× to 7× and is optional for higher oversampling rates.

The falling edge detection logic continuously looks for falling edges. If an edge is detected, the sample clock is resynchronized to bit times (unless resynchronization has been disabled). This improves the reliability of the receiver in the presence of noise or mismatched baud rates. It does not improve worst case analysis because some characters do not have any extra falling edges anywhere in the character frame.

In the case of a framing error, provided the received character was not a break character, the sampling logic that searches for a falling edge is filled with three logic 1 samples so that a new start bit can be detected almost immediately.

50.4.3.2 Receiver wakeup operation

Receiver wakeup and receiver address matching is a hardware mechanism that allows an LPUART receiver to ignore the characters in a message intended for a different receiver.

During receiver wakeup, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up control bit (LPUART_CTRL[RWU]). When RWU bit and LPUART_S2[RWUID] bit are set, the status flags associated with the receiver, with the exception of the idle bit, IDLE, are inhibited from setting, thus eliminating the software overhead for handling the unimportant message characters. At the end of a message, or at the beginning of the next message, all receivers automatically force LPUART_CTRL[RWU] to 0 so all receivers wake up in time to look at the first character(s) of the next message.

During receiver address matching, the address matching is performed in hardware and the LPUART receiver will ignore all characters that do not meet the address match requirements.

Table 50-3. Receiver Wakeup Options

RWU	MA1 | MA2	MATCFG	WAKE:RWUID	Receiver Wakeup
0	0	X	X	Normal operation
1	0	00	00	Receiver wakeup on idle line, IDLE flag not set
1	0	00	01	Receiver wakeup on idle line, IDLE flag set

Table continues on the next page...

Table 50-3. Receiver Wakeup Options (continued)

RWU	MA1 | MA2	MATCFG	WAKE:RWUID	Receiver Wakeup
1	0	00	10	Receiver wakeup on address mark
1	1	11	X0	Receiver wakeup on data match
0	1	00	X0	Address mark address match, IDLE flag not set for discarded characters
0	1	00	X1	Address mark address match, IDLE flag set for discarded characters
0	1	01	X0	Idle line address match
0	1	10	X0	Address match on and address match off, IDLE flag not set for discarded characters
0	1	10	X1	Address match on and address match off, IDLE flag set for discarded characters

50.4.3.2.1 Idle-line wakeup

When wake is cleared, the receiver is configured for idle-line wakeup. In this mode, LPUART_CTRL[RWU] is cleared automatically when the receiver detects a full character time of the idle-line level. The LPUART_CTRL[M] and LPUART_BAUD[M10] control bit selects 8-bit to 10-bit data mode and the LPUART_BAUD[SBNS] bit selects 1-bit or 2-bit stop bit number that determines how many bit times of idle are needed to constitute a full character time, 10 to 13 bit times because of the start and stop bits.

When LPUART_CTRL[RWU] is one and LPUART_STAT[RWUID] is zero, the idle condition that wakes up the receiver does not set the LPUART_STAT[IDLE] flag. The receiver wakes up and waits for the first data character of the next message that sets the LPUART_STAT[RDRF] flag and generates an interrupt if enabled. When LPUART_STAT[RWUID] is one, any idle condition sets the LPUART_STAT[IDLE] flag and generates an interrupt if enabled, regardless of whether LPUART_CTRL[RWU] is zero or one.

The idle-line type (LPUART_CTRL[ILT]) control bit selects one of two ways to detect an idle line. When LPUART_CTRL[ILT] is cleared, the idle bit counter starts after the start bit so the stop bit and any logic 1s at the end of a character count toward the full

character time of idle. When LPUART_CTRL[ILT] is set, the idle bit counter does not start until after the stop bit time, so the idle detection is not affected by the data in the last character of the previous message.

50.4.3.2.2 Address-mark wakeup

When LPUART_CTRL[WAKE] is set, the receiver is configured for address-mark wakeup. In this mode, LPUART_CTRL[RWU] is cleared automatically when the receiver detects a logic 1 in the most significant bit of a received character.

Address-mark wakeup allows messages to contain idle characters, but requires the MSB be reserved for use in address frames. The logic 1 in the MSB of an address frame clears the LPUART_CTRL[RWU] bit before the stop bits are received and sets the LPUART_STAT[RDRF] flag. In this case, the character with the MSB set is received even though the receiver was sleeping during most of this character time.

50.4.3.2.3 Data match wakeup

When LPUART_CTRL[RWU] is set and LPUART_BAUD[MATCFG] equals 11, the receiver is configured for data match wakeup. In this mode, LPUART_CTRL[RWU] is cleared automatically when the receiver detects a character that matches MATCH[MA1] field when BAUD[MAEN1] is set, or that matches MATCH[MA2] when BAUD[MAEN2] is set.

50.4.3.2.4 Address Match operation

Address match operation is enabled when the LPUART_BAUD[MAEN1] or LPUART_BAUD[MAEN2] bit is set and LPUART_BAUD[MATCFG] is equal to 00. In this function, a character received by the LPUART_RX pin with a logic 1 in the bit position immediately preceding the stop bit is considered an address and is compared with the associated MATCH[MA1] or MATCH[MA2] field. The character is only transferred to the receive buffer, and LPUART_STAT[RDRF] is set, if the comparison matches. All subsequent characters received with a logic 0 in the bit position immediately preceding the stop bit 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 characters with logic zero in the bit position immediately preceding the stop bit are also discarded. If both the LPUART_BAUD[MAEN1] and LPUART_BAUD[MAEN2] bits are negated, the receiver operates normally and all data received is transferred to the receive data buffer.

Address match operation functions in the same way for both MATCH[MA1] and MATCH[MA2] fields.

- If only one of LPUART_BAUD[MAEN1] and LPUART_BAUD[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 LPUART_BAUD[MAEN1] and LPUART_BAUD[MAEN2] are asserted, a marked address is compared with both match registers and data is transferred only on a match with either register.

50.4.3.2.5 Idle Match operation

Idle match operation is enabled when the LPUART_BAUD[MAEN1] or LPUART_BAUD[MAEN2] bit is set and LPUART_BAUD[MATCFG] is equal to 01. In this function, the first character received by the LPUART_RX pin after an idle line condition is considered an address and is compared with the associated MA1 or MA2 register. The character is only transferred to the receive buffer, and LPUART_STAT[RDRF] is set, if the comparison matches. All subsequent characters are considered to be data associated with the address and are transferred to the receive data buffer until the next idle line condition is detected. If no address match occurs then no transfer is made to the receive data buffer, and all following frames until the next idle condition are also discarded. If both the LPUART_BAUD[MAEN1] and LPUART_BAUD[MAEN2] bits are negated, the receiver operates normally and all data received is transferred to the receive data buffer.

Idle match operation functions in the same way for both MA1 and MA2 registers.

- If only one of LPUART_BAUD[MAEN1] and LPUART_BAUD[MAEN2] is asserted, the first character after an idle line is compared only with the associated match register and data is transferred to the receive data buffer only on a match.
- If LPUART_BAUD[MAEN1] and LPUART_BAUD[MAEN2] are asserted, the first character after an idle line is compared with both match registers and data is transferred only on a match with either register.

50.4.3.2.6 Match On Match Off operation

Match on, match off operation is enabled when both LPUART_BAUD[MAEN1] and LPUART_BAUD[MAEN2] are set and LPUART_BAUD[MATCFG] is equal to 10. In this function, a character received by the LPUART_RX pin that matches MATCH[MA1] is received and transferred to the receive buffer, and LPUART_STAT[RDRF] is set. All subsequent characters are considered to be data and are also transferred to the receive data buffer, until a character is received that matches MATCH[MA2] register. The character that matches MATCH[MA2] and all following characters are discarded, this

continues until another character that matches MATCH[MA1] is received. If both the LPUART_BAUD[MAEN1] and LPUART_BAUD[MAEN2] bits are negated, the receiver operates normally and all data received is transferred to the receive data buffer.

NOTE

Match on, match off operation requires both LPUART_BAUD[MAEN1] and LPUART_BAUD[MAEN2] to be asserted.

50.4.3.3 Hardware flow control

To support hardware flow control, the receiver can be programmed to automatically deassert and assert LPUART_RTS.

- LPUART_RTS remains asserted until the transfer is complete, even if the transmitter is disabled midway through a data transfer. See Transceiver driver enable using LPUART_RTS for more details.
- If the receiver request-to-send functionality is enabled, the receiver automatically deasserts LPUART_RTS if the number of characters in the receiver data register is full or a start bit is detected that will cause the receiver data register to be full.
- The receiver asserts LPUART_RTS when the number of characters in the receiver data register is not full and has not detected a start bit that will cause the receiver data register to be full. It is not affected if STAT[RDRF] is asserted.
- Even if LPUART_RTS is deasserted, the receiver continues to receive characters until the receiver data buffer is overrun.
- If the receiver request-to-send functionality is disabled, the receiver LPUART_RTS remains deasserted.

50.4.3.4 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 OSR oversampling baud rate clock counter that filters noise and indicates when a 1 is received.

50.4.3.4.1 Start bit detection

When STAT[RXINV] is cleared, the first falling edge of the received character corresponds to the start bit. The infrared decoder resets its counter. At this time, the receiver also begins its start bit detection process. After the start bit is detected, the

receiver synchronizes its bit times to this start bit time. For the rest of the character reception, the infrared decoder's counter and the receiver's bit time counter count independently from each other.

50.4.3.4.2 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 oversampling baud clock can be undetected by it regardless of whether it is seen in the first or second half of the count.

50.4.3.4.3 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.

50.4.3.4.4 High-bit detection

At OSR oversampling baud rate 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.

50.4.4 Additional LPUART functions

The following sections describe additional LPUART functions.

50.4.4.1 8-bit, 9-bit and 10-bit data modes

The LPUART transmitter and receiver can be configured to operate in 9-bit data mode by setting the LPUART_CTRL[M] or 10-bit data mode by setting LPUART_CTRL[M10]. In 9-bit mode, there is a ninth data bit in 10-bit mode there is a tenth data bit. For the transmit data buffer, these bits are stored in LPUART_CTRL[T8] and LPUART_CTRL[T9]. For the receiver, these bits are held in LPUART_CTRL[R8] and LPUART_CTRL[R9]. They are also accessible via 16-bit or 32-bit accesses to the LPUART_DATA register.

For coherent 8-bit writes to the transmit data buffer, write to LPUART_CTRL[T8] and LPUART_CTRL[T9] before writing to LPUART_DATA[7:0]. For 16-bit and 32-bit writes to the LPUART_DATA register all 10 transmit bits are written to the transmit data buffer at the same time.

If the bit values to be transmitted as the ninth and tenth bit of a new character are the same as for the previous character, it is not necessary to write to LPUART_CTRL[T8] and LPUART_CTRL[T9] again. When data is transferred from the transmit data buffer to the transmit shifter, the value in LPUART_CTRL[T8] and LPUART_CTRL[T9] is copied at the same time data is transferred from LPUART_DATA[7:0] to the shifter.

The 9-bit data mode is typically used with parity to allow eight bits of data plus the parity in the ninth bit, or it is used with address-mark wakeup so the ninth data bit can serve as the wakeup bit. The 10-bit data mode is typically used with parity and address-mark wakeup so the ninth data bit can serve as the wakeup bit and the tenth bit as the parity bit. In custom protocols, the ninth and/or tenth bits can also serve as software-controlled markers.

50.4.4.2 Idle length

An idle character is a character where the start bit, all data bits and stop bits are in the mark postion. The CTRL[ILT] register can be configured to start detecting an idle character from the previous start bit (any data bits and stop bits count towards the idle character detection) or from the previous stop bit.

The number of idle characters that must be received before an idle line condition is detected can also be configured using the CTRL[IDLECFG] field. This field configures the number of idle characters that must be received before the STAT[IDLE] flag is set, the STAT[RAF] flag is cleared and the DATA[IDLINE] flag is set with the next received character.

Idle-line wakeup and idle match operation are also affected by the CTRL[IDLECFG] field. When address match or match on/off operation is enabled, setting the STAT[RWUID] bit will cause any discarded characters to be treated as if they were idle characters.

50.4.4.3 Loop mode

When LPUART_CTRL[LOOPS] is set, the LPUART_CTRL[RSRC] bit in the same register chooses between loop mode (LPUART_CTRL[RSRC] = 0) or single-wire mode (LPUART_CTRL[RSRC] = 1). Loop mode is sometimes used to check software,

independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is internally connected to the receiver input and the LPUART_RX pin is not used by the LPUART.

50.4.4.4 Single-wire operation

When LPUART_CTRL[LOOPS] is set, the RSRC bit in the same register chooses between loop mode (LPUART_CTRL[RSRC] = 0) or single-wire mode (LPUART_CTRL[RSRC] = 1). Single-wire mode implements a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the LPUART_TX pin (the LPUART_RX pin is not used).

In single-wire mode, the LPUART_CTRL[TXDIR] bit controls the direction of serial data on the LPUART_TX pin. When LPUART_CTRL[TXDIR] is cleared, the LPUART_TX pin is an input to the receiver and the transmitter is temporarily disconnected from the LPUART_TX pin so an external device can send serial data to the receiver. When LPUART_CTRL[TXDIR] is set, the LPUART_TX pin is an output driven by the transmitter, the internal loop back connection is disabled, and as a result the receiver cannot receive characters that are sent out by the transmitter.

50.4.5 Infrared interface

The LPUART 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 LPUART. 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 LPUART has an infrared transmit encoder and receive decoder. The LPUART 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 LPUART. The narrow pulses are then stretched by the infrared receive decoder to get back to a serial bit stream to be received by the LPUART. 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 high pulses.

The infrared submodule receives its clock sources from the LPUART. One of these two clocks are selected in the infrared submodule to generate either 1/OSR, 2/OSR, 3/OSR, or 4/OSR narrow pulses during transmission.

50.4.5.1 Infrared transmit encoder

The infrared transmit encoder converts serial bits of data from transmit shift register to the LPUART_TX signal. A narrow pulse is transmitted for a zero bit and no pulse for a one bit. The narrow pulse is sent at the start of the bit with a duration of 1/OSR, 2/OSR, 3/OSR, or 4/OSR of a bit time. A narrow low pulse is transmitted for a zero bit when LPUART_CTRL[TXINV] is cleared, while a narrow high pulse is transmitted for a zero bit when LPUART_CTRL[TXINV] is set.

50.4.5.2 Infrared receive decoder

The infrared receive block converts data from the LPUART_RX 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 low pulse is expected for a zero bit when LPUART_STAT[RXINV] is cleared, while a narrow high pulse is expected for a zero bit when LPUART_STAT[RXINV] is set. This receive decoder meets the edge jitter requirement as defined by the IrDA serial infrared physical layer specification.

50.4.6 Interrupts and status flags

The LPUART transmitter has two status flags that can optionally generate hardware interrupt requests. Transmit data register empty LPUART_STAT[TDRE]) indicates when there is room in the transmit data buffer to write another transmit character to LPUART_DATA. If the transmit interrupt enable LPUART_CTRL[TIE]) bit is set, a hardware interrupt is requested when LPUART_STAT[TDRE] is set. Transmit complete (LPUART_STAT[TC]) indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with LPUART_TX at the inactive level. This flag is often used in systems with modems to determine when it is safe to turn off the modem. If the transmit complete interrupt enable (LPUART_CTRL[TCIE]) bit is set, a hardware interrupt is requested when LPUART_STAT[TC] is set. Instead of hardware interrupts, software polling may be used to monitor the LPUART_STAT[TDRE] and LPUART_STAT[TC] status flags if the corresponding LPUART_CTRL[TIE] or LPUART_CTRL[TCIE] local interrupt masks are cleared.

When a program detects that the receive data register is full (LPUART_STAT[RDRF] = 1), it gets the data from the receive data register by reading LPUART_DATA. The LPUART_STAT[RDRF] flag is cleared by reading LPUART_DATA.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the LPUART_RX line remains idle for an extended period of time. IDLE is cleared by writing 1 to the LPUART_STAT[IDLE] flag. After LPUART_STAT[IDLE] has been cleared, it cannot become set again until the receiver has received at least one new character and has set LPUART_STAT[RDRF].

If the associated error was detected in the received character that caused LPUART_STAT[RDRF] to be set, the error flags - noise flag (LPUART_STAT[NF]), framing error (LPUART_STAT[FE]), and parity error flag (LPUART_STAT[PF]) - are set at the same time as LPUART_STAT[RDRF]. These flags are not set in overrun cases.

If LPUART_STAT[RDRF] was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (LPUART_STAT[OR]) flag is set instead of the data along with any associated NF, FE, or PF condition is lost.

If the received character matches the contents of MATCH[MA1] and/or MATCH[MA2] then the LPUART_STAT[MA1F] and/or LPUART_STAT[MA2F] flags are set at the same time that LPUART_STAT[RDRF] is set.

At any time, an active edge on the LPUART_RX serial data input pin causes the LPUART_STAT[RXEDGIF] flag to set. The LPUART_STAT[RXEDGIF] flag is cleared by writing a 1 to it. This function depends on the receiver being enabled (LPUART_CTRL[RE] = 1).

Chapter 51
FlexIO

51.1 Chip-specific FlexIO information

51.1.1 FlexIO Instantiation

This section summarize the features and module configurations of FlexIO.

This device contain one FlexIO modules with 32 pins, 8 counters and 8 shifters.

Amongst other features, the FlexIO module in this device supports:

1. Emulation in serial/parallel communication protocols
2. Programmable logic blocks allowing the implementation of digital logic functions on-chip and configurable interaction of internal and external modules

51.1.2 FlexIO trigger options

FlexIO has a selectable trigger input source controlled by FlexIO_TIMCTLn[TRGSEL] (4-bit field) to use for starting the counter and/or reloading the counter. The options available are shown in the following table:

Table 51-1. FlexIO trigger Options

FlexIO_TIMCTLn[TRGSEL] (4-bit field)	Selected source
0000	External trigger pin input (EXTRG_IN)
0001	CMP0 output
0010	Reserved
0011	Reserved
0100	PIT trigger 0
0101	PIT trigger 1
0110	PIT trigger 2

Table continues on the next page...

Table 51-1. FlexIO trigger Options (continued)

FlexIO_TIMCTLn[TRGSEL] (4-bit field)	Selected source
0111	PIT trigger 3
1000	TPM0 overflow
1001	TPM1 trigger
1010	TPM2 trigger
1011	LPTMR1 trigger
1100	RTC alarm
1101	RTC seconds
1110	LPTMR0 trigger
1111	Reserved

51.2 Introduction

51.2.1 Overview

The FlexIO is a highly configurable module providing a wide range of functionality including:

- Emulation of a variety of serial/parallel communication protocols
- Flexible 16-bit timers with support for a variety of trigger, reset, enable and disable conditions
- Programmable logic blocks allowing the implementation of digital logic functions on-chip and configurable interaction of internal and external modules
- Programmable state machine for offloading basic system control functions from CPU

These functions are provided by the FlexIO while adhering to the following key objectives:

- Low software/CPU overhead: less overhead than software bit-banging, more overhead than dedicated peripheral IP.
- Area/Power efficient implementation: more efficient than integrating multiple peripherals for each desired protocol or adding external board-level discrete logic.

51.2.2 Features

The FlexIO module is capable of supporting a wide range of protocols including, but not limited to:

- UART
- I2C
- SPI
- I2S
- Camera IF
- Motorola 68K/Intel 8080 bus
- PWM/Waveform generation

The following key features are provided:

- Array of 32-bit shift registers with transmit, receive and data match modes
- Double buffered shifter operation for continuous data transfer
- Shifter concatenation to support large transfer sizes
- Automatic start/stop bit generation
- 1, 2, 4, 8, 16 or 32 multi-bit shift widths for parallel interface support
- Interrupt, DMA or polled transmit/receive operation
- Programmable baud rates independent of bus clock frequency, with support for asynchronous operation during stop modes
- Highly flexible 16-bit timers with support for a variety of internal or external trigger, reset, enable and disable conditions
- Programmable logic mode for integrating external digital logic functions on-chip or combining pin/shifter/timer functions to generate complex outputs
- Programmable state machine for offloading basic system control functions from CPU with support for up to 8 states, 8 outputs and 3 selectable inputs per state

51.2.3 Block Diagram

The following diagram gives a high-level overview of the configuration of FlexIO timers and shifters.

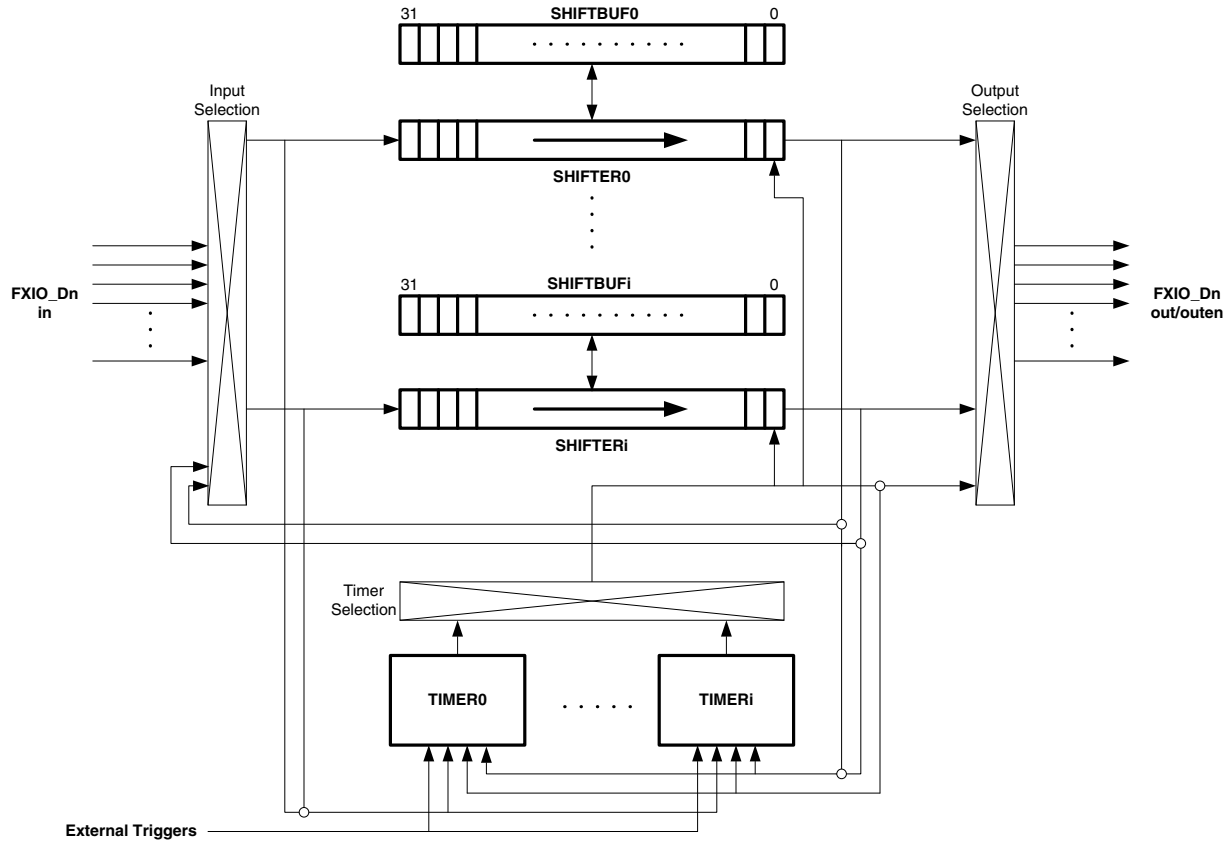


Figure 51-1. FlexIO block diagram

51.2.4 Modes of operation

The FlexIO module supports the chip modes described in the following table.

Table 51-2. Chip modes supported by the FlexIO module

Chip mode	FlexIO Operation
Run	Normal operation
Stop/Wait	Can continue operating provided the Doze Enable bit (CTRL[DOZEN]) is set and the FlexIO is using an external or internal clock source which remains operating during stop/wait modes.
Low Leakage Stop	The Doze Enable (CTRL[DOZEN]) bit is ignored and the FlexIO will wait for all Timers to complete any pending operation before acknowledging low leakage mode entry.
Debug	Can continue operating provided the Debug Enable bit (CTRL[DBGE]) is set.

51.2.5 FlexIO Signal Descriptions

Signal	Description	I/O
FXIO_Dn (n=0...31)	Bidirectional FlexIO Shifter and Timer pin inputs/outputs	I/O

51.3 Memory Map and Registers

FLEXIO memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4005_F000	Version ID Register (FLEXIO0_VERID)	32	R	0101_0001h	51.3.1/1524
4005_F004	Parameter Register (FLEXIO0_PARAM)	32	R	See section	51.3.2/1524
4005_F008	FlexIO Control Register (FLEXIO0_CTRL)	32	R/W	0000_0000h	51.3.3/1525
4005_F00C	Pin State Register (FLEXIO0_PIN)	32	R	0000_0000h	51.3.4/1526
4005_F010	Shifter Status Register (FLEXIO0_SHIFTSTAT)	32	w1c	0000_0000h	51.3.5/1527
4005_F014	Shifter Error Register (FLEXIO0_SHIFTERR)	32	w1c	0000_0000h	51.3.6/1527
4005_F018	Timer Status Register (FLEXIO0_TIMSTAT)	32	w1c	0000_0000h	51.3.7/1528
4005_F020	Shifter Status Interrupt Enable (FLEXIO0_SHIFTSIEN)	32	R/W	0000_0000h	51.3.8/1529
4005_F024	Shifter Error Interrupt Enable (FLEXIO0_SHIFTEIEN)	32	R/W	0000_0000h	51.3.9/1529
4005_F028	Timer Interrupt Enable Register (FLEXIO0_TIMIEN)	32	R/W	0000_0000h	51.3.10/ 1530
4005_F030	Shifter Status DMA Enable (FLEXIO0_SHIFTSDEN)	32	R/W	0000_0000h	51.3.11/ 1530
4005_F040	Shifter State Register (FLEXIO0_SHIFTSTATE)	32	R/W	0000_0000h	51.3.12/ 1531
4005_F080	Shifter Control N Register (FLEXIO0_SHIFTCTL0)	32	R/W	0000_0000h	51.3.13/ 1532
4005_F084	Shifter Control N Register (FLEXIO0_SHIFTCTL1)	32	R/W	0000_0000h	51.3.13/ 1532
4005_F088	Shifter Control N Register (FLEXIO0_SHIFTCTL2)	32	R/W	0000_0000h	51.3.13/ 1532
4005_F08C	Shifter Control N Register (FLEXIO0_SHIFTCTL3)	32	R/W	0000_0000h	51.3.13/ 1532
4005_F090	Shifter Control N Register (FLEXIO0_SHIFTCTL4)	32	R/W	0000_0000h	51.3.13/ 1532
4005_F094	Shifter Control N Register (FLEXIO0_SHIFTCTL5)	32	R/W	0000_0000h	51.3.13/ 1532
4005_F098	Shifter Control N Register (FLEXIO0_SHIFTCTL6)	32	R/W	0000_0000h	51.3.13/ 1532
4005_F09C	Shifter Control N Register (FLEXIO0_SHIFTCTL7)	32	R/W	0000_0000h	51.3.13/ 1532

Table continues on the next page...

FLEXIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4005_F100	Shifter Configuration N Register (FLEXIO0_SHIFTCFG0)	32	R/W	0000_0000h	51.3.14/ 1533
4005_F104	Shifter Configuration N Register (FLEXIO0_SHIFTCFG1)	32	R/W	0000_0000h	51.3.14/ 1533
4005_F108	Shifter Configuration N Register (FLEXIO0_SHIFTCFG2)	32	R/W	0000_0000h	51.3.14/ 1533
4005_F10C	Shifter Configuration N Register (FLEXIO0_SHIFTCFG3)	32	R/W	0000_0000h	51.3.14/ 1533
4005_F110	Shifter Configuration N Register (FLEXIO0_SHIFTCFG4)	32	R/W	0000_0000h	51.3.14/ 1533
4005_F114	Shifter Configuration N Register (FLEXIO0_SHIFTCFG5)	32	R/W	0000_0000h	51.3.14/ 1533
4005_F118	Shifter Configuration N Register (FLEXIO0_SHIFTCFG6)	32	R/W	0000_0000h	51.3.14/ 1533
4005_F11C	Shifter Configuration N Register (FLEXIO0_SHIFTCFG7)	32	R/W	0000_0000h	51.3.14/ 1533
4005_F200	Shifter Buffer N Register (FLEXIO0_SHIFTBUF0)	32	R/W	0000_0000h	51.3.15/ 1535
4005_F204	Shifter Buffer N Register (FLEXIO0_SHIFTBUF1)	32	R/W	0000_0000h	51.3.15/ 1535
4005_F208	Shifter Buffer N Register (FLEXIO0_SHIFTBUF2)	32	R/W	0000_0000h	51.3.15/ 1535
4005_F20C	Shifter Buffer N Register (FLEXIO0_SHIFTBUF3)	32	R/W	0000_0000h	51.3.15/ 1535
4005_F210	Shifter Buffer N Register (FLEXIO0_SHIFTBUF4)	32	R/W	0000_0000h	51.3.15/ 1535
4005_F214	Shifter Buffer N Register (FLEXIO0_SHIFTBUF5)	32	R/W	0000_0000h	51.3.15/ 1535
4005_F218	Shifter Buffer N Register (FLEXIO0_SHIFTBUF6)	32	R/W	0000_0000h	51.3.15/ 1535
4005_F21C	Shifter Buffer N Register (FLEXIO0_SHIFTBUF7)	32	R/W	0000_0000h	51.3.15/ 1535
4005_F280	Shifter Buffer N Bit Swapped Register (FLEXIO0_SHIFTBUFBIS0)	32	R/W	0000_0000h	51.3.16/ 1535
4005_F284	Shifter Buffer N Bit Swapped Register (FLEXIO0_SHIFTBUFBIS1)	32	R/W	0000_0000h	51.3.16/ 1535
4005_F288	Shifter Buffer N Bit Swapped Register (FLEXIO0_SHIFTBUFBIS2)	32	R/W	0000_0000h	51.3.16/ 1535
4005_F28C	Shifter Buffer N Bit Swapped Register (FLEXIO0_SHIFTBUFBIS3)	32	R/W	0000_0000h	51.3.16/ 1535
4005_F290	Shifter Buffer N Bit Swapped Register (FLEXIO0_SHIFTBUFBIS4)	32	R/W	0000_0000h	51.3.16/ 1535
4005_F294	Shifter Buffer N Bit Swapped Register (FLEXIO0_SHIFTBUFBIS5)	32	R/W	0000_0000h	51.3.16/ 1535

Table continues on the next page...

FLEXIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4005_F298	Shifter Buffer N Bit Swapped Register (FLEXIO0_SHIFTBUFBIS6)	32	R/W	0000_0000h	51.3.16/ 1535
4005_F29C	Shifter Buffer N Bit Swapped Register (FLEXIO0_SHIFTBUFBIS7)	32	R/W	0000_0000h	51.3.16/ 1535
4005_F300	Shifter Buffer N Byte Swapped Register (FLEXIO0_SHIFTBUFBYS0)	32	R/W	0000_0000h	51.3.17/ 1536
4005_F304	Shifter Buffer N Byte Swapped Register (FLEXIO0_SHIFTBUFBYS1)	32	R/W	0000_0000h	51.3.17/ 1536
4005_F308	Shifter Buffer N Byte Swapped Register (FLEXIO0_SHIFTBUFBYS2)	32	R/W	0000_0000h	51.3.17/ 1536
4005_F30C	Shifter Buffer N Byte Swapped Register (FLEXIO0_SHIFTBUFBYS3)	32	R/W	0000_0000h	51.3.17/ 1536
4005_F310	Shifter Buffer N Byte Swapped Register (FLEXIO0_SHIFTBUFBYS4)	32	R/W	0000_0000h	51.3.17/ 1536
4005_F314	Shifter Buffer N Byte Swapped Register (FLEXIO0_SHIFTBUFBYS5)	32	R/W	0000_0000h	51.3.17/ 1536
4005_F318	Shifter Buffer N Byte Swapped Register (FLEXIO0_SHIFTBUFBYS6)	32	R/W	0000_0000h	51.3.17/ 1536
4005_F31C	Shifter Buffer N Byte Swapped Register (FLEXIO0_SHIFTBUFBYS7)	32	R/W	0000_0000h	51.3.17/ 1536
4005_F380	Shifter Buffer N Bit Byte Swapped Register (FLEXIO0_SHIFTBUFBBS0)	32	R/W	0000_0000h	51.3.18/ 1536
4005_F384	Shifter Buffer N Bit Byte Swapped Register (FLEXIO0_SHIFTBUFBBS1)	32	R/W	0000_0000h	51.3.18/ 1536
4005_F388	Shifter Buffer N Bit Byte Swapped Register (FLEXIO0_SHIFTBUFBBS2)	32	R/W	0000_0000h	51.3.18/ 1536
4005_F38C	Shifter Buffer N Bit Byte Swapped Register (FLEXIO0_SHIFTBUFBBS3)	32	R/W	0000_0000h	51.3.18/ 1536
4005_F390	Shifter Buffer N Bit Byte Swapped Register (FLEXIO0_SHIFTBUFBBS4)	32	R/W	0000_0000h	51.3.18/ 1536
4005_F394	Shifter Buffer N Bit Byte Swapped Register (FLEXIO0_SHIFTBUFBBS5)	32	R/W	0000_0000h	51.3.18/ 1536
4005_F398	Shifter Buffer N Bit Byte Swapped Register (FLEXIO0_SHIFTBUFBBS6)	32	R/W	0000_0000h	51.3.18/ 1536
4005_F39C	Shifter Buffer N Bit Byte Swapped Register (FLEXIO0_SHIFTBUFBBS7)	32	R/W	0000_0000h	51.3.18/ 1536
4005_F400	Timer Control N Register (FLEXIO0_TIMCTL0)	32	R/W	0000_0000h	51.3.19/ 1537
4005_F404	Timer Control N Register (FLEXIO0_TIMCTL1)	32	R/W	0000_0000h	51.3.19/ 1537
4005_F408	Timer Control N Register (FLEXIO0_TIMCTL2)	32	R/W	0000_0000h	51.3.19/ 1537
4005_F40C	Timer Control N Register (FLEXIO0_TIMCTL3)	32	R/W	0000_0000h	51.3.19/ 1537

Table continues on the next page...

FLEXIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4005_F410	Timer Control N Register (FLEXIO0_TIMCTL4)	32	R/W	0000_0000h	51.3.19/ 1537
4005_F414	Timer Control N Register (FLEXIO0_TIMCTL5)	32	R/W	0000_0000h	51.3.19/ 1537
4005_F418	Timer Control N Register (FLEXIO0_TIMCTL6)	32	R/W	0000_0000h	51.3.19/ 1537
4005_F41C	Timer Control N Register (FLEXIO0_TIMCTL7)	32	R/W	0000_0000h	51.3.19/ 1537
4005_F480	Timer Configuration N Register (FLEXIO0_TIMCFG0)	32	R/W	0000_0000h	51.3.20/ 1539
4005_F484	Timer Configuration N Register (FLEXIO0_TIMCFG1)	32	R/W	0000_0000h	51.3.20/ 1539
4005_F488	Timer Configuration N Register (FLEXIO0_TIMCFG2)	32	R/W	0000_0000h	51.3.20/ 1539
4005_F48C	Timer Configuration N Register (FLEXIO0_TIMCFG3)	32	R/W	0000_0000h	51.3.20/ 1539
4005_F490	Timer Configuration N Register (FLEXIO0_TIMCFG4)	32	R/W	0000_0000h	51.3.20/ 1539
4005_F494	Timer Configuration N Register (FLEXIO0_TIMCFG5)	32	R/W	0000_0000h	51.3.20/ 1539
4005_F498	Timer Configuration N Register (FLEXIO0_TIMCFG6)	32	R/W	0000_0000h	51.3.20/ 1539
4005_F49C	Timer Configuration N Register (FLEXIO0_TIMCFG7)	32	R/W	0000_0000h	51.3.20/ 1539
4005_F500	Timer Compare N Register (FLEXIO0_TIMCMP0)	32	R/W	0000_0000h	51.3.21/ 1541
4005_F504	Timer Compare N Register (FLEXIO0_TIMCMP1)	32	R/W	0000_0000h	51.3.21/ 1541
4005_F508	Timer Compare N Register (FLEXIO0_TIMCMP2)	32	R/W	0000_0000h	51.3.21/ 1541
4005_F50C	Timer Compare N Register (FLEXIO0_TIMCMP3)	32	R/W	0000_0000h	51.3.21/ 1541
4005_F510	Timer Compare N Register (FLEXIO0_TIMCMP4)	32	R/W	0000_0000h	51.3.21/ 1541
4005_F514	Timer Compare N Register (FLEXIO0_TIMCMP5)	32	R/W	0000_0000h	51.3.21/ 1541
4005_F518	Timer Compare N Register (FLEXIO0_TIMCMP6)	32	R/W	0000_0000h	51.3.21/ 1541
4005_F51C	Timer Compare N Register (FLEXIO0_TIMCMP7)	32	R/W	0000_0000h	51.3.21/ 1541
4005_F680	Shifter Buffer N Nibble Byte Swapped Register (FLEXIO0_SHIFTBUFNBS0)	32	R/W	0000_0000h	51.3.22/ 1542
4005_F684	Shifter Buffer N Nibble Byte Swapped Register (FLEXIO0_SHIFTBUFNBS1)	32	R/W	0000_0000h	51.3.22/ 1542

Table continues on the next page...

FLEXIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4005_F688	Shifter Buffer N Nibble Byte Swapped Register (FLEXIO0_SHIFTBUFNBS2)	32	R/W	0000_0000h	51.3.22/ 1542
4005_F68C	Shifter Buffer N Nibble Byte Swapped Register (FLEXIO0_SHIFTBUFNBS3)	32	R/W	0000_0000h	51.3.22/ 1542
4005_F690	Shifter Buffer N Nibble Byte Swapped Register (FLEXIO0_SHIFTBUFNBS4)	32	R/W	0000_0000h	51.3.22/ 1542
4005_F694	Shifter Buffer N Nibble Byte Swapped Register (FLEXIO0_SHIFTBUFNBS5)	32	R/W	0000_0000h	51.3.22/ 1542
4005_F698	Shifter Buffer N Nibble Byte Swapped Register (FLEXIO0_SHIFTBUFNBS6)	32	R/W	0000_0000h	51.3.22/ 1542
4005_F69C	Shifter Buffer N Nibble Byte Swapped Register (FLEXIO0_SHIFTBUFNBS7)	32	R/W	0000_0000h	51.3.22/ 1542
4005_F700	Shifter Buffer N Half Word Swapped Register (FLEXIO0_SHIFTBUFHWS0)	32	R/W	0000_0000h	51.3.23/ 1542
4005_F704	Shifter Buffer N Half Word Swapped Register (FLEXIO0_SHIFTBUFHWS1)	32	R/W	0000_0000h	51.3.23/ 1542
4005_F708	Shifter Buffer N Half Word Swapped Register (FLEXIO0_SHIFTBUFHWS2)	32	R/W	0000_0000h	51.3.23/ 1542
4005_F70C	Shifter Buffer N Half Word Swapped Register (FLEXIO0_SHIFTBUFHWS3)	32	R/W	0000_0000h	51.3.23/ 1542
4005_F710	Shifter Buffer N Half Word Swapped Register (FLEXIO0_SHIFTBUFHWS4)	32	R/W	0000_0000h	51.3.23/ 1542
4005_F714	Shifter Buffer N Half Word Swapped Register (FLEXIO0_SHIFTBUFHWS5)	32	R/W	0000_0000h	51.3.23/ 1542
4005_F718	Shifter Buffer N Half Word Swapped Register (FLEXIO0_SHIFTBUFHWS6)	32	R/W	0000_0000h	51.3.23/ 1542
4005_F71C	Shifter Buffer N Half Word Swapped Register (FLEXIO0_SHIFTBUFHWS7)	32	R/W	0000_0000h	51.3.23/ 1542
4005_F780	Shifter Buffer N Nibble Swapped Register (FLEXIO0_SHIFTBUFNIS0)	32	R/W	0000_0000h	51.3.24/ 1543
4005_F784	Shifter Buffer N Nibble Swapped Register (FLEXIO0_SHIFTBUFNIS1)	32	R/W	0000_0000h	51.3.24/ 1543
4005_F788	Shifter Buffer N Nibble Swapped Register (FLEXIO0_SHIFTBUFNIS2)	32	R/W	0000_0000h	51.3.24/ 1543
4005_F78C	Shifter Buffer N Nibble Swapped Register (FLEXIO0_SHIFTBUFNIS3)	32	R/W	0000_0000h	51.3.24/ 1543
4005_F790	Shifter Buffer N Nibble Swapped Register (FLEXIO0_SHIFTBUFNIS4)	32	R/W	0000_0000h	51.3.24/ 1543
4005_F794	Shifter Buffer N Nibble Swapped Register (FLEXIO0_SHIFTBUFNIS5)	32	R/W	0000_0000h	51.3.24/ 1543
4005_F798	Shifter Buffer N Nibble Swapped Register (FLEXIO0_SHIFTBUFNIS6)	32	R/W	0000_0000h	51.3.24/ 1543
4005_F79C	Shifter Buffer N Nibble Swapped Register (FLEXIO0_SHIFTBUFNIS7)	32	R/W	0000_0000h	51.3.24/ 1543

51.3.1 Version ID Register (FLEXIOx_VERID)

.

Address: 4005_F000h base + 0h offset = 4005_F000h

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	MAJOR								MINOR								FEATURE															
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

FLEXIOx_VERID field descriptions

Field	Description
31–24 MAJOR	Major Version Number This read only field returns the major version number for the module specification.
23–16 MINOR	Minor Version Number This read only field returns the minor version number for the module specification.
FEATURE	Feature Specification Number This read only field returns the feature set number. 0x0000 Standard features implemented. 0x0001 Supports state, logic and parallel modes.

51.3.2 Parameter Register (FLEXIOx_PARAM)

.

Address: 4005_F000h base + 4h offset = 4005_F004h

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	TRIGGER								PIN								TIMER								SHIFTER							
W																																
Reset	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

FLEXIOx_PARAM field descriptions

Field	Description
31–24 TRIGGER	Trigger Number Number of external triggers implemented.
23–16 PIN	Pin Number Number of Pins implemented.

Table continues on the next page...

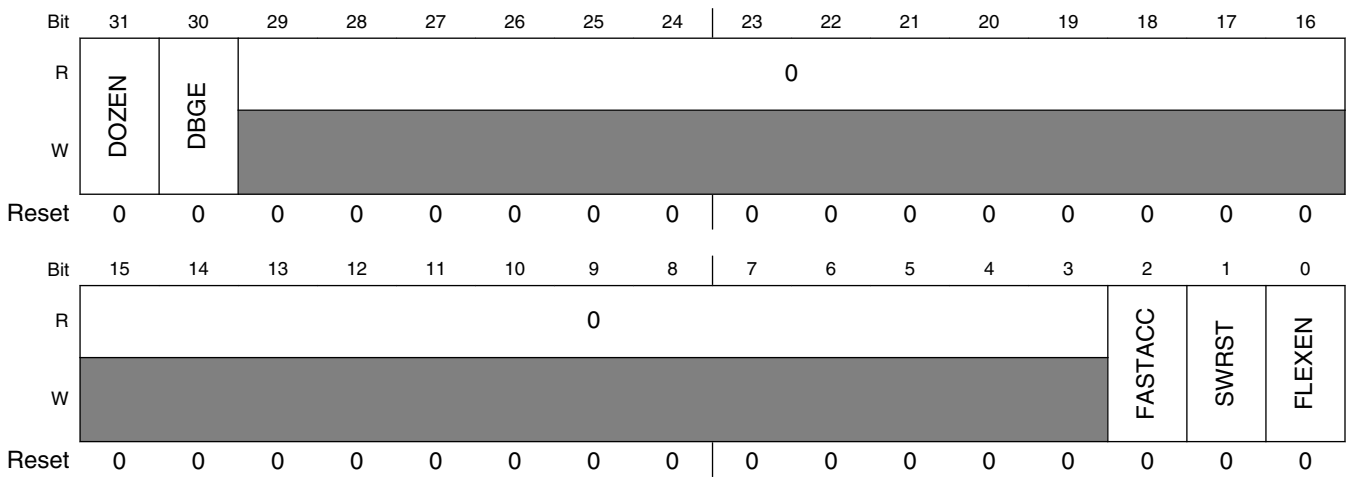
FLEXIOx_PARAM field descriptions (continued)

Field	Description
15–8 TIMER	Timer Number Number of Timers implemented.
SHIFTER	Shifter Number Number of Shifters implemented.

51.3.3 FlexIO Control Register (FLEXIOx_CTRL)

.

Address: 4005_F000h base + 8h offset = 4005_F008h



FLEXIOx_CTRL field descriptions

Field	Description
31 DOZEN	Doze Enable Disables FlexIO operation in Doze modes. This field is ignored and the FlexIO always disabled in low-leakage stop modes. 0 FlexIO enabled in Doze modes. 1 FlexIO disabled in Doze modes.
30 DBGE	Debug Enable Enables FlexIO operation in Debug mode. 0 FlexIO is disabled in debug modes. 1 FlexIO is enabled in debug modes
29–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FLEXIOx_CTRL field descriptions (continued)

Field	Description
2 FASTACC	Fast Access Enables fast register accesses to FlexIO registers, but requires the FlexIO clock to be at least twice the frequency of the bus clock. 0 Configures for normal register accesses to FlexIO 1 Configures for fast register accesses to FlexIO
1 SWRST	Software Reset The FlexIO Control Register is not affected by the software reset, all other logic in the FlexIO is affected by the software reset and register accesses are ignored until this bit is cleared. This register bit will remain set until cleared by software, and the reset has cleared in the FlexIO clock domain. 0 Software reset is disabled 1 Software reset is enabled, all FlexIO registers except the Control Register are reset.
0 FLEXEN	FlexIO Enable 0 FlexIO module is disabled. 1 FlexIO module is enabled.

51.3.4 Pin State Register (FLEXIOx_PIN)

Address: 4005_F000h base + Ch offset = 4005_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	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

FLEXIOx_PIN field descriptions

Field	Description
PDI	Pin Data Input Returns the input data on each of the FlexIO pins.

51.3.5 Shifter Status Register (FLEXIOx_SHIFTSTAT)

.

Address: 4005_F000h base + 10h offset = 4005_F010h

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																								SSF							
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

FLEXIOx_SHIFTSTAT field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SSF	Shifter Status Flag The shifter status flag is updated when one of the following events occurs: For SMOD=Receive, the status flag is set when SHIFTBUF has been loaded with data from Shifter (SHIFTBUF is full), and the status flag is cleared when SHIFTBUF register is read. For SMOD=Transmit, the status flag is set when SHIFTBUF data has been transferred to the Shifter (SHIFTBUF is empty) or when initially configured for SMOD=Transmit, and the status flag is cleared when the SHIFTBUF register is written. For SMOD=Match Store, the status flag is set when a match has occured between SHIFTBUF and Shifter, and the status flag is cleared when the SHIFTBUF register is read. For SMOD=Match Continuous, returns the current match result between the SHIFTBUF and Shifter. For SMOD=State, the status flag for a shifter will set when it is selected by the current state pointer. For SMOD=Logic, returns the current value of the programmable logic block output. The status flag can also be cleared by writing a logic one to the flag for all modes except Match Continuous/State/Logic. 0 Status flag is clear 1 Status flag is set

51.3.6 Shifter Error Register (FLEXIOx_SHIFTERR)

.

Address: 4005_F000h base + 14h offset = 4005_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																								SEF							
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

FLEXIOx_SHIFTERR field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEF	Shifter Error Flags The shifter error flag is set when one of the following events occurs: For SMOD=Receive, indicates Shifter was ready to store new data into SHIFTBUF before the previous data was read from SHIFTBUF (SHIFTBUF Overrun), or indicates that the received start or stop bit does not match the expected value. For SMOD=Transmit, indicates Shifter was ready to load new data from SHIFTBUF before new data had been written into SHIFTBUF (SHIFTBUF Underrun). For SMOD=Match Store, indicates a match event occured before the previous match data was read from SHIFTBUF (SHIFTBUF Overrun). For SMOD=Match Continuous, the error flag is set when a match has occured between SHIFTBUF and Shifter. For SMOD=Logic, the error flag is set when the output of the programmable logic block has asserted. Can be cleared by writing logic one to the flag. For SMOD=Match Continuous, can also be cleared when the SHIFTBUF register is read. 0 Shifter Error Flag is clear 1 Shifter Error Flag is set

51.3.7 Timer Status Register (FLEXIOx_TIMSTAT)

.

Address: 4005_F000h base + 18h offset = 4005_F018h

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																								TSF							
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

FLEXIOx_TIMSTAT field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TSF	Timer Status Flags The timer status flag sets depending on the timer mode, and can be cleared by writing logic one to the flag. In 8-bit counter mode, the timer status flag is set when the upper 8-bit counter equals zero and decrements, this also causes the counter to reload with the value in the compare register. In 8-bit PWM mode, the timer status flag is set when the upper 8-bit counter equals zero and decrements, this also causes the counter to reload with the value in the compare register..

Table continues on the next page...

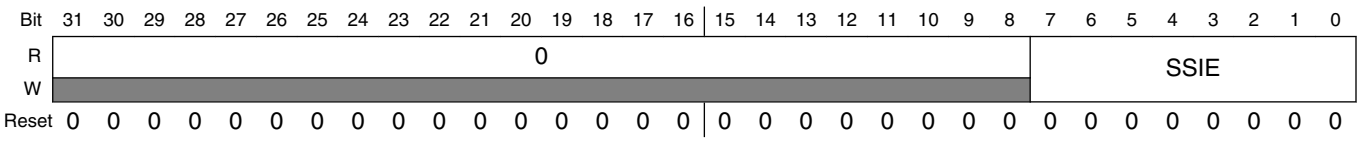
FLEXIOx_TIMSTAT field descriptions (continued)

Field	Description
	In 16-bit counter mode, the timer status flag is set when the 16-bit counter equals zero and decrements, this also causes the counter to reload with the value in the compare register.. 0 Timer Status Flag is clear 1 Timer Status Flag is set

51.3.8 Shifter Status Interrupt Enable (FLEXIOx_SHIFTSIEN)

.

Address: 4005_F000h base + 20h offset = 4005_F020h



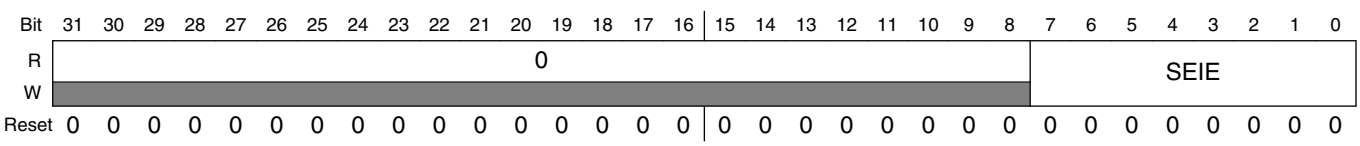
FLEXIOx_SHIFTSIEN field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SSIE	Shifter Status Interrupt Enable Enables interrupt generation when corresponding SSF is set. 0 Shifter Status Flag interrupt disabled 1 Shifter Status Flag interrupt enabled

51.3.9 Shifter Error Interrupt Enable (FLEXIOx_SHIFTEIEN)

.

Address: 4005_F000h base + 24h offset = 4005_F024h



FLEXIOx_SHIFTEIEN field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

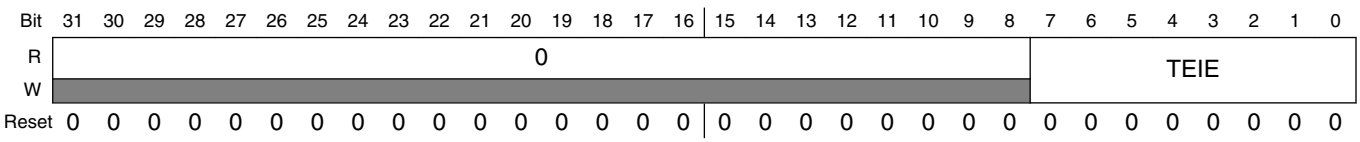
FLEXIOx_SHIFTEIEN field descriptions (continued)

Field	Description
SEIE	Shifter Error Interrupt Enable Enables interrupt generation when corresponding SEF is set. 0 Shifter Error Flag interrupt disabled 1 Shifter Error Flag interrupt enabled

51.3.10 Timer Interrupt Enable Register (FLEXIOx_TIMIEN)

.

Address: 4005_F000h base + 28h offset = 4005_F028h



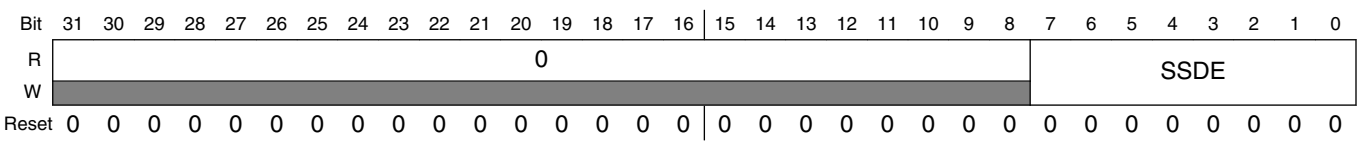
FLEXIOx_TIMIEN field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TEIE	Timer Status Interrupt Enable Enables interrupt generation when corresponding TSF is set. 0 Timer Status Flag interrupt is disabled 1 Timer Status Flag interrupt is enabled

51.3.11 Shifter Status DMA Enable (FLEXIOx_SHIFTSDEN)

.

Address: 4005_F000h base + 30h offset = 4005_F030h



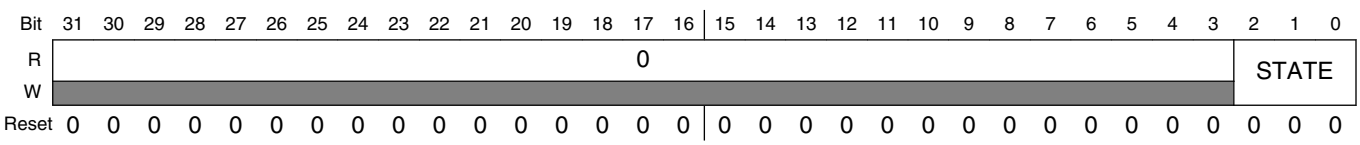
FLEXIOx_SHIFTSDEN field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SSDE	Shifter Status DMA Enable Enables DMA request generation when corresponding SSF is set. 0 Shifter Status Flag DMA request is disabled 1 Shifter Status Flag DMA request is enabled

51.3.12 Shifter State Register (FLEXIOx_SHIFTSTATE)

.

Address: 4005_F000h base + 40h offset = 4005_F040h



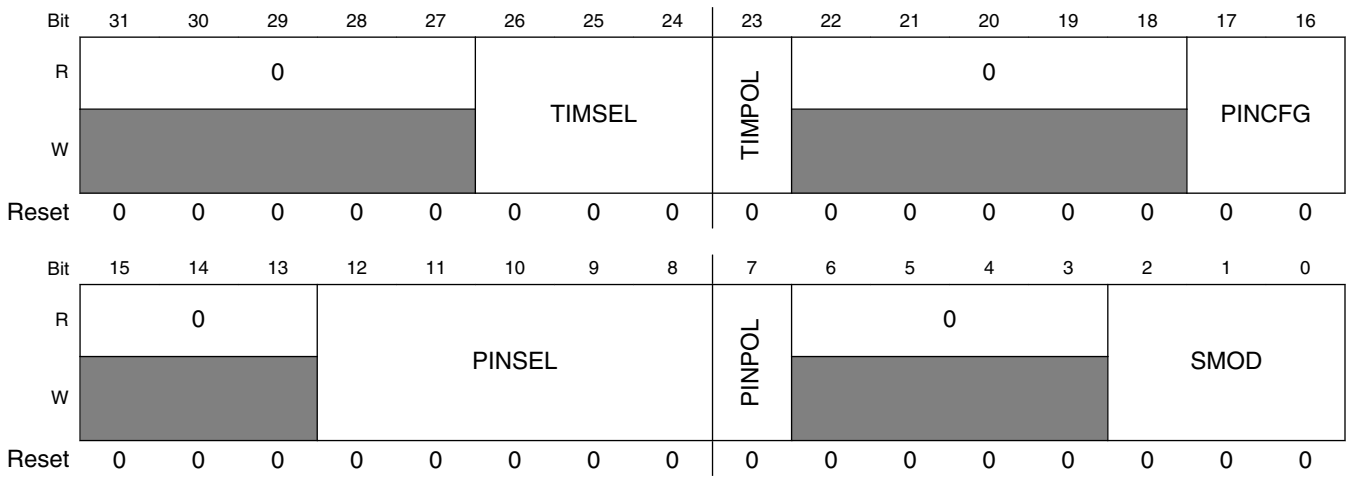
FLEXIOx_SHIFTSTATE field descriptions

Field	Description
31–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
STATE	Current State Pointer The current state field maintains a pointer to keep track of the current Shifter (configured for State mode) enabled to drive outputs and compute the next state. See 'State Mode' section for more detail.

51.3.13 Shifter Control N Register (FLEXIO*x*_SHIFTCTL*n*)

.

Address: 4005_F000h base + 80h offset + (4d × i), where i=0d to 7d



FLEXIO*x*_SHIFTCTL*n* field descriptions

Field	Description
31–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–24 TIMSEL	Timer Select Selects which Timer is used for controlling the logic/shift register and generating the Shift clock.
23 TIMPOL	Timer Polarity 0 Shift on posedge of Shift clock 1 Shift on negedge of Shift clock
22–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–16 PINCFG	Shifter Pin Configuration 00 Shifter pin output disabled 01 Shifter pin open drain or bidirectional output enable 10 Shifter pin bidirectional output data 11 Shifter pin output
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 PINSEL	Shifter Pin Select Selects which pin is used by the Shifter input or output.
7 PINPOL	Shifter Pin Polarity 0 Pin is active high 1 Pin is active low

Table continues on the next page...

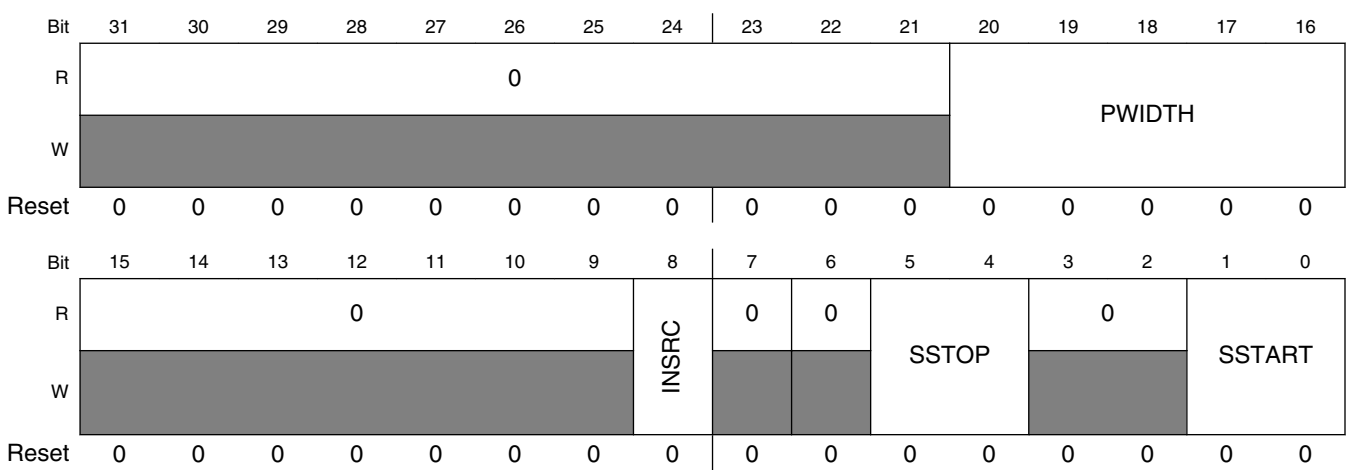
FLEXIOx_SHIFTCTLn field descriptions (continued)

Field	Description
6–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SMOD	Shifter Mode Configures the mode of the Shifter. 000 Disabled. 001 Receive mode. Captures the current Shifter content into the SHIFTBUF on expiration of the Timer. 010 Transmit mode. Load SHIFTBUF contents into the Shifter on expiration of the Timer. 011 Reserved. 100 Match Store mode. Shifter data is compared to SHIFTBUF content on expiration of the Timer. 101 Match Continuous mode. Shifter data is continuously compared to SHIFTBUF contents. 110 State mode. SHIFTBUF contents are used for storing programmable state attributes. 111 Logic mode. SHIFTBUF contents are used for implementing programmable logic look up table.

51.3.14 Shifter Configuration N Register (FLEXIOx_SHIFTCFGn)

.

Address: 4005_F000h base + 100h offset + (4d × i), where i=0d to 7d



FLEXIOx_SHIFTCFGn field descriptions

Field	Description
31–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–16 PWIDTH	Parallel Width For all Shifters, this register field configures the number of bits to be shifted on each Shift clock as follows: 1-bit shift for PWIDTH=0 4-bit shift for PWIDTH=1...3

Table continues on the next page...

FLEXIOx_SHIFTCFG*n* field descriptions (continued)

Field	Description
	8-bit shift for PWIDTH=4...7 16-bit shift for PWIDTH=8...15 32-bit shift for PWIDTH=16...31 For Shifters which support parallel transmit (SHIFTER0, SHIFTER4) or parallel receive (SHIFTER3, SHIFTER7), this register field, together with PSEL, also selects the pins to be driven or sampled on each Shift clock as follows: FXIO_D[PSEL+PWIDTH]:FXIO_D[PSEL] For SMOD=State, this field is used to disable state outputs. See 'State Mode' section for more detail.
15–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 INSRC	Input Source Selects the input source for the shifter. 0 Pin 1 Shifter N+1 Output
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–4 SSTOP	Shifter Stop bit For SMOD=Transmit, this field allows automatic stop bit insertion if the selected timer has also enabled a stop bit. For SMOD=Receive or Match Store, this field allows automatic stop bit checking if the selected timer has also enabled a stop bit. For SMOD=State, this field is used to disable state outputs. See 'State Mode' section for more detail. For SMOD=Logic, this field is used to mask logic pin inputs. See 'Logic Mode' section for more detail. 00 Stop bit disabled for transmitter/receiver/match store 01 Reserved for transmitter/receiver/match store 10 Transmitter outputs stop bit value 0 on store, receiver/match store sets error flag if stop bit is not 0 11 Transmitter outputs stop bit value 1 on store, receiver/match store sets error flag if stop bit is not 1
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SSTART	Shifter Start bit For SMOD=Transmit, this field allows automatic start bit insertion if the selected timer has also enabled a start bit. For SMOD=Receive or Match Store, this field allows automatic start bit checking if the selected timer has also enabled a start bit. For SMOD=State, this field is used to disable state outputs. See 'State Mode' section for more detail. For SMOD=Logic, this field is used to mask logic pin inputs. See 'Logic Mode' section for more detail. 00 Start bit disabled for transmitter/receiver/match store, transmitter loads data on enable 01 Start bit disabled for transmitter/receiver/match store, transmitter loads data on first shift

Table continues on the next page...

FLEXIOx_SHIFTCFGn field descriptions (continued)

Field	Description
	10 Transmitter outputs start bit value 0 before loading data on first shift, receiver/match store sets error flag if start bit is not 0 11 Transmitter outputs start bit value 1 before loading data on first shift, receiver/match store sets error flag if start bit is not 1

51.3.15 Shifter Buffer N Register (FLEXIOx_SHIFTBUFn)

.

Address: 4005_F000h base + 200h offset + (4d × 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 W	SHIFTBUF																															
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

FLEXIOx_SHIFTBUFn field descriptions

Field	Description
SHIFTBUF	Shift Buffer Shift buffer data is used for a variety of functions depending on the SMOD setting: For SMOD=Receive, Shifter data is transferred into SHIFTBUF at the expiration of Timer. For SMOD=Transmit, SHIFTBUF data is transferred into the Shifter before the Timer begins. For SMOD=Match Store/Continuous, SHIFTBUF[31:16] contains the data to be matched with the Shifter contents. The Match is checked either continuosly (Match Continous mode) or when the Timer expires (Match Store mode). SHIFTBUF[15:0] can be used to mask the match result (1=mask, 0=no mask). In Match Store mode, Shifter data [31:16] is written to SHIFTBUF[31:16] whenever a match event occurs. For SMOD=Logic, SHIFTBUF[31:0] is used to implement a 5-input, 32-bit programmable logic look-up table. See 'Logic Mode' section for more detail. For SMOD=State, SHIFTBUF[31:24] is used to drive the output value when this shifter is selected by the current state pointer and SHIFTBUF[23:0] is used to configure the value of the next state transition. See 'State Mode' section for more detail.

51.3.16 Shifter Buffer N Bit Swapped Register (FLEXIOx_SHIFTBUFBISn)

.

Address: 4005_F000h base + 280h offset + (4d × 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 W	SHIFTBUFBIS																															
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

FLEXIOx_SHIFTBUFBISn field descriptions

Field	Description
SHIFTBUFBIS	Shift Buffer Alias to SHIFTBUF register, except reads/writes to this register are bit swapped. Reads return SHIFTBUF[0:31].

51.3.17 Shifter Buffer N Byte Swapped Register (FLEXIOx_SHIFTBUFBYSn)

.

Address: 4005_F000h base + 300h offset + (4d × 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 W	SHIFTBUFBYS																															
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

FLEXIOx_SHIFTBUFBYSn field descriptions

Field	Description
SHIFTBUFBYS	Shift Buffer Alias to SHIFTBUF register, except reads/writes to this register are byte swapped. Reads return { SHIFTBUF[7:0], SHIFTBUF[15:8], SHIFTBUF[23:16], SHIFTBUF[31:24] }.

51.3.18 Shifter Buffer N Bit Byte Swapped Register (FLEXIOx_SHIFTBUFBBSn)

.

Address: 4005_F000h base + 380h offset + (4d × 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 W	SHIFTBUFBBS																															
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

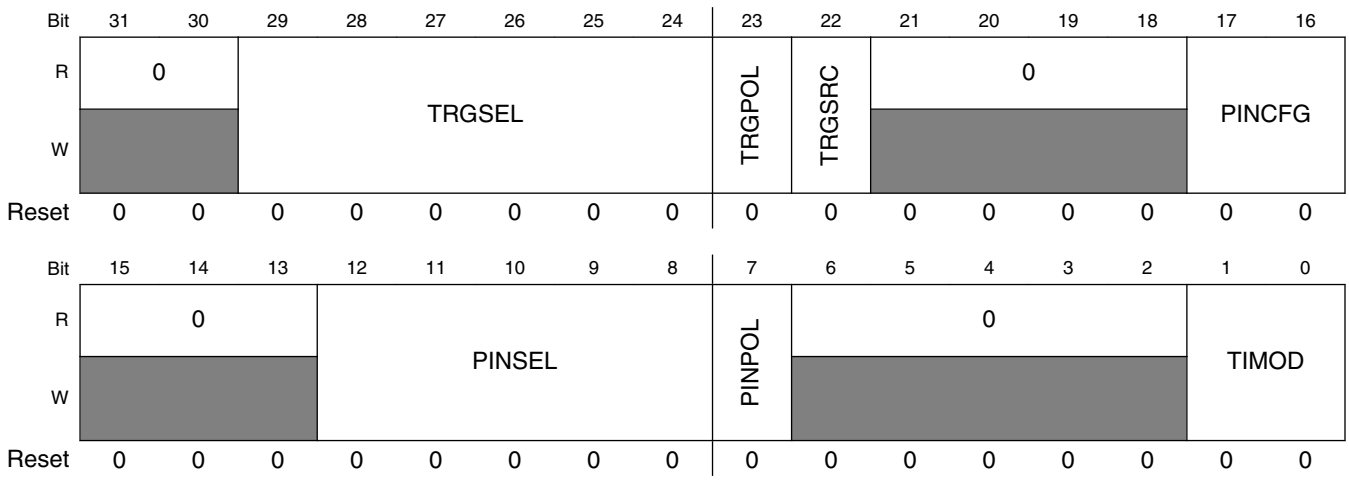
FLEXIOx_SHIFTBUFBBSn field descriptions

Field	Description
SHIFTBUFBBS	Shift Buffer Alias to SHIFTBUF register, except reads/writes to this register are bit swapped within each byte. Reads return { SHIFTBUF[24:31], SHIFTBUF[16:23], SHIFTBUF[8:15], SHIFTBUF[0:7] }.

51.3.19 Timer Control N Register (FLEXIOx_TIMCTLn)

.

Address: 4005_F000h base + 400h offset + (4d × i), where i=0d to 7d



FLEXIOx_TIMCTLn field descriptions

Field	Description
31–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–24 TRGSEL	Trigger Select The valid values for TRGSEL will depend on the FLEXIO_PARAM register. • When TRGSRC = 1, the valid values for N will depend on PIN, TIMER, SHIFTER fields in the FLEXIO_PARAM register. • When TRGSRC = 0, the valid values for N will depend on TRIGGER field in FLEXIO_PARAM register. Refer to the chip configuration section for external trigger selection. The internal trigger selection is configured as follows: {N,00} pin 2N input {N,01} shifter N status flag {N,10} pin 2N+1 input {N,11} timer N trigger output
23 TRGPOL	Trigger Polarity 0 Trigger active high 1 Trigger active low
22 TRGSRC	Trigger Source 0 External trigger selected 1 Internal trigger selected
21–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

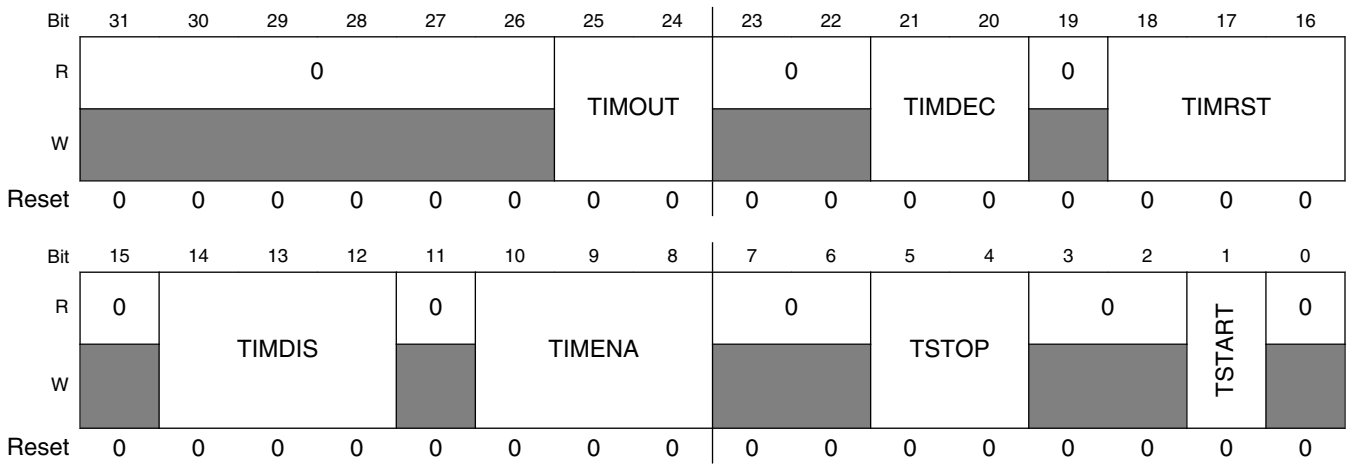
FLEXIO*x*_TIMCTL*n* field descriptions (continued)

Field	Description
17–16 PINCFG	Timer Pin Configuration 00 Timer pin output disabled 01 Timer pin open drain or bidirectional output enable 10 Timer pin bidirectional output data 11 Timer pin output
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 PINSEL	Timer Pin Select Selects which pin is used by the Timer input or output.
7 PINPOL	Timer Pin Polarity 0 Pin is active high 1 Pin is active low
6–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TIMOD	Timer Mode In 8-bit counter mode, the lower 8-bits of the counter and compare register are used to configure the baud rate of the timer shift clock and the upper 8-bits are used to configure the shifter bit count. In 8-bit PWM mode, the lower 8-bits of the counter and compare register are used to configure the high period of the timer shift clock and the upper 8-bits are used to configure the low period of the timer shift clock. The shifter bit count is configured using another timer or external signal. In 16-bit counter mode, the full 16-bits of the counter and compare register are used to configure either the baud rate of the shift clock or the shifter bit count. 00 Timer Disabled. 01 Dual 8-bit counters baud/bit mode. 10 Dual 8-bit counters PWM mode. 11 Single 16-bit counter mode.

51.3.20 Timer Configuration N Register (FLEXIO*x*_TIMCFG*n*)

The options to enable or disable the timer using the Timer N-1 enable or disable are reserved when N is evenly divisible by 4 (eg: Timer 0).

Address: 4005_F000h base + 480h offset + (4d × i), where i=0d to 7d



FLEXIO*x*_TIMCFG*n* field descriptions

Field	Description
31–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–24 TIMOUT	Timer Output Configures the initial state of the Timer Output and whether it is affected by the Timer reset. 00 Timer output is logic one when enabled and is not affected by timer reset 01 Timer output is logic zero when enabled and is not affected by timer reset 10 Timer output is logic one when enabled and on timer reset 11 Timer output is logic zero when enabled and on timer reset
23–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–20 TIMDEC	Timer Decrement Configures the source of the Timer decrement and the source of the Shift clock. 00 Decrement counter on FlexIO clock, Shift clock equals Timer output. 01 Decrement counter on Trigger input (both edges), Shift clock equals Timer output. 10 Decrement counter on Pin input (both edges), Shift clock equals Pin input. 11 Decrement counter on Trigger input (both edges), Shift clock equals Trigger input.
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–16 TIMRST	Timer Reset

Table continues on the next page...

FLEXIO*x*_TIMCFG*n* field descriptions (continued)

Field	Description
	Configures the condition that causes the timer counter (and optionally the timer output) to be reset. In 8-bit counter mode, the timer reset will only reset the lower 8-bits that configure the baud rate. In all other modes, the timer reset will reset the full 16-bits of the counter. 000 Timer never reset 001 Reserved 010 Timer reset on Timer Pin equal to Timer Output 011 Timer reset on Timer Trigger equal to Timer Output 100 Timer reset on Timer Pin rising edge 101 Reserved 110 Timer reset on Trigger rising edge 111 Timer reset on Trigger rising or falling edge
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–12 TIMDIS	Timer Disable Configures the condition that causes the Timer to be disabled and stop decrementing. 000 Timer never disabled 001 Timer disabled on Timer N-1 disable 010 Timer disabled on Timer compare 011 Timer disabled on Timer compare and Trigger Low 100 Timer disabled on Pin rising or falling edge 101 Timer disabled on Pin rising or falling edge provided Trigger is high 110 Timer disabled on Trigger falling edge 111 Reserved
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–8 TIMENA	Timer Enable Configures the condition that causes the Timer to be enabled and start decrementing. 000 Timer always enabled 001 Timer enabled on Timer N-1 enable 010 Timer enabled on Trigger high 011 Timer enabled on Trigger high and Pin high 100 Timer enabled on Pin rising edge 101 Timer enabled on Pin rising edge and Trigger high 110 Timer enabled on Trigger rising edge 111 Timer enabled on Trigger rising or falling edge
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–4 TSTOP	Timer Stop Bit The stop bit can be added on a timer compare (between each word) or on a timer disable. When stop bit is enabled, configured shifters will output the contents of the stop bit when the timer is disabled. When stop bit is enabled on timer disable, the timer remains disabled until the next rising edge of the shift clock. If configured for both timer compare and timer disable, only one stop bit is inserted on timer disable. 00 Stop bit disabled

Table continues on the next page...

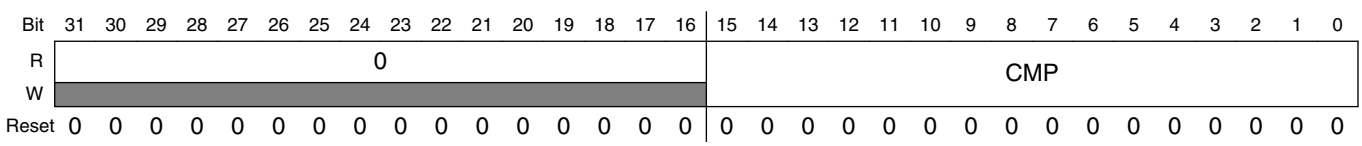
FLEXIO*x*_TIMCFG*n* field descriptions (continued)

Field	Description
	01 Stop bit is enabled on timer compare 10 Stop bit is enabled on timer disable 11 Stop bit is enabled on timer compare and timer disable
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 TSTART	Timer Start Bit When start bit is enabled, configured shifters will output the contents of the start bit when the timer is enabled and the timer counter will reload from the compare register on the first rising edge of the shift clock. 0 Start bit disabled 1 Start bit enabled
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

51.3.21 Timer Compare N Register (FLEXIO*x*_TIMCMP*n*)

.

Address: 4005_F000h base + 500h offset + (4d × i), where i=0d to 7d



FLEXIO*x*_TIMCMP*n* field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CMP	Timer Compare Value The timer compare value is loaded into the timer counter when the timer is first enabled, when the timer is reset and when the timer decrements down to zero. In dual 8-bit counters baud/bit mode, the lower 8-bits configures the baud rate divider equal to (CMP[7:0] + 1) * 2. The upper 8-bits configure the number of bits in each word equal to (CMP[15:8] + 1) / 2. In dual 8-bit counters PWM mode, the lower 8-bits configure the high period of the output to (CMP[7:0] + 1) and the upper 8-bits configure the low period of the output to (CMP[15:8] + 1). In 16-bit counter mode, the compare value can be used to generate the baud rate divider (if shift clock source is timer output) to equal (CMP[15:0] + 1) * 2. When the shift clock source is a pin or trigger input, the compare register is used to set the number of bits in each word equal to (CMP[15:0] + 1) / 2.

51.3.22 Shifter Buffer N Nibble Byte Swapped Register (FLEXIO*x*_SHIFTBUFNBS*n*)

.

Address: 4005_F000h base + 680h offset + (4d × 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 W	SHIFTBUFNBS																															
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

FLEXIO*x*_SHIFTBUFNBS*n* field descriptions

Field	Description
SHIFTBUFNBS	Shift Buffer Alias to SHIFTBUF register, except reads/writes to this register are nibble swapped within each byte. Reads return { SHIFTBUF[27:24], SHIFTBUF[31:28], SHIFTBUF[19:16], SHIFTBUF[23:20], SHIFTBUF[11:8], SHIFTBUF[15:12], SHIFTBUF[3:0], SHIFTBUF[7:4] }.

51.3.23 Shifter Buffer N Half Word Swapped Register (FLEXIO*x*_SHIFTBUFHWS*n*)

.

Address: 4005_F000h base + 700h offset + (4d × 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 W	SHIFTBUFHWS																															
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

FLEXIO*x*_SHIFTBUFHWS*n* field descriptions

Field	Description
SHIFTBUFHWS	Shift Buffer Alias to SHIFTBUF register, except reads/writes to this register are half word swapped. Reads return { SHIFTBUF[15:0], SHIFTBUF[31:24] }.

51.3.24 Shifter Buffer N Nibble Swapped Register (FLEXIOx_SHIFTBUFNIS*n*)

.

Address: 4005_F000h base + 780h offset + (4d × 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 W	SHIFTBUFNIS																															
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

FLEXIOx_SHIFTBUFNIS*n* field descriptions

Field	Description
SHIFTBUFNIS	Shift Buffer Alias to SHIFTBUF register, except reads/writes to this register are nibble swapped. Reads return { SHIFTBUF[3:0], SHIFTBUF[7:4], SHIFTBUF[11:8], SHIFTBUF[15:12], SHIFTBUF[19:16], SHIFTBUF[23:20], SHIFTBUF[27:24], SHIFTBUF[31:28] }.

51.4 Functional description

51.4.1 Shifter operation

Shifters are responsible for buffering and shifting data into or out of the FlexIO. The timing of shift, load and store events are controlled by the Timer assigned to the Shifter via the SHIFTCTL[TIMSEL] register. The Shifters are designed to support either DMA, interrupt or polled operation. The following block diagram provides a detailed view of the Shifter microarchitecture.

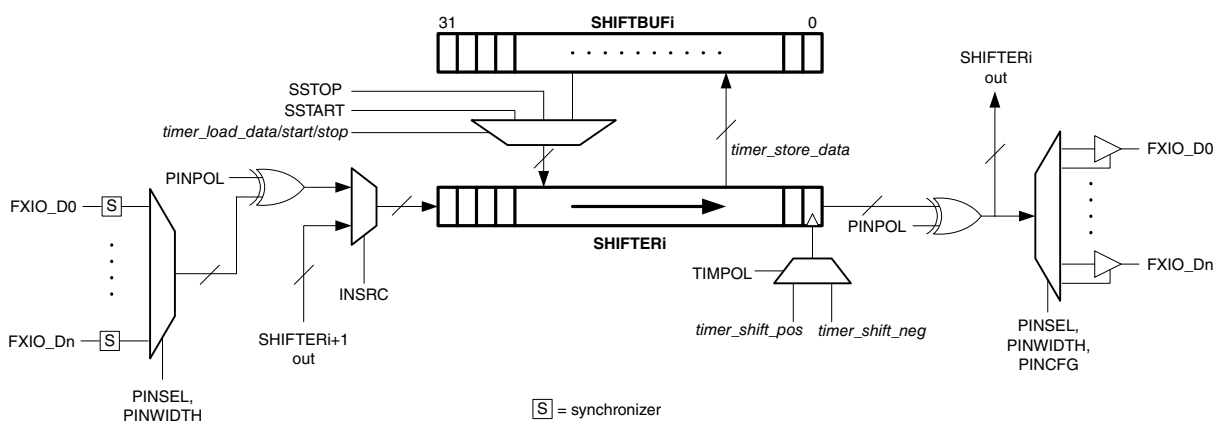


Figure 51-2. Shifter Microarchitecture

51.4.1.1 Transmit Mode

When configured for Transmit mode (SHIFTCTL[SMOD]=Transmit), the shifter will load data from the SHIFTBUF register and shift data out when a load event is signalled by the assigned Timer. An optional start/stop bit can also be automatically loaded before/after SHIFTBUF data by configuring the SHIFTCFG[SSTART], TIMCFG[TSTART] or SHIFTCFG[SSTOP], TIMCFG[TSTOP] registers in the Shifter and Timer. Note that the shifter will immediately load a stop bit when the Shifter is initially configured for Transmit mode if a stop bit is enabled.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests will set when data has been loaded from the SHIFTBUF register into the Shifter or when the Shifter is initially configured into Transmit mode. The flag will clear when new data has been written into the SHIFTBUF register.

The Shifter Error Flag (SHIFTERR[SEF]) and any enabled interrupts will set when an attempt to load data from an empty SHIFTBUF register occurs (buffer underrun). The flag can be cleared by writing it with logic 1.

51.4.1.2 Receive Mode

When configured for Receive mode (SHIFTCTL[SMOD]=Receive), the shifter will shift data in and store data into the SHIFTBUF register when a store event is signalled by the assigned Timer. Checking for a start/stop bit can be enabled before/after shifter data is sampled by configuring the SHIFTCFG[SSTART], TIMCFG[TSTART] or SHIFTCFG[SSTOP], TIMCFG[TSTOP] registers in the Shifter and Timer.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests will set when data has been stored into the SHIFTBUF register from the Shifter. The flag will clear when the data has been read from the SHIFTBUF register.

The Shifter Error Flag (SHIFTERR[SEF]) and any enabled interrupts will set when an attempt to store data into a full SHIFTBUF register occurs (buffer overrun) or when a mismatch occurs on a start/stop bit check. The flag can be cleared by writing it with logic 1.

51.4.1.3 Match Store Mode

When configured for Match Store mode (SHIFTCTL[SMOD]=Match Store), the shifter will shift data in, check for a match result and store matched data into the SHIFTBUF register when a store event is signalled by the assigned Timer. Checking for a start/stop bit can be enabled before/after shifter data is sampled by configuring the SHIFTCFG[SSTART], TIMCFG[TSTART] or SHIFTCFG[SSTOP], TIMCFG[TSTOP] registers in the Shifter and Timer. Up to 16-bits of data can be compared using SHIFTBUF[31:16] to configure the data to be matched and SHIFTBUF[15:0] to mask the match result.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests will set when a match occurs and matched data has been stored into the SHIFTBUF register from the Shifter. The flag will clear when the matched data has been read from the SHIFTBUF register.

The Shifter Error Flag (SHIFTERR[SEF]) and any enabled interrupts will set when an attempt to store matched data into a full SHIFTBUF register occurs (buffer overrun) or when a mismatch occurs on a start/stop bit check. The flag can be cleared by writing it with logic 1.

51.4.1.4 Match Continuous Mode

When configured for Match Continuous mode (SHIFTCTL[SMOD]=Match Continuous), the shifter will shift data in and continuously check for a match result whenever a shift event is signalled by the assigned Timer. Up to 16-bits of data can be compared using SHIFTBUF[31:16] to configure the data to be matched and SHIFTBUF[15:0] to mask the match result.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests will set when a match occurs. The flag will clear automatically as soon as there is no longer a match between Shifter data and SHIFTBUF register.

The Shifter Error Flag (SHIFTERR[SEF]) and any enabled interrupts will set when a match occurs. The flag will clear when there is a read from the SHIFTBUF register or it written with logic 1.

51.4.1.5 State Mode

Using State mode enables the user to implement any state machine with up to 8 states, 8 outputs and 3 selectable inputs per state. This feature allows basic control functions to be offloaded from the CPU using FlexIO hardware.

When configured for State mode (SHIFTCTL[SMOD]=State), the SHIFTBUF register is used to drive the output and compute next state values when the Shifter has been selected by the current state pointer (SHIFTSTATE[STATE]). The following diagram provides a detailed view of Shifter microarchitecture when configured for State mode.

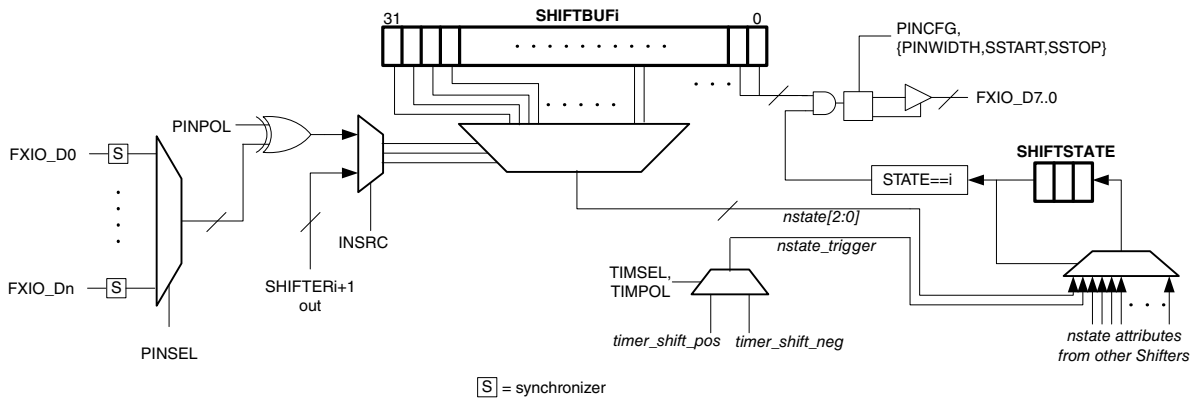


Figure 51-3. State Microarchitecture

When Shifter i has been selected by the current state pointer, output pins FXIO_D[7:0] will be driven by SHIFTBUFi[31:24] using the configuration set by SHIFTCTL[PINCFG]. When set, {PWIDTH[3:0],SSTOP[1:0],SSTART[1:0]} are respectively used to disable the output drive on pins FXIO_D[7:0] for state machine applications which require less than 8 output pins.

The next state value is computed using the 3 input pins selected by SHIFTCFG[PSEL] together with SHIFTBUFi[23:0]. Note that each state could potentially use a different set of 3 input pins. The following table details how the next state value is computed when the current state pointer is pointing to Shifter i.

Table 51-3. Next State computation for SHIFTSTATE[STATE]=i

FXIO_D[PSEL+2]	FXIO_D[PSEL+1]	FXIO_D[PSEL]	Next State Value
0	0	0	SHIFTBUFi[2:0]
0	0	1	SHIFTBUFi[5:3]
0	1	0	SHIFTBUFi[8:6]
0	1	1	SHIFTBUFi[11:9]
...	...	...	...
1	1	1	SHIFTBUFi[23:21]

Note that other shifters/timers could potentially be configured to drive the input pins of a given state, allowing the user to create complex combinations of shifters/timers as desired e.g. the output of a Shifter configured for logic mode could potentially be used to drive a state machine input.

The next state transition is triggered using the Timer output selected by SHIFTCTL[TIMSEL] with polarity controlled by SHIFTCTL[TIMPOL]. Note that each state could potentially use a different Timer to trigger each next state transition, allowing a variety of internal/external trigger sources and clocking configurations to be used (see Timer section for more detail).

The current state pointer defaults to Shifter 0 at reset, however it can be written by the user to select a different Shifter for the initial state. If the current state pointer selects a Shifter which is not configured for State mode, then outputs will not be driven and a next state transition is never triggered.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests will set whenever the Shifter has been selected by the current state pointer. The flag will clear when the current state pointer is updated to a different Shifter.

51.4.1.6 Logic Mode

Using logic mode enables the user to implement a small amount of programmable digital logic within a FlexIO Shifter. This feature allows board-level glue logic to be integrated on-chip using FlexIO hardware.

When configured for Logic mode (SHIFTCTL[SMOD]=Logic), the SHIFTBUF register is used to implement a 5-input, 32-bit programmable logic look-up table. The following diagram provides a detailed view of Shifter microarchitecture when configured for Logic mode.

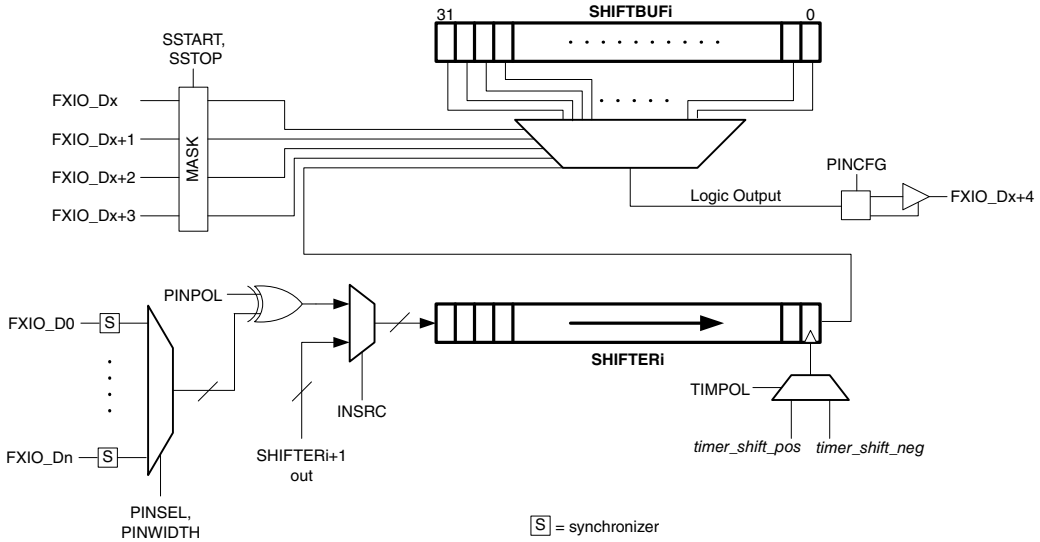


Figure 51-4. Logic Microarchitecture

The look-up table is driven using 4 pin inputs (maskable using SHIFTCFG[SSTOP] and SHFITCFG[SSTART]) plus 1 input from the internal shifter and can be configured to drive an output pin using the SHIFTCTL[PINCFG] field. Pin inputs and outputs are fixed for each logic look-up table and are not selectable. The following table lists the logic output value selected by the look-up table for Shifter 'i'.

Table 51-4. Logic Look-up table for Shifter 'i'

SHIFTERi[0]	FXIO_D[x+3][1]	FXIO_D[x+2]	FXIO_D[x+1]	FXIO_D[x]	Logic Output to FXIO_D[x+4]
0	0	0	0	0	SHIFTBUFi[0]
0	0	0	0	1	SHIFTBUFi[1]
0	0	0	1	0	SHIFTBUFi[2]
0	0	0	1	1	SHIFTBUFi[3]
...	...	...	...	...	...
1	1	1	1	1	SHIFTBUFi[31]

1. for Shifter i=0...3, x=i

   for Shifter i=4...7, x=i+4

To minimize output glitches, SHIFTCFG[SSTOP] and SHIFTCFG[SSTART] can be used to mask unused input pins. When set, {SSTOP[1:0], SSTART[1:0]} will mask FXIO_D[x+3]...FXIO_D[x] inputs respectively, so that any transitions on these pins will not cause the logic output to glitch.

Note that other shifters/timers could potentially be configured to drive the input pins of a given look-up table (without synchronization), allowing the user to concatenate look-up tables or create complex combinations of shifters/timers as desired.

SHIFTCFG[PWIDTH] will control the number of delay stages introduced by the internal shifter input (SHIFTERi[0]). For example, when configured for 1-bit shift (PWIDTH=0), the internal shifter will introduce a 32 Shift clock delay before passing its input (selected by SHIFTCTL[PSEL]) to the look-up table. When configured for 32-bit shift (PWIDTH=16...31), the internal shifter will introduce a 1 Shift clock delay to its input.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests will set whenever the output pin allocated to the logic look-up table has a value of 1 (after being synchronized to the FlexIO clock). The flag will clear when the output pin has a value of 0. This also allows the SSF flag to be used as a trigger to a Timer if desired.

The Shifter Error Flag (SHIFTERR[SEF]) and any enabled interrupts will set when the output pin allocated to the logic look-up table has a value of 1. The flag can be cleared by writing it with logic 1.

51.4.2 Timer operation

The FlexIO 16-bit timers control the loading, shifting and storing of the shift registers, the counters load the contents of the compare register and decrement down to zero on the FlexIO clock. They can perform generic timer functions such as generating a clock or select output or a PWM waveform. Timers can be configured to enable in response to a trigger, pin or shifter condition; decrement always or only on a trigger or pin edge; reset in response to a trigger or pin condition; and disable on a trigger or pin condition or on a timer compare. Timers can optionally include a start condition and/or stop condition.

Each timer operates independently, although a timer can be configured to enable or disable at the same time as the previous timer (eg: timer1 can enable or disable at the same time as timer 0) and a timer output can be used to trigger any other timer. The trigger used by each timer is configured independently and can be configured to be a timer output, shifter status flag, pin input or an external trigger input (refer to the chip configuration section for details on the external trigger connections). The trigger configuration is separate from the pin configuration, which can be configured for input, output data or output enable.

The Timer Configuration Register (TIMCFGn) should be configured before setting the Timer Mode (TIMOD). Once the TIMOD is configured for the desired mode, when the condition configured by timer enable (TIMENA) is detected then the following events occur.

- Timer counter will load the current value of the Compare Register and start decrementing as configured by TIMDEC.

- Timer output will set depending on the TIMOUT configuration.
- Transmit shifters controlled by this timer will either output their start bit value, or load the shift register from the shift buffer and output the first bit, as configured by SSTART.

The Timer will then generate the timer output and timer shift clock depending on the TIMOD and TIMDEC fields. The shifter clock is either equal to the timer output (when TIMDEC=00 or 01) or equal to the decrement clock (when TIMDEC=10 or 11). When TIMDEC is configured to decrement from a pin or trigger, the timer will decrement on both rising and falling edges.

When the Timer is configured to reset as configured in the TIMRST field then the Timer counter will load the current value of the Compare Register again, the timer output may also be affected by the reset as configured in TIMOUT.

If the Timer start bit is enabled, the timer counter will reload with the compare register on the first rising edge of the shift clock after the timer starts decrementing. If there is no falling edge on the shift clock before the first rising edge (for example, when TIMOUT=1), a shifter that is configured to shift on falling edge and load on the first shift will not load correctly.

When configured for 8-bit counter mode, whenever the lower 8-bit counter decrements to zero the timer output will toggle, the lower 8-bit counter register will reload from the compare register and the upper 8-bit counter will decrement. For 8-bit PWM mode, the lower 8-bit counter will only decrement when the output is high and the upper 8-bit counter will only decrement when the output is low. The timer output will toggle whenever either lower or upper 8-bit counter decrements to zero.

When the timer decrements to zero, a compare event occurs depending on the timer mode. For 8-bit counter or PWM modes, both halves of the counter must equal zero and the upper half must decrement for the timer compare event to occur, while in 16-bit mode the entire counter must equal zero and decrement. The timer compare event will cause the timer status flag to set, the timer counter to load the contents of the timer compare register, the timer output to toggle, any configured transmit shift registers to load and any configured receive shift registers to store .

When the is Timer is configured to add a stop bit on each compare, the following additional events will occur.

- Transmit shifters controlled by this timer will output their stop bit value (if configured by SSTOP).
- Receive shifters controlled by this timer will store the contents of the shift register in their shift buffer, as configured by SSTOP.
- On the first rising edge of the shifter clock after the compare, the timer counter will reload the current value of the Compare Register.

Transmit shifters must be configured to load on the first shift when the timer is configured to insert a stop bit on each compare.

When the condition configured by timer disable (TIMDIS) is detected, the following events occur.

- Timer counter will reload the current value of the Compare Register and start decrementing as configured by TIMDEC.
- Timer output will clear.
- Transmit shifters controlled by this timer will output their stop bit value (if configured by SSTOP).
- Receive shifters controlled by this timer will store the contents of the shift register in their shift buffer, as configured by SSTOP.

If the timer stop bit is enabled, the timer counter will continue decrementing until the next rising edge of the shift clock is detected, at which point it will finish. A timer enable condition can be detected in the same cycle as a timer disable condition (if timer stop bit is disabled), or on the first rising edge of the shift clock after the disable condition (if stop bit is enabled). Receive shift registers will stop bit enabled will store the contents of the shift register into the shift buffer and verify the state of the input data on the configured shift edge while the timer is in the stop state condition. If there is no configured edge between the timer disable and the next rising edge of the shift clock then the final store and verify do not occur.

51.4.3 Pin operation

The pin configuration for each timer and shifter can be configured to use any FlexIO pin with either polarity. Each timer and shifter can be configured as an input, output data, output enable or bidirectional output. A pin configured for output enable can be used as an open drain (with inverted polarity, since the output enable assertion would cause logic zero to be output on the pin) or to control the enable on the bidirectional output. Any timer or shifter could be configured to control the output enable for a pin where the bidirectional output data is driven by another timer or shifter.

In addition, shifters can also be configured to use multiple FlexIO pins in parallel using the SHIFTCFG[PWIDTH] field. PWIDTH is used to configure the following settings of a shifter:

1. Number of bits shifted per Shift clock.
2. Number of pins driven by the shifter per Shift clock (only on shifters supporting parallel transmit i.e. SHIFTER0, SHIFTER4).
3. Number of pins sampled by the shifter per Shift clock (only on Shifter supporting parallel receive i.e. SHIFTER3, SHIFTER7).

When configured for parallel shift, either 4, 8, 16 or 32-bits can be shifted on every Shift clock. If an adjacent shifter is selected as the input source (SHIFTCFG[INSRC]=1), the least significant 4, 8, 16 or 32-bits from the adjacent shifter will be sampled on each Shift clock.

For shifters supporting parallel receive (SHIFTER3, SHIFTER7), the shifter can be configured to sample multiple pins (SHIFTCFG[INSRC]=0), with PWIDTH and PSEL selecting the pins as follows: FXIO_D[PSEL+PWIDTH]:FXIO_D[PSEL]. Note that if PWIDTH is less than the number of bits being shifted on each Shift clock, then the most significant bits will be masked with 0 e.g. if PSEL=7 and PWIDTH=6, then SHIFTER[31:24] will sample {0,0,FXIO_D[12:7]} on each Shift clock.

For shifters supporting parallel transmit (SHIFTER0, SHIFTER4), the shifter can be configured to drive multiple pins using SHIFTCTL[PINCFG], with PWIDTH and PSEL selecting the pins as follows: FXIO_D[PSEL+PWIDTH]:FXIO_D[PSEL]. Note that if PWIDTH is less than the number of bits being shifted on each Shift clock, then the most significant pins will not be driven e.g. if PSEL=7 and PWIDTH=6, then SHIFTER[5:0] will drive only FXIO_D[12:7] on each Shift clock.

When configuring a pin as an input (this includes a timer trigger configured as a pin input), the input signal is first synchronized to the FlexIO clock before the signal is used by a timer or shifter. This introduces a small latency of between 0.5 to 1.5 FlexIO clock cycles when using an external pin input to generate an output or control a shifter. This sets the maximum setup time at 1.5 FlexIO clock cycles.

If an input is used by more than one timer or shifter then the synchronization occurs once to ensure any edge is seen on the same cycle by all timers and shifters using that input.

Note that FlexIO pins are also connected internally, configuring a FlexIO shifter or timer to output data on an unused pin will make an internal connection that allows other shifters and timer to use this pin as an input. This allows a shifter output to be used to trigger a timer or a timer output to be shifted into a shifter. This path is also synchronized to the FlexIO clock and therefore incurs a 1 cycle latency.

So when using a Pin input as a Timer Trigger, Timer Clock or Shifter Data Input, the following synchronization delays occur:

1. 0.5 – 1.5 FlexIO clock cycles for external pin
2. 1 FlexIO clock cycle for an internally driven pin

For timing considerations such as output valid time and input setup time for specific applications (SPI Master, SPI Slave, I2C Master, I2S Master, I2S Slave) please refer to the FlexIO Application Information Section.

51.5 Application Information

This section provides examples for a variety of FlexIO module applications.

51.5.1 UART Transmit

UART transmit can be supported using one Timer, one Shifter and one Pin (two Pins if supporting CTS). The start and stop bit insertion is handled automatically and multiple transfers can be supported using DMA controller. The timer status flag can be used to indicate when the stop bit of each word is transmitted.

Break and idle characters require software intervention, before transmitting a break or idle character the SSTART and SSTOP fields should be altered to transmit the required state and the data to transmit must equal 0xFF or 0x00. Supporting a second stop bit requires the stop bit to be inserted into the data stream using software (and increasing the number of bits to transmit). Note that when performing byte writes to SHIFTBUFn (or SHIFTBUFBIS for transmitting MSB first), the rest of the register remains unaltered allowing an address mark bit or additional stop bit to remain undisturbed.

FlexIO does not support automatic insertion of parity bits.

Table 51-5. UART Transmit Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0032	Configure start bit of 0 and stop bit of 1.
SHIFTCTLn	0x0003_0002	Configure transmit using Timer 0 on posedge of clock with output data on Pin 0. Can invert output data by setting PINPOL, or can support open drain by setting PINPOL=0x1 and PINCFG=0x1.
TIMCMPn	0x0000_0F01	Configure 8-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0000_2222	Configure start bit, stop bit, enable on trigger low and disable on compare. Can support CTS by configuring TIMEN=0x3.
TIMCTLn	0x01C0_0001	Configure dual 8-bit counter using Shifter 0 status flag as inverted internal trigger source. Can support CTS by configuring PINSEL=0x1 (for Pin 1) and PINPOL=0x1.
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF[7:0] to initiate an 8-bit

Table 51-5. UART Transmit Configuration

Register	Value	Comments
		transfer, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBBS[7:0] register instead.

51.5.2 UART Receive

UART receive can be supported using one Timer, one Shifter and one Pin (two Timers and two Pins if supporting RTS). The start and stop bit verification is handled automatically and multiple transfers can be supported using the DMA controller. The timer status flag can be used to indicate when the stop bit of each word is received.

Triple voting of the received data is not supported by FlexIO, data is sampled only once in the middle of each bit. Another timer can be used to implement a glitch filter on the incoming data, another Timer can also be used to detect an idle line of programmable length. Break characters will cause the error flag to set and the shifter buffer register will return 0x00.

FlexIO does not support automatic verification of parity bits.

Table 51-6. UART Receiver Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0032	Configure start bit of 0 and stop bit of 1.
SHIFTCTLn	0x0080_0001	Configure receive using Timer 0 on negedge of clock with input data on Pin 0. Can invert input data by setting PINPOL.
TIMCMPn	0x0000_0F01	Configure 8-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0204_2422	Configure start bit, stop bit, enable on pin posedge and disable on compare. Enable resynchronization to received data with TIMOUT=0x2 and TIMRST=0x4.
TIMCTLn	0x0000_0081	Configure dual 8-bit counter using inverted Pin 0 input.
SHIFTBUFn	Data to receive	Received data can be read from SHIFTBUFBYS[7:0], use the Shifter Status Flag to indicate when data can be

Table 51-6. UART Receiver Configuration

Register	Value	Comments
		read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS[7:0] register instead.

The UART Receiver with RTS configuration uses a 2nd Timer to generate the RTS output. The RTS will assert when the start bit is detected and negate when the data is read from the shifter buffer register. No start bit will be detected while the RTS is asserted, the received data is simply ignored.

Table 51-7. UART Receiver with RTS Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0032	Configure start bit of 0 and stop bit of 1.
SHIFTCTLn	0x0080_0001	Configure receive using Timer 0 on negedge of clock with input data on Pin 0. Can invert input data by setting PINPOL.
TIMCMPn	0x0000_0F01	Configure 8-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0204_2522	Configure start bit, stop bit, enable on pin posedge with trigger low and disable on compare. Enable resynchronization to received data with TIMOUT=0x2 and TIMRST=0x4.
TIMCTLn	0x03C0_0081	Configure dual 8-bit counter using inverted Pin 0 input. Trigger is internal using inverted Pin 1 input.
TIMCMP(n+1)	0x0000_FFFF	Never compare.
TIMCFG(n+1)	0x0030_6100	Enable on Timer N enable and disable on trigger falling edge. Decrement on trigger to ensure no compare.
TIMCTL(n+1)	0x0143_0083	Configure 16-bit counter and output on Pin 1. Trigger is internal using Shifter 0 flag.
SHIFTBUFn	Data to receive	Received data can be read from SHIFTBUFBYS[7:0], use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS[7:0] register instead.

51.5.3 SPI Master

SPI master mode can be supported using two Timers, two Shifters and four Pins. Either CPHA=0 or CPHA=1 can be supported and transfers can be supported using the DMA controller. For CPHA=1, the select can remain asserted for multiple transfers and the timer status flag can be used to indicate the end of the transfer.

The stop bit is used to guarantee a minimum of 1 clock cycle between the slave select negating and before the next transfer. Writing to the transmit buffer by either core or DMA is used to initiate each transfer.

Due to synchronization delays, the setup time for the serial input data is 1.5 FlexIO clock cycles, so the maximum baud rate is divide by 4 of the FlexIO clock frequency.

Table 51-8. SPI Master (CPHA=0) Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0000	Start and stop bit disabled.
SHIFTCTLn	0x0083_0002	Configure transmit using Timer 0 on negedge of clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0000_0101	Configure receive using Timer 0 on posedge of clock with input data on Pin 1.
TIMCMPn	0x0000_3F01	Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0100_2222	Configure start bit, stop bit, enable on trigger high and disable on compare, initial clock state is logic 0. Set PINPOL to invert the output shift clock.
TIMCTLn	0x01C3_0201	Configure dual 8-bit counter using Pin 2 output (shift clock), with Shifter 0 flag as the inverted trigger.
TIMCMP(n+1)	0x0000_FFFF	Never compare.
TIMCFG(n+1)	0x0000_1100	Enable when Timer 0 is enabled and disable when Timer 0 is disabled.
TIMCTL(n+1)	0x0003_0383	Configure 16-bit counter (never compare) using inverted Pin 3 output (as slave select).
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can

Table continues on the next page...

Table 51-8. SPI Master (CPHA=0) Configuration (continued)

Register	Value	Comments
		support MSB first transfer by writing to SHIFTBUFBBS register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUFBYS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS register instead.

Table 51-9. SPI Master (CPHA=1) Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0021	Start bit loads data on first shift.
SHIFTCTLn	0x0003_0002	Configure transmit using Timer 0 on posedge of clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0080_0101	Configure receive using Timer 0 on negedge of clock with input data on Pin 1.
TIMCMPn	0x0000_3F01	Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0100_2222	Configure start bit, stop bit, enable on trigger high and disable on compare, initial clock state is logic 0. Set PINPOL to invert the output shift clock. Set TIMDIS=3 to keep slave select asserted for as long as there is data in the transmit buffer.
TIMCTLn	0x01C3_0201	Configure dual 8-bit counter using Pin 2 output (shift clock), with Shifter 0 flag as the inverted trigger.
TIMCMP(n+1)	0x0000_FFFF	Never compare.
TIMCFG(n+1)	0x0000_1100	Enable when Timer 0 is enabled and disable when Timer 0 is disabled.
TIMCTL(n+1)	0x0003_0383	Configure 16-bit counter (never compare) using inverted Pin 3 output (as slave select).
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBBS register instead.

Table continues on the next page...

Table 51-9. SPI Master (CPHA=1) Configuration (continued)

Register	Value	Comments
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUFBYS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS register instead.

51.5.4 SPI Slave

SPI slave mode can be supported using one Timer, two Shifters and four Pins. Either CPHA=0 or CPHA=1 can be supported and transfers can be supported using the DMA controller. For CPHA=1, the select can remain asserted for multiple transfers and the timer status flag can be used to indicate the end of the transfer.

The transmit data must be written to the transmit buffer register before the external slave select asserts, otherwise the shifter error flag will be set.

Due to synchronization delays, the output valid time for the serial output data is 2.5 FlexIO clock cycles, so the maximum baud rate is divide by 6 of the FlexIO clock frequency.

Table 51-10. SPI Slave (CPHA=0) Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0000	Start and stop bit disabled.
SHIFTCTLn	0x0083_0002	Configure transmit using Timer 0 on falling edge of shift clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0000_0101	Configure receive using Timer 0 on rising edge of shift clock with input data on Pin 1.
TIMCMPn	0x0000_003F	Configure 32-bit transfer. Set TIMCMP[15:0] = (number of bits x 2) - 1.
TIMCFGn	0x0120_6600	Configure enable on trigger rising edge and disable on trigger falling edge, initial clock state is logic 0 and decrement on pin input.
TIMCTLn	0x06C0_0203	Configure 16-bit counter using Pin 2 input (shift clock), with Pin 3 input (slave select) as the inverted trigger.
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written

Table continues on the next page...

Table 51-10. SPI Slave (CPHA=0) Configuration (continued)

Register	Value	Comments
		using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBBS register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUFBYS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS register instead.

Table 51-11. SPI Slave (CPHA=1) Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0001	Shifter configured to load on first shift and stop bit disabled.
SHIFTCTLn	0x0003_0002	Configure transmit using Timer 0 on rising edge of shift clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0080_0101	Configure receive using Timer 0 on falling edge of shift clock with input data on Pin 1.
TIMCMPn	0x0000_003F	Configure 32-bit transfer. Set TIMCMP[15:0] = (number of bits x 2) - 1.
TIMCFGn	0x0120_6602	Configure start bit, enable on trigger rising edge, disable on trigger falling edge, initial clock state is logic 0 and decrement on pin input.
TIMCTLn	0x06C0_0203	Configure 16-bit counter using Pin 2 input (shift clock), with Pin 3 input (slave select) as the inverted trigger.
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBBS register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUFBYS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS register instead.

51.5.5 I2C Master

I2C master mode can be supported using two Timers, two Shifters and two Pins. One timer is used to generate the SCL output and one timer is used to control the shifters. The two shifters are used to transmit and receive for every word, when receiving the transmitter must transmit 0xFF to tristate the output. FlexIO inserts a stop bit after every word to generate/verify the ACK/NACK. FlexIO waits for the first write to the transmit data buffer before enabling SCL generation. Data transfers can be supported using the DMA controller and the shifter error flag will set on transmit underrun or receive overflow.

The first timer generates the bit clock for the entire packet (START to Repeated START/ STOP), so the compare register needs to be programmed with the total number of clock edges in the packet (minus one). The timer supports clock stretching using the reset counter when pin equal to output (although this increases both the clock high and clock low periods by at least 1 FlexIO clock cycle each). The second timer uses the SCL input pin to control the transmit/receive shift registers, this enforces an SDA data hold time by an extra 2 FlexIO clock cycles.

Both the transmit and receive shifters need to be serviced for each word in the transfer, the transmit shifter must transmit 0xFF when receiving and the receive shifter returns the data actually present on the SDA pin. The transmit shifter will load 1 additional word on the last falling edge of SCL pin, this word should be 0x00 if generating a STOP condition or 0xFF if generating a repeated START condition. During the last word of a master-receiver transfer, the transmit SSTOP bit should be set by software to generate a NACK.

The receive shift register will assert an error interrupt if a NACK is detected, but software is responsible for generating the STOP or repeated START condition. If a NACK is detected during master-transmit, the interrupt routine should immediately write the transmit shifter register with 0x00 (if generating STOP) or 0xFF (if generating repeated START). Software should then wait for the next rising edge on SCL and then disable both timers. The transmit shifter should then be disabled after waiting the setup delay for a repeated START or STOP condition.

Due to synchronization delays, the data valid time for the transmit output is 2 FlexIO clock cycles, so the maximum baud rate is divide by 6 of the FlexIO clock frequency.

The I2C master data valid is delayed 2 cycles because the clock output is passed through a synchronizer before clocking the transmit/receive shifter (to guarantee some SDA hold time). Since the SCL output is synchronous with FlexIO clock, the synchronization delay is 1 cycle and then 1 cycle to generate the output.

Table 51-12. I2C Master Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0032	Start bit enabled (logic 0) and stop bit enabled (logic 1).
SHIFTCTLn	0x0101_0082	Configure transmit using Timer 1 on rising edge of clock with inverted output enable (open drain output) on Pin 0.
SHIFTCFG(n+1)	0x0000_0020	Start bit disabled and stop bit enabled (logic 0) for ACK/NACK detection.
SHIFTCTL(n+1)	0x0180_0001	Configure receive using Timer 1 on falling edge of clock with input data on Pin 0.
TIMCMPn	0x0000_2501	Configure 2 word transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of words x 18) + 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0102_2222	Configure start bit, stop bit, enable on trigger high, disable on compare, reset if output equals pin. Initial clock state is logic 0 and is not affected by reset.
TIMCTLn	0x01C1_0101	Configure dual 8-bit counter using Pin 1 output enable (SCL open drain), with Shifter 0 flag as the inverted trigger.
TIMCMP(n+1)	0x0000_000F	Configure 8-bit transfer. Set TIMCMP[15:0] = (number of bits x 2) - 1.
TIMCFG(n+1)	0x0020_1112	Enable when Timer 0 is enabled, disable when Timer 0 is disabled, enable start bit and stop bit at end of each word, decrement on pin input.
TIMCTL(n+1)	0x01C0_0183	Configure 16-bit counter using inverted Pin 1 input (SCL).
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUFBBS[7:0], use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUFBIS[7:0], use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request.

51.5.6 I2S Master

I2S master mode can be supported using two Timers, two Shifters and four Pins. One timer is used to generate the bit clock and control the shifters and one timer is used to generate the frame sync. FlexIO waits for the first write to the transmit data buffer before

enabling bit clock and frame sync generation. Data transfers can be supported using the DMA controller and the shifter error flag will set on transmit underrun or receive overflow.

The bit clock frequency is an even integer divide of the FlexIO clock frequency, and the initial frame sync assertion occurs at the same time as the first bit clock edge. The timer uses the start bit to ensure the frame sync is generated one clock cycle before the first output data.

Due to synchronization delays, the setup time for the receiver input is 1.5 FlexIO clock cycles, so the maximum baud rate is divide by 4 of the FlexIO clock frequency.

Table 51-13. I2S Master Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0001	Load transmit data on first shift and stop bit disabled.
SHIFTCTLn	0x0003_0002	Configure transmit using Timer 0 on rising edge of clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0080_0101	Configure receive using Timer 0 on falling edge of clock with input data on Pin 1.
TIMCMPn	0x0000_3F01	Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0000_0202	Configure start bit, enable on trigger high and never disable. Initial clock state is logic 1.
TIMCTLn	0x01C3_0201	Configure dual 8-bit counter using Pin 2 output (bit clock), with Shifter 0 flag as the inverted trigger. Set PINPOL to invert the output shift clock.
TIMCMP(n+1)	0x0000_007F	Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:0] = (number of bits x baud rate divider) - 1.
TIMCFG(n+1)	0x0000_0100	Enable when Timer 0 is enabled and never disable.
TIMCTL(n+1)	0x0003_0383	Configure 16-bit counter using inverted Pin 3 output (as frame sync).
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUFBIS, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support LSB first transfer by writing to SHIFTBUF register instead.

Table continues on the next page...

Table 51-13. I2S Master Configuration (continued)

Register	Value	Comments
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUFBIS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support LSB first transfer by reading from SHIFTBUF register instead.

51.5.7 I2S Slave

I2S slave mode can be supported using two Timers, two Shifters and four Pins (for single transmit and single receive, other combinations of transmit and receive are possible).

The transmit data must be written to the transmit buffer register before the external frame sync asserts, otherwise the shifter error flag will be set.

Due to synchronization delays, the output valid time for the serial output data is 2.5 FlexIO clock cycles, so the maximum baud rate is divide by 6 of the FlexIO clock frequency.

The output valid time of I2S slave is max 2.5 cycles because there is a maximum 1.5 cycle delay on the clock synchronization plus 1 cycle to output the data

Table 51-14. I2S Slave Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0000	Start and stop bit disabled.
SHIFTCTLn	0x0103_0002	Configure transmit using Timer 1 on rising edge of shift clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0180_0101	Configure receive using Timer 1 on falling edge of shift clock with input data on Pin 1.
TIMCMPn	0x0000_007D	Configure two 32-bit transfers per frame. Set TIMCMP[15:0] = (number of bits x 4) - 3.
TIMCFGn	0x0030_2400	Configure enable on pin rising edge (inverted frame sync) and disable on compare, initial clock state is logic 1 and decrement on trigger input (bit clock).
TIMCTLn	0x0440_0383	Configure 16-bit counter using inverted Pin 3 input (frame sync), with Pin 2 input (bit clock) as the trigger.

Table continues on the next page...

Table 51-14. I2S Slave Configuration (continued)

Register	Value	Comments
TIMCMP(n+1)	0x0000_003F	Configure 32-bit transfers. Set TIMCMP[15:0] = (number of bits x 2) - 1.
TIMCFG(n+1)	0x0020_3500	Configure enable on pin rising edge with trigger high and disable on compare with trigger low, initial clock state is logic 0 and decrement on pin input.
TIMCTL(n+1)	0x0340_0203	Configure 16-bit counter using Pin 2 input (bit clock), with Timer 0 output as the trigger.
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUFBIS, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support LSB first transfer by writing to SHIFTBUF register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUFBIS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support LSB first transfer by reading from SHIFTBUF register instead.

51.5.8 Camera Interface

Camera Interface can be supported using one Timer, one or more Shifters and multiple Pins. Multiple transfers can be supported using DMA controller.

The example below describes FlexIO configuration for interfacing to an 8-bit CMOS sensor with PCLK, VSYNC, HREF and D[7:0] outputs. The example uses a 128-bit buffer to capture 16-pixels of image data before interrupt or DMA transfer, however a bigger or smaller buffer may be used depending on system DMA performance and FlexIO resource usage by other applications. Note that additional timers may be used to track number of pixels per row and number of rows per frame or HREF/VSYNC may be assigned as GPIO interrupts for software tracking.

Table 51-15. Camera Interface Configuration for 8-bit CMOS sensor

Register	Value	Comments
SHIFTCFGn...n+2[1]	0x0007_0100	Configure 8-bit parallel shift in from adjacent shifter.
SHIFTCFGn+3	0x0007_0000	Configure 8-bit parallel shift in from pins FXIO_D[7:0] (D[7:0]).
SHIFTCTLn...n+3	0x0080_0001	Configure receive using Timer 0 on negedge of clock.

Table continues on the next page...

Table 51-15. Camera Interface Configuration for 8-bit CMOS sensor (continued)

Register	Value	Comments
TIMCMPn	0x0000_001F	Configure 16-pixel (8 bits/pixel x 16 pixels = 128-bits) transfer. Set TIMCMP[15:0] = (number of pixels x 2) - 1.
TIMCFGn	0x0120_6600	Configure enable on trigger (HREF) rising edge and disable on trigger falling edge, initial Shift clock state is logic 0 and decrement on PCLK input.
TIMCTLn	0x12C0_0803	Configure 16-bit counter using FXIO_D[8] input (PCLK), with FXIO_D[9] input (HREF) as the inverted trigger.
SHIFTBUFn...n+3	Data to receive	Received data can be read from SHIFTBUFn...n+3, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request.

1. n=0 or 4

51.5.9 Motorola 68K/Intel 8080 Bus Interface

The Motorola 68K and Intel 8080 bus are two parallel interfaces commonly used by Smart/Asynchronous LCD controllers. In conjunction with GPIO, FlexIO is able to drive these interfaces using one Timer and one Shifter, although additional Shifters could be used to support large transfers via the DMA controller.

The configuration below provides an example of how to drive a 16-bit 68K or 8080 bus. For a 8080 bus, two GPIO are used to drive the nCS and RS pins. For a 68K bus, an additional GPIO is required to drive the RDWR pin.

Table 51-16. Motorola 68K/Intel 8080 Write Configuration

Register	Value	Comments
SHIFTCFG0...7	0x000F_0100	Configure 16-bit parallel shift in from adjacent shifter.
SHIFTCTL0	0x0003_0002	Configure transmit using Timer 0 on posedge of clock with data output to FXIO_D[15:0].
SHIFTCTL1...7	0x0000_0002	Configure transmit using Timer 0 on posedge of clock.
TIMCMP0	0x0000_0001 (1-beat) 0x0000_1F01 (16-beats)	Configure 1 or 16-beat transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of beats x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.

Table continues on the next page...

Table 51-16. Motorola 68K/Intel 8080 Write Configuration (continued)

Register	Value	Comments
TIMCFG0	0x0000_2200	Configure enable on trigger high and disable on compare. Timer output high on enable.
TIMCTL0	0x01C3_1001 (1-beat) 0x1DC3_1001 (16-beats)	Configure dual 8-bit counter using FXIO_D[16] output (EN pin for 68K, WR pin for 8080), with Shifter 0 (1-beat) or Shifter 7 (16-beats) flag as the inverted trigger.
SHIFTBUF0...7	Data to transmit	Transmit data can be written to SHIFTBUF0 (1-beat) or SHIFTBUF0...7 (16-beats) to initiate a transfer, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request.

Table 51-17. Motorola 68K/Intel 8080 Read Configuration

Register	Value	Comments
SHIFTCFG0...6	0x000F_0100	Configure 16-bit parallel shift in from adjacent shifter.
SHIFTCFG7	0x000F_0000	Configure 16-bit parallel shift in from pin.
SHIFTCTL0...7	0x0080_0001	Configure receive using Timer 0 on negedge of clock with data input from FXIO_D[15:0].
TIMCMP0	0x0000_0001 (1-beat) 0x0000_1F01 (16-beats)	Configure 1 or 16-beat transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of beats x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFG0	0x0000_2200	Configure enable on trigger high and disable on compare. Timer output high on enable.
TIMCTL0	0x01C3_1101 (Intel 8080) 0x01C3_1001 (Motorola 68K)	Configure dual 8-bit counter using either FXIO_D[16] output (EN pin for 68K) or FXIO_D[17] output (RD pin for 8080), with Shifter 0 flag as the inverted trigger.
SHIFTBUF0...7	Data received	Received data can be read from SHIFTBUF0 (1-beat) or SHIFTBUF0...7 (16-beats), use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request.

In general, any operation to a 68K/8080 bus slave will begin with a register write cycle followed by one or more data read or write cycles. To accomplish this, the following program flow should be used:

1. Configure FlexIO with 1-beat write configuration
2. Configure GPIO to assert nCS, RS pins (and deassert RDWR pin for 68K)

3. Write register index data to SHIFTBUF0[15:0]
4. Configure GPIO to deassert RS pin (and assert RDWR pin for 68K data read)
5. Configure FlexIO with desired read or write configuration (e.g. 1 or 16-beats)
6. Use the Shifter Status Flag to trigger interrupt or DMA driven data transfers to/from SHIFTBUF registers
7. Configure GPIO to deassert nCS pin

51.5.10 State Machine Example

The configuration below details a hypothetical state machine example to illustrate the flexibility allowed when using Shifter state mode. In this example, FlexIO waits for the FXIO_D[2] pin to assert and then drives a complementary square wave output at a frequency of FLEXIO_CLK/131072 on the FXIO_D[1:0] pins while the comparator output is asserted (assumes comparator is connected to external trigger 15). The state diagram below shows the states and transitions implemented by this example.

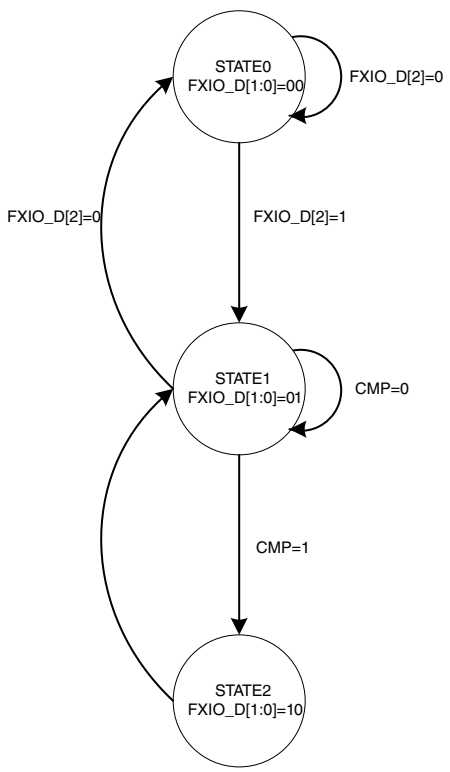


Figure 51-5. Example State Diagram

Table 51-18. State Machine Configuration

Register	Value	Comments
SHIFTCFG0...2	0x0000_0003	Enable FXIO_D[1:0] as outputs.
SHIFTCTL0	0x0080_0206	Configure for State mode using FXIO_D[4:2] as inputs to select next state and Timer0 output low to trigger next state.
SHIFTBUF0	0x0020_8208	State0: Drive FXIO_D[1:0]=00, transition to State0 if FXIO_D[2]=0, State1 if FXIO_D[2]=1.
SHIFTCTL1	0x0000_0206	Configure for State mode using FXIO_D[4:2] as inputs to select next state and Timer0 output high to trigger next state.
SHIFTBUF1	0x0140_8408	State1: Drive FXIO_D[1:0]=01, transition to State0 if FXIO_D[2]=0, State1 if CMP=0, State2 if CMP=1 (FXIO_D[3]=1)
SHIFTCTL2	0x0080_0206	Configure for State mode using FXIO_D[4:2] as inputs to select next state and Timer0 output low to trigger next state.
SHIFTBUF2	0x0224_9249	State2: Drive FXIO_D[1:0]=10, transition to State1 when Timer0 output low
TIMCMP0	0x0000_FFFF	Configure baud rate of divide by 131072 of the FlexIO clock. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFG0	0x0000_0000	Configure timer always enabled.
TIMCTL0	0x0000_0003	Configure single 16-bit counter.
TIMCFG1	0x0010_7600	Configure timer enabled on trigger rising edge, disabled on trigger falling edge, decrement on trigger edge.
TIMCTL1	0x0F03_0303	Configure timer output enabled on FXIO_D[3], external trigger 15 (comparator output) selected.

Chapter 52
Touch Sensing Input (TSI)

52.1 Chip-specific TSI information

52.1.1 Number of inputs

This device includes one TSI module containing up to 16 inputs. In low-power modes, one selectable pin is active.

52.1.2 TSI module functionality in MCU operation modes

Table 52-1. TSI module functionality in MCU operation modes

MCU operation mode	TSI clock sources	TSI operation mode when GENCS[TSIEN] is 1	Functional electrode pins	Required GENCS[STPE] state
Run	BUSCLK, MCGIRCLK, OSCERCLK	Active mode	All	Don't care
Wait	BUSCLK, MCGIRCLK, OSCERCLK	Active mode	All	Don't care
Stop	MCGIRCLK, OSCERCLK	Active mode	All	1
VLPR	BUSCLK, MCGIRCLK, OSCERCLK	Active mode	All	Don't care
VLPW	BUSCLK, MCGIRCLK, OSCERCLK	Active mode	All	Don't care
VLPS	OSCERCLK	Active mode	All	1
LLS	LPOCLK, VLPOSCCLK	Low power mode	Determined by PEN[LPSP]	1
VLLS3	LPOCLK, VLPOSCCLK	Low power mode	Determined by PEN[LPSP]	1
VLLS2	LPOCLK, VLPOSCCLK	Low power mode	Determined by PEN[LPSP]	1

Table continues on the next page...

Table 52-1. TSI module functionality in MCU operation modes (continued)

MCU operation mode	TSI clock sources	TSI operation mode when GENCS[TSIEN] is 1	Functional electrode pins	Required GENCS[STPE] state
VLLS1	LPOCLK, VLPOSCCLK	Low power mode	Determined by PEN[LPSP]	1
VLLS0	VLPOSCCLK[1]	Low power mode	Determined by PEN[LPSP]	1

1. This clock must be 32 kHz RTC.

52.1.3 TSI clocks

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

Table 52-2. TSI clock connections

Module clock	Chip clock
BUSCLK	Bus clock
MCGIRCLK	MCGIRCLK
OSCERCLK	OSCERCLK
LPOCLK	1 kHz LPO clock
VLPOSCCLK	ERCLK32K

52.1.4 TSI Interrupts

The TSI has multiple sources of interrupt requests. However, these sources are OR'd together to generate a single interrupt request. When a TSI interrupt occurs, read the TSI status register to determine the exact interrupt source.

52.1.5 Shield drive signal

The shield drive signal is not supported on this device. Ignore this feature in the TSI chapter.

52.2 Introduction

The touch sensing input (TSI) module provides capacitive touch sensing detection with high sensitivity and enhanced robustness.

Each TSI pin implements the capacitive measurement by a current source scan, charging and discharging the electrode, once or several times. A reference oscillator ticks the scan time and stores the result in a 16-bit register when the scan completes. Meanwhile, an interrupt request is submitted to CPU for post-processing if TSI interrupt is enabled and DMA function is not selected.The TSI module can be periodically triggered to work in low power mode with ultra-low current adder and wake CPU at the end of scan or the conversion result is out of the range specified by TSI threshold. It provides a solid capacitive measurement module to the implementation of touch keyboard, rotaries and sliders.

52.2.1 Features

TSI features includes:

- Support up to 16 external electrodes
- Automatic detection of electrode capacitance across all operational power modes
- Internal reference oscillator for high-accuracy measurement
- Configurable software or hardware scan trigger
- Fully support NXP touch sensing software (TSS) library, see www.nxp.com/touchsensing.
- Capability to wake MCU from low power modes
- Compensate for temperature and supply voltage variations
- High sensitivity change with 16-bit resolution register
- Configurable up to 4096 scan times.
- Support DMA data transfer

For electrode design recommendations, refer to AN3863: Designing Touch Sensing Electrodes

52.2.2 Modes of operation

This module supports the following operation modes.

Table 52-3. Operating modes

Mode	Description
Stop and low power stop	TSI module is fully functional in all of the stop modes as long as TSI_GENCS[STPE] is set. The channel specified by TSI_DATA[TSICH] will be scanned upon the trigger. After scan finishes, either end-of-scan or out-of-range interrupt can be selected to bring MCU out of low power modes.

Table continues on the next page...

Table 52-3. Operating modes (continued)

Mode	Description
Wait	TSI module is fully functional in this mode. When a scan completes, TSI submits an interrupt request to CPU if the interrupt is enabled.
Run	TSI module is fully functional in this mode. When a scan completes, TSI submits an interrupt request to CPU if the interrupt is enabled.

52.2.3 Block diagram

The following figure is a block diagram of the TSI module.

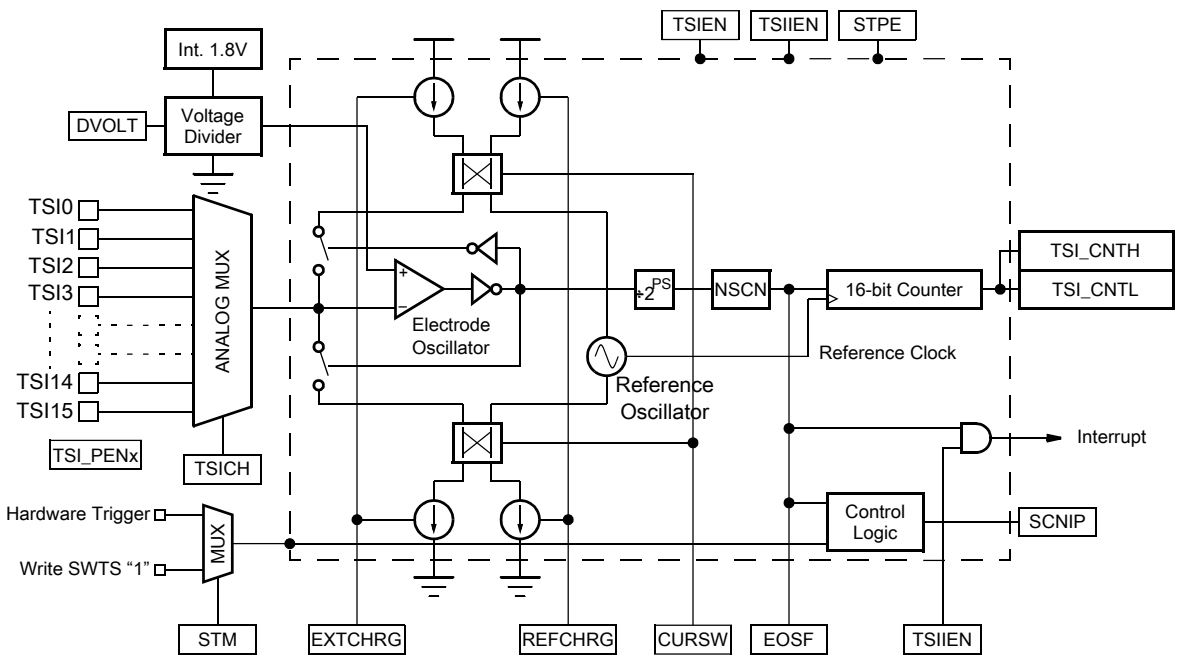


Figure 52-1. TSI module block diagram

52.3 External signal description

The TSI module contains up to 16 external pins for touch sensing. The table found here describes each of the TSI external pins.

Table 52-4. TSI signal description

Name	Port	Direction	Function	Reset state
TSI[15:0]	TSI	I/O	TSI capacitive pins. Switches driver that	I/O

Table 52-4. TSI signal description

Name	Port	Direction	Function	Reset state
			connects directly to the electrode pins TSI[15:0] can operate as GPIO pins.	

52.3.1 TSI[15:0]

When TSI functionality is enabled , the TSI analog portion uses the corresponding channel to connect external on-board touch capacitors. The PCB connection between the pin and the touch pad must be kept as short as possible to reduce distribution capacity on board.

52.4 Register definition

This section describes the memory map and control/status registers for the TSI module.

TSI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_5000	TSI General Control and Status Register (TSI0_GENCS)	32	R/W	0000_0000h	52.4.1/1573
4004_5004	TSI DATA Register (TSI0_DATA)	32	R/W	0000_0000h	52.4.2/1578
4004_5008	TSI Threshold Register (TSI0_TSHD)	32	R/W	0000_0000h	52.4.3/1579

52.4.1 TSI General Control and Status Register (TSIx_GENCS)

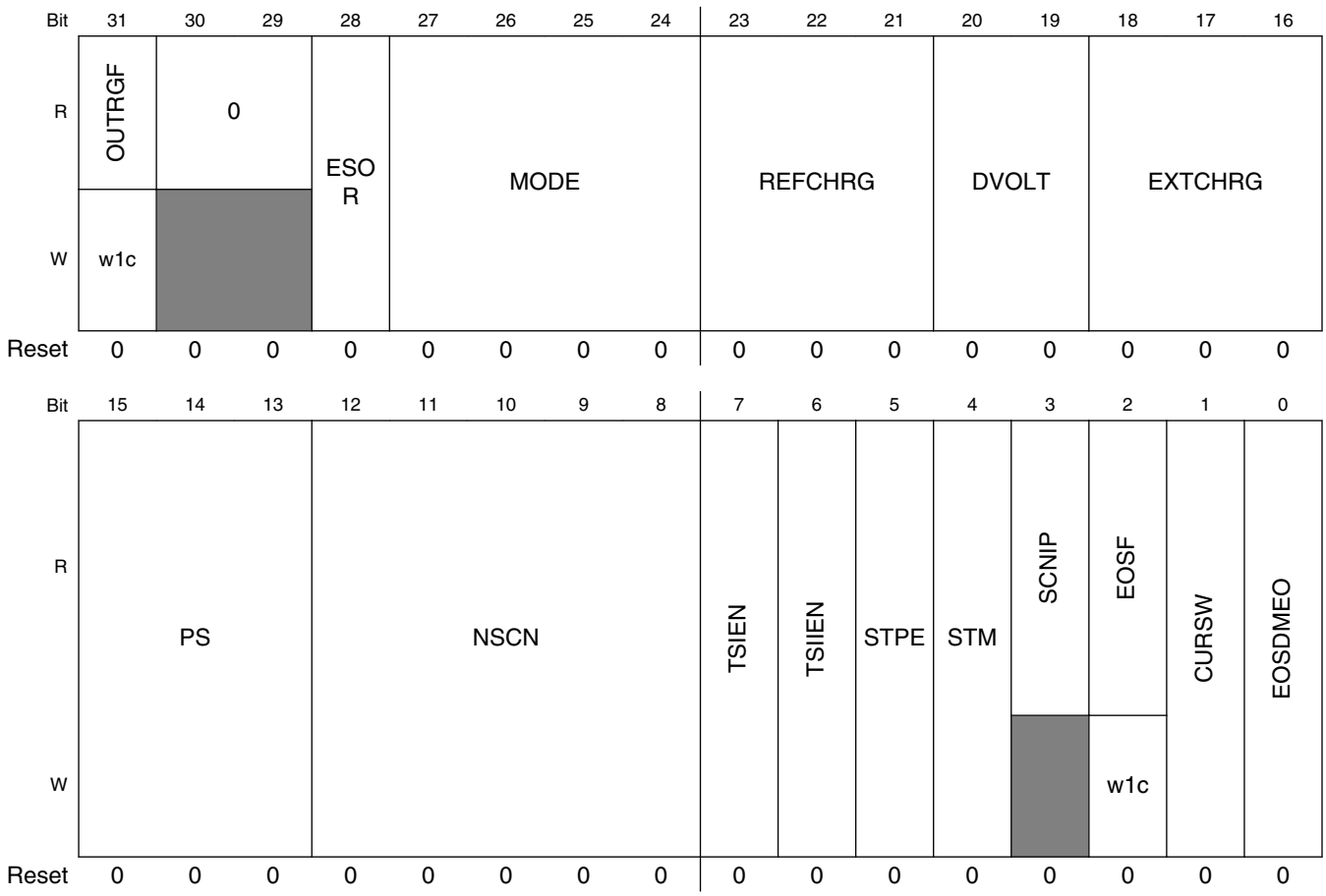
This control register provides various control and configuration information for the TSI module.

NOTE

When TSI is working, the configuration bits (GENCS[TSIEN], GENCS[TSIIEN], and GENCS[STM]) must not be changed.
The EOSF flag is kept until the software acknowledge it.

Register definition

Address: 4004_5000h base + 0h offset = 4004_5000h



TSIx_GENCS field descriptions

Field	Description
31 OUTRGF	Out of Range Flag. This flag is set if the result register of the enabled electrode is out of the range defined by the TSI_THRESHOLD register. This flag is set only when TSI is configured in non-noise detection mode. It can be read once the CPU wakes. Write "1" , when this flag is set, to clear it.
30–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 ESOR	End-of-scan or Out-of-Range Interrupt Selection This bit is used to select out-of-range or end-of-scan event to generate an interrupt. 0 Out-of-range interrupt is allowed. 1 End-of-scan interrupt is allowed.
27–24 MODE	TSI analog modes setup and status bits. Set up TSI analog modes, especially, setting MODE[3:2] to not 2'b00 will configure TSI to noise detection modes. MODE[1:0] take no effect on TSI operation mode and should always write to 2'b00 for setting up. When reading this field will return the analog status. Refer to chapter "Noise detection mode" for details. 0000 Set TSI in capacitive sensing(non-noise detection) mode.

Table continues on the next page...

TSIx_GENCS field descriptions (continued)

Field	Description
	0100 Set TSI analog to work in single threshold noise detection mode and the frequency limitation circuit is disabled. 1000 Set TSI analog to work in single threshold noise detection mode and the frequency limitation circuit is enabled to work in higher frequencies operations. 1100 Set TSI analog to work in automatic noise detection mode.
23–21 REFCHRG	REFCHRG These bits indicate the reference oscillator charge and discharge current value. 000 500 nA. 001 1 µA. 010 2 µA. 011 4 µA. 100 8 µA. 101 16 µA. 110 32 µA. 111 64 µA.
20–19 DVOLT	DVOLT These bits indicate the oscillator's voltage rails as below. 00 DV = 1.026 V; V_P = 1.328 V; V_m = 0.302 V. 01 DV = 0.592 V; V_P = 1.111 V; V_m = 0.519 V. 10 DV = 0.342 V; V_P = 0.986 V; V_m = 0.644 V. 11 DV = 0.197 V; V_P = 0.914 V; V_m = 0.716 V.
18–16 EXTCHRG	EXTCHRG These bits indicate the electrode oscillator charge and discharge current value. 000 500 nA. 001 1 µA. 010 2 µA. 011 4 µA. 100 8 µA. 101 16 µA. 110 32 µA. 111 64 µA.
15–13 PS	PS These bits indicate the prescaler of the output of electrode oscillator. 000 Electrode Oscillator Frequency divided by 1 001 Electrode Oscillator Frequency divided by 2 010 Electrode Oscillator Frequency divided by 4 011 Electrode Oscillator Frequency divided by 8 100 Electrode Oscillator Frequency divided by 16 101 Electrode Oscillator Frequency divided by 32 110 Electrode Oscillator Frequency divided by 64 111 Electrode Oscillator Frequency divided by 128

Table continues on the next page...

TSIx_GENCS field descriptions (continued)

Field	Description
12–8 NSCN	NSCN These bits indicate the scan number for each electrode. The scan number is equal to NSCN + 1, which allows the scan time ranges from 1 to 32. By default, NSCN is configured as 0, which asserts the TSI scans once on the selected eletrode channel. 00000 Once per electrode 00001 Twice per electrode 00010 3 times per electrode 00011 4 times per electrode 00100 5 times per electrode 00101 6 times per electrode 00110 7 times per electrode 00111 8 times per electrode 01000 9 times per electrode 01001 10 times per electrode 01010 11 times per electrode 01011 12 times per electrode 01100 13 times per electrode 01101 14 times per electrode 01110 15 times per electrode 01111 16 times per electrode 10000 17 times per electrode 10001 18 times per electrode 10010 19 times per electrode 10011 20 times per electrode 10100 21 times per electrode 10101 22 times per electrode 10110 23 times per electrode 10111 24 times per electrode 11000 25 times per electrode 11001 26 times per electrode 11010 27 times per electrode 11011 28 times per electrode 11100 29 times per electrode 11101 30 times per electrode 11110 31 times per electrode 11111 32 times per electrode
7 TSIEN	Touch Sensing Input Module Enable This bit enables TSI module. 0 TSI module disabled. 1 TSI module enabled.
6 TSIIEN	Touch Sensing Input Interrupt Enable This bit enables TSI module interrupt request to CPU when the scan completes. The interrupt will wake MCU from low power mode if this interrupt is enabled.

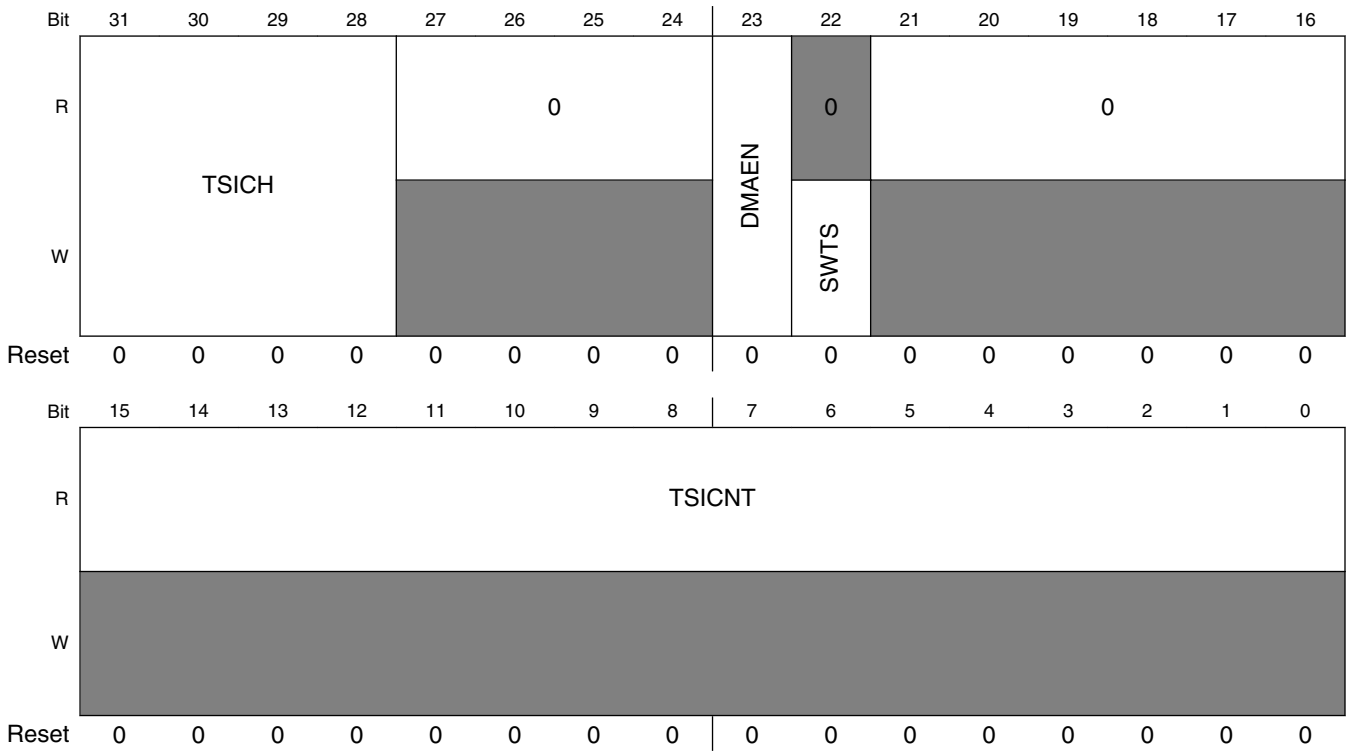
Table continues on the next page...

TSIx_GENCS field descriptions (continued)

Field	Description
	0 TSI interrupt is disabled. 1 TSI interrupt is enabled.
5 STPE	TSI STOP Enable This bit enables TSI module function in low power modes (stop, VLPS, LLS and VLLS{3,2,1}). 0 TSI is disabled when MCU goes into low power mode. 1 Allows TSI to continue running in all low power modes.
4 STM	Scan Trigger Mode This bit specifies the trigger mode. User is allowed to change this bit when TSI is not working in progress. 0 Software trigger scan. 1 Hardware trigger scan.
3 SCNIP	Scan In Progress Status This read-only bit indicates if scan is in progress. This bit will get asserted after the analog bias circuit is stable after a trigger and it changes automatically by the TSI. 0 No scan in progress. 1 Scan in progress.
2 EOSF	End of Scan Flag This flag is set when all active electrodes are finished scanning after a scan trigger. Write "1" , when this flag is set, to clear it. 0 Scan not complete. 1 Scan complete.
1 CURSW	CURSW This bit specifies if the current sources of electrode oscillator and reference oscillator are swapped. 0 The current source pair are not swapped. 1 The current source pair are swapped.
0 EOSDMEO	End-of-Scan DMA Transfer Request Enable Only This bit makes simultaneous DMA request at End-of-Scan and Interrupt at Out-of-Range possible. EOSDMEO has precedence to ESOR when trying to set this bit and ESOR bit. When EOSDMEO = 1, End-of-Scan will generate DMA request and Out-of-Range will generate interrupt. 0 Do not enable the End-of-Scan DMA transfer request only. Depending on ESOR state, either Out-of-Range or End-of-Scan can trigger a DMA transfer request and interrupt. 1 Only the End-of-Scan event can trigger a DMA transfer request. The Out-of-Range event only and always triggers an interrupt if TSIIE is set.

52.4.2 TSI DATA Register (TSIx_DATA)

Address: 4004_5000h base + 4h offset = 4004_5004h



TSIx_DATA field descriptions

Field	Description
31–28 TSICH	TSICH These bits specify current channel to be measured. In hardware trigger mode (TSI_GENCS[STM] = 1), the scan will not start until the hardware trigger occurs. In software trigger mode (TSI_GENCS[STM] = 0), the scan starts immediately when TSI_DATA[SWTS] bit is written by 1. 0000 Channel 0. 0001 Channel 1. 0010 Channel 2. 0011 Channel 3. 0100 Channel 4. 0101 Channel 5. 0110 Channel 6. 0111 Channel 7. 1000 Channel 8. 1001 Channel 9. 1010 Channel 10. 1011 Channel 11. 1100 Channel 12. 1101 Channel 13.

Table continues on the next page...

TSIx_DATA field descriptions (continued)

Field	Description
	1110 Channel 14. 1111 Channel 15.
27–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 DMAEN	DMA Transfer Enabled This bit is used together with the TSI interrupt enable bits(TSIIE, ESOR) to generate a DMA transfer request instead of an interrupt. 0 Interrupt is selected when the interrupt enable bit is set and the corresponding TSI events assert. 1 DMA transfer request is selected when the interrupt enable bit is set and the corresponding TSI events assert.
22 SWTS	Software Trigger Start This write-only bit is a software start trigger. When STM bit is clear, write "1" to this bit will start a scan. The electrode channel to be scanned is determinated by TSI_DATA[TSICH] bits. 0 No effect. 1 Start a scan to determine which channel is specified by TSI_DATA[TSICH].
21–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TSICNT	TSI Conversion Counter Value These read-only bits record the accumulated scan counter value ticked by the reference oscillator.

52.4.3 TSI Threshold Register (TSIx_TSHD)

Address: 4004_5000h base + 8h offset = 4004_5008h

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 W	THRESH																THRESL															
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

TSIx_TSHD field descriptions

Field	Description
31–16 THRESH	TSI Wakeup Channel High-threshold This half-word specifies the high threshold of the wakeup channel.
THRESL	TSI Wakeup Channel Low-threshold This half-word specifies the low threshold of the wakeup channel.

52.5 Functional description

52.5.1 Capacitance measurement

The electrode pin capacitance measurement uses a dual oscillator approach. The frequency of the TSI electrode oscillator depends on the external electrode capacitance and the TSI module configuration. After going to a configurable prescaler, the TSI electrode oscillator signal goes to the input of the module counter. The time for the module counter to reach its module value is measured using the TSI reference oscillator. The measured electrode capacitance is directly proportional to the time.

52.5.1.1 TSI electrode oscillator

The TSI electrode oscillator circuit is illustrated in the following figure. A configurable constant current source is used to charge and discharge the external electrode capacitance. A buffer hysteresis defines the oscillator delta voltage. The delta voltage defines the margin of high and low voltage which are the reference input of the comparator in different time.

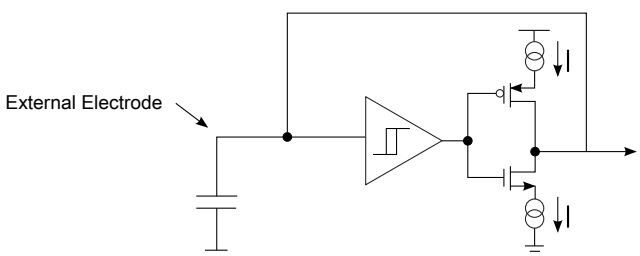


Figure 52-2. TSI electrode oscillator circuit

The current source applied to the pad capacitance is controlled by the GENCS[EXTCHRG]. The hysteresis delta voltage is defined in the module electrical specifications present in the device Data Sheet. The figure below shows the voltage amplitude waveform of the electrode capacitance charging and discharging with a programmable current.

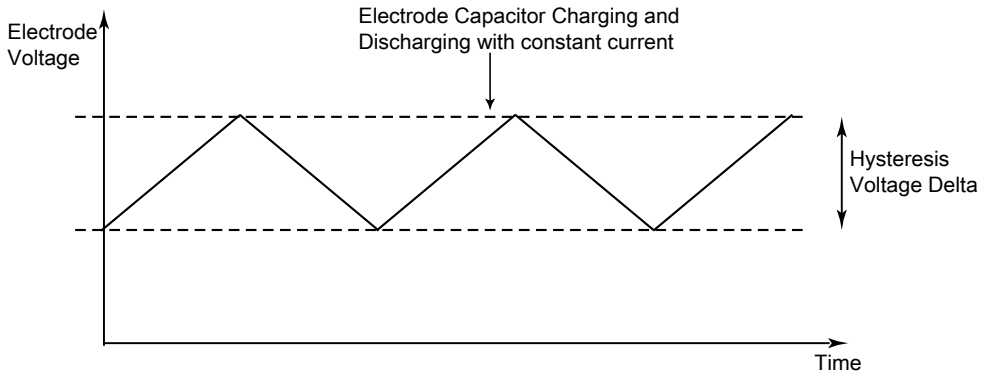


Figure 52-3. TSI electrode oscillator chart

The oscillator frequency is give by the following equation

$$F_{elec} = \frac{I}{2 * C_{elec} * \Delta V}$$

Equation 4. TSI electrode oscillator frequency

Where:

I: constant current

C_{elec}: electrode capacitance

ΔV: Hysteresis delta voltage

So by this equation, for example, an electrode with C_{elec}= 20 pF, with a current source of I = 16 μA and ΔV = 600 m V have the following oscillation frequency:

$$F_{elec} = \frac{16\ \mu A}{2 * 20pF * 600mV} = 0.67MHz$$

Equation 5. TSI electrode oscillator frequency

The current source is used to accommodate the TSI electrode oscillator frequency with different electrode capacitance sizes.

52.5.1.2 Electrode oscillator and counter module control

The TSI oscillator frequency signal goes through a prescaler defined by the GENCS[PS] and then enters in a modulus counter. GENCS[NSCN] defines the maximum count value of the modulus counter.

The pin capacitance sampling time is given by the time the module counter takes to go from 0 to its maximum value, defined by NSCN. The electrode sample time is expressed by the following equation:

$$T_{cap\_samp} = \frac{PS * NSCN}{F_{elec}}$$

Using Equation 1

$$T_{cap\_samp} = \frac{2 * PS * NSCN * C_{elec} * \Delta V}{I}$$

Equation 6. Electrode sampling time

Where:

PS: prescaler value

NSCN: module counter maximum value

I: constant current

C_{elec}: electrode capacitance

ΔV: Hysteresis delta voltage

By this equation we have that an electrode with C = 20 pF, with a current source of I = 16 µA and ΔV = 600 mV, PS = 2 and NSCN = 16 have the following sampling time:

$$T_{cap\_samp} = \frac{2*2*16*20pF*600mV}{16\mu A} = 48\mu s$$

52.5.1.3 TSI reference oscillator

The TSI reference oscillator has the same topology of the TSI electrode oscillator. The TSI reference oscillator instead of using an external capacitor for the electrode oscillator has an internal reference capacitor.

The TSI reference oscillator has an independent programmable current source controlled by GENCS[REFCHRG].

The reference oscillator frequency is given by the following equation:

$$F_{ref\_osc} = \frac{I_{ref}}{2 * C_{ref} * \Delta V}$$

Equation 7. TSI reference oscillator frequency

Where:

C_{ref}: Internal reference capacitor

I_{ref}: Reference oscillator current source

ΔV : Hysteresis delta voltage

Considering C_{ref} = 1.0 pF, I_{ref} = 12 µA and ΔV = 600 mV, follows

$$F_{ref\_osc} = \frac{12\mu A}{2 * 1.0pF * 600mV} = 10.0MHz$$

52.5.2 TSI measurement result

The capacitance measurement result is defined by the number of TSI reference oscillator periods during the sample time and is stored in the TSICHnCNT register.

TSICHnCNT = T_{cap_samp} * F_{ref_osc}

Using Equation 2 and Equation 1 follows:

$$TSICHnCNT = \frac{I_{ref} * PS * NSCN}{C_{ref} * I_{elec}} * C_{elec}$$

Equation 8. Capacitance result value

In the example where F_{ref_osc} = 10.0 MHz and T_{cap_samp} = 48 µs, TSICHnCNT = 480

52.5.3 Enable TSI module

The TSI module can be fully functional in run, wait and low power modes. The TSI_GENCS[TSIEN] bit must be set to enable the TSI module in run and wait mode. When TSI_GENCS[STPE] bit is set, it allows the TSI module to work in low power mode.

52.5.4 Software and hardware trigger

The TSI module allows a software or hardware trigger to start a scan. When a software trigger is applied (TSI_GENCS[STM] bit clear), the TSI_GENCS[SWTS] bit must be written "1" to start the scan electrode channel that is identified by TSI_DATA[TSICH]. When a hardware trigger is applied (TSI_GENCS[STM] bit set), the TSI will not start scanning until the hardware trigger arrives. The hardware trigger is different depending on the MCU configuration. Generally, it could be an event that RTC overflows. See chip configuration section for details.

52.5.5 Scan times

The TSI provides multi-scan function. The number of scans is indicated by TSI_GENCS[NSCN] that allow the scan number from 1 to 32. When TSI_GENCS[NSCN] is set to 0 (only once), the single scan is engaged. The 16-bit counter accumulates all scan results until the NSCN time scan completes, and users can read TSI_DATA[TSICNT] to get this accumulation. When DMA transfer is enabled, the counter values can also be read out by DMA engine.

52.5.6 Clock setting

TSI is built with dual oscillator architecture. In normal sensing application, the reference oscillator clock is the only clock source for operations. The reference clock is used to measure the electrode oscillator by ticking a 16-bit counter. The reference oscillator frequency depends on the current source setting. Please refer to the Current source for more details.

The output of electrode oscillator has several prescalers up to 128 indicated by TSI_GENCS[PS]. This allows a flexible counter configuration for different electrode oscillator frequency.

52.5.7 Reference voltage

The TSI module offers a internal reference voltage for both electrode oscillator and reference oscillator. The internal reference voltage can work in low power modes even when the MCU regulator is partially powered down, which is ideally for low-power touch detection.

The charge and discharge difference voltage is configurable upon the setting of TSI_GENCS[DVOLT]. The following table shows the all the delta voltage configurations.

NOTE

Table 52-5. Delta voltage configuration

DVOLT	V_p (V)	V_m (V)	ΔV (V)
00	1.328	0.302	1.026
01	1.111	0.519	0.592
10	0.986	0.644	0.342
11	0.914	0.716	0.198

52.5.8 Current source

The TSI module supports eight different current source power to increment from 500 nA to 64 μA. TSI_GENCS[EXTCHRG] determines the current of electrode oscillator that charges and discharges external electrodes. The TSI_GENCS[REFCHRG] determines the current of reference oscillator on which the internal reference clock depends. The lower current source takes more time for charge and discharge, which is useful to detect high-accuracy change. The higher current source takes less time, which can be used to charge a big electrode by less power consumption.

TSI_GENCS[CURSW] allows the current source to swap, so that the reference oscillator and electrode oscillator use the opposite current sources. When TSI_GENCS[CURSW] is set and the current sources are swapped, TSI_GENCS[EXTCHRG] and TSI_GENCS[REFCHRG] still control the corresponding current sources, that is, TSI_GENCS[EXTCHRG] controls the reference oscillator current and TSI_GENCS[REFCHRG] controls the electrode oscillator current.

52.5.9 End of scan

As a scan starts, [SCNIP] bit is set to indicate scan is in progress. When the scan completes, the [EOSF] bit is set. Before clearing the [EOSF] bit, the value in TSI_DATA[TSICNT] must be read. If the TSI_GENCS[TSIIEN] and TSI_GENCS[ESOR] are set andTSI_GENCS[DMAEN]is not set, an interrupt is submitted to CPU for post-processing immediately. The interrupt is also optional to wake MCU to execute ISR if it is in low power mode. When DMA function is enabled by setting TSI_GENCS[TSIIEN] and TSI_GENCS[ESOR], as soon as scan completes, a DMA transfer request is asserted to DMA controller for data movement, generally, DMA engine will fetch TSI conversion result from TSI_DATA register,store it to other memory space and then refresh the TSI scan channel index(TSI_DATA[TSICH]) for next loop. When DMA transfer is done, TSI_GENCS[EOSF] is cleared automatically.

52.5.10 Out-of-range interrupt

If enabled, TSI will scan the electrode specified by TSI_DATA[TSICH] as soon as the trigger arrives. The TSI_GENCS[OUTRGF] flag generates a TSI interrupt request if the TSI_GENCS[TSIIE] bit is set and GENCS[ESOR] bit is cleared. With this configuration, after the end-of-electrode scan, the electrode capacitance will be converted and stored to the result register TSI_DATA[TSICNT], the out-of-range interrupt is only requested if

there is a considerable capacitance change defined by the TSI_TSHD. For instance, if in low power mode the electrode capacitance does not vary, the out-of-range interrupt does not interrupt the CPU. This interrupt will not happen in noise detection mode. It is worthy to note that when the counter value reaches 0xFFFF is treated as an extreme case the out-of-range will not happen. Also in noise detection mode, the out-of-range will not assert either.

52.5.11 Wake up MCU from low power modes

In low power modes, once enabled by TSI_GENCS[STPE] and TSI_GENCS[TSIIE], TSI can bring MCU out of its low power modes(STOP, VLPS, VLLS,etc) by either end of scan or out of range interrupt, that is, if TSI_GENCS[ESOR] is set, end of scan interrupt is selected and otherwise, out of range is selected.

52.5.12 DMA function support

Transmit by DMA is supported only when TSI_DATA[DMAEN] is set. A DMA transfer request is asserted when all the flags based on TSI_GENCS[ESOR] settings and TSI_GENCS[TSIIE] are set. Then the on-chip DMA controller detects this request and transfers data between memory space and TSI register space. After the data tranfer, DMA DONE is asserted to clear TSI_GENCS[EOSF] automatically. This function is normally used by DMA controller to get the conversion result from TSI_DATA[TSICNT] upon a end-of-scan event and then refresh the channel index(TSI_DATA[TSICH]) for next trigger.

52.5.13 Noise detection mode

The noise detection mode is used to detect power of noise. In this mode the thresholds are incremented internally by TSI until the point that there is no noise voltage trepassing the threshold.

The noise detection mode change the circuit configuration as shown in the following figure. With this configuration, it is possible to detect touch with high levels of EMC noise present. To enter this mode, set GENCS[MODE] field to 1100b.

In noise detection mode the reference oscillator has the same configuration except the output goes to Counter2 and this counter will have its maximum count set by NSCNx2$^{(PS)}$. This means this oscillator will setup the noise detection mode sense duration as shown in Figure 52-4.

The blocks of external oscillator is changed and instead of an oscillator the circuit implements an RF amplitude detection. The threshold for this amplitude detection is set by DVOLT register bits. Be noted There is no oscillation on external pad (just if it is caused by external noise) in this mode.

Also the external voltage is biased by vmid voltage with a Rs series resistance.

The vmid voltage is defined as V(vmid) = (V(vp) + V(vm))/2.

The Rs value is defined by GENCS[EXTCHRG] register bits. See Figure 52-5 for more information on noise mode TSI circuit.

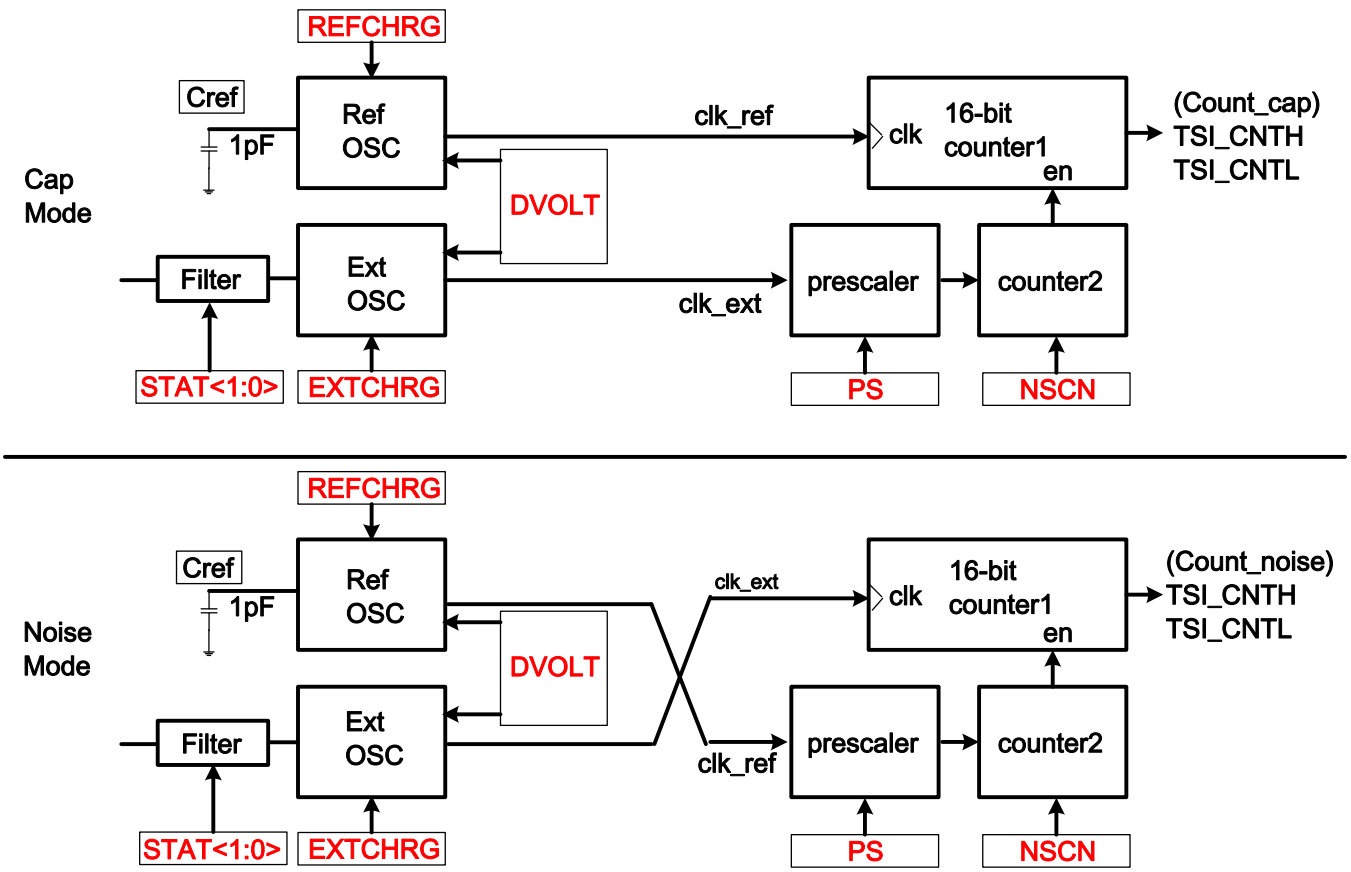


Figure 52-4. TSI noise detection mode block diagram

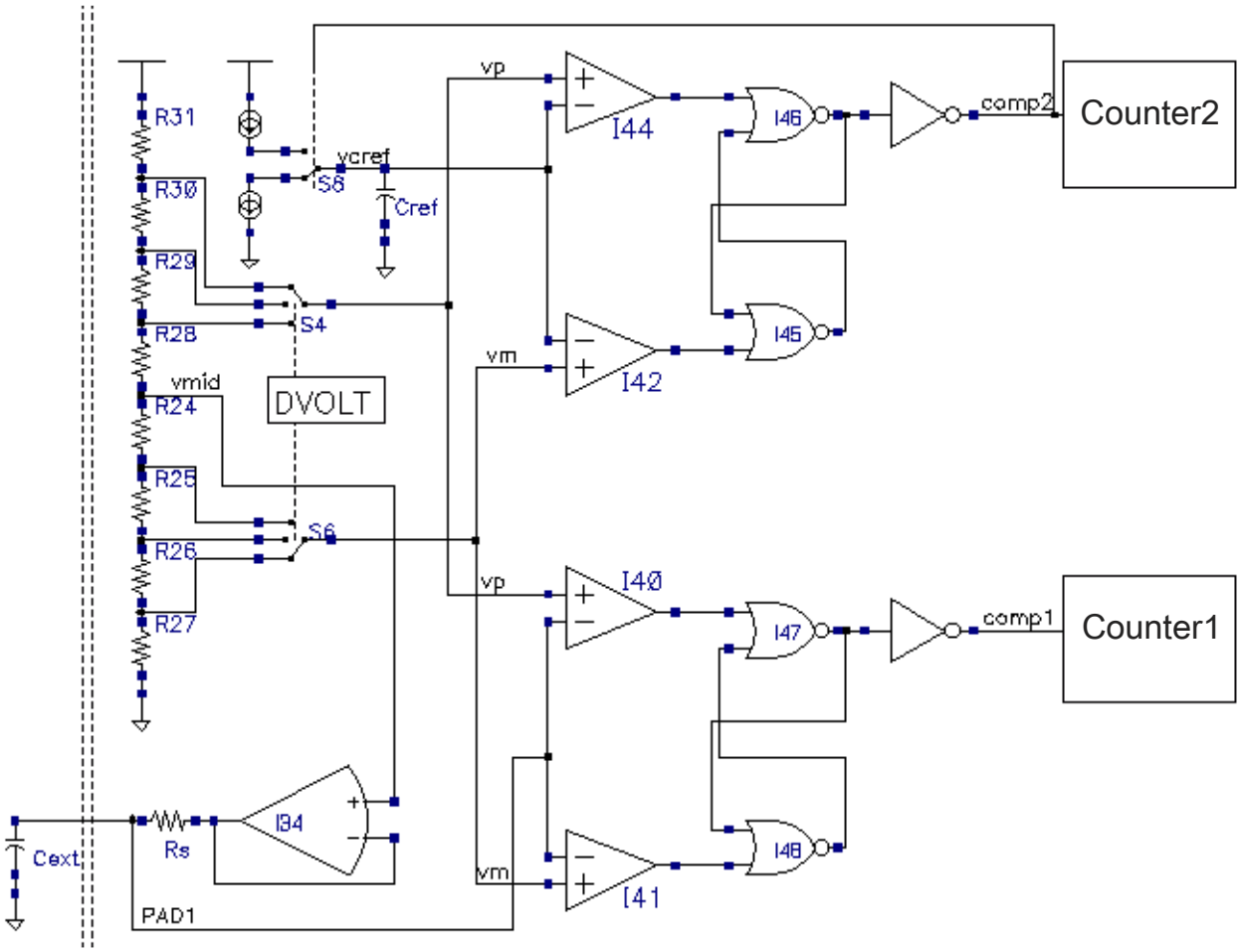


Figure 52-5. TSI circuit in noise detection mode

The table below shows all DVOLT values both in Bits and in V, all Rs values both in EXTCHRG Bits and in kΩ in order. The values indicated by valid black points can be used. The valid Rs/Dvolt values are: 184K/0.29V, 77K/0.29V, 77K/0.43V, 32K/0.43V, 32K/0.73V, 14K/0.73V, 14K/1.03V, 5K/1.03V

To determine the noise level, the TSI noise detection algorithm shall be performed by scanning this table following the arrow direction starting at maximum Rs and minimum DVOLT.

DVOLT (Bits)	Rs (Bits) (kΩ) / (V)	111	110	101	100	011
		5.5	14	32	77	184
11	0.29					
10	0.43					
01	0.73					
00	1.03					

The TSI noise detection algorithm shall be done by application software (assume software poll method is used to check TSI scan status) as described below, note the values in below steps are just used to illustrated the algorithm flow, the actual value should be consistent with the valid combinations as shown in the table above.

1. Enable TSI by setting GENCS[TSIEN] and select a channel by writing a channel number to GENCS[TSICH] just as does for normal function mode;
2. Enable noise detection mode by writing 11b to GENCS[MODE] (MODE bit 3 and 2 with 11);
3. Initialize the noise RF amplitude and noise detection mode sense duration (T) as below:
   a. Initialize Rs to the max value by writing 011b to GENCS[EXTCHRG] and GENCS[DVOLT] to the minimum value by writing 11b to GENCS[DVOLT];
   b. Set up GENCS[REFCHRG], GENCS[PS] and GENCS[NSCN] bits to set the noise detection mode sense duration (T). T= (2 x (2^{PS}) x NSCN x Cref x ΔV)/ Iref .

NOTE

NOTE: This time needs to be enough to detect the number of WINDOW bits for the minimum noise frequency. The minimum value of T (Tmin) is calculated as below: Tmin = (WINDOW+1)/F_{noise_min}; Where F_{noise_min} is the minimum noise frequency (0.15 MHz) and WINDOW is 2. This results in Tmin = 20 µs. Also this algorithm needs to be consistent with the valid Rs/Dvold combinations in above table.

4. Start TSI scan with software trigger or hardware trigger just as does for normal function mode
5. Wait until TSI scan is complete (GENCS[EOSF] = 1);
6. Read TSI counter value in TSICNT and then clear GENCS[EOSF] flag;

7. Check whether the TSI counter value is within the given counter window (WINDOW, can be 2 or 3 or 5): If the TSI counter value < WINDOW (i.e., the noise level is detected), go to step 12; Otherwise continue with the next step (meaning the noise level is too large);
8. If (Rs = minimum value) (i.e., GENCS[EXTCHRG] =000b, noise level is the largest at given DVOLT), go to step 10; otherwise continue with the next step;
9. Reduce Rs value by incrementing GENCS[EXTCHRG] by 1 and then go to step 4; (This action detects the next high level of noise)
10. If (DVOLT = maximum value) (i.e., GENCS[DVOLT] =00b), this means noise is too large to detect, go to END; otherwise continue with the next step;
11. Increase DVOLT to the next level by decrementing GENCS[DVOLT] by 1 and set Rs to the max value, then go to step 4; (It means noise level is higher, so need find high DVOLT)
12. Reduce Rs value by incrementing GENCS[EXTCHRG] by 1 if (Rs > minimum value) (i.e., GENCS[EXTCHRG] < 111b), and then go to END (Now a matching DVOLT corresponding to the noise level is found)
13. Reduce DVOLT by incrementing GENCS[DVOLT] by 1 if (Rs = maximum value) (i.e., GENCS[EXTCHRG] = 011b); (Now a matching DVOLT corresponding to the noise level is found)
14. END:

NOTE

The END condition of above algorithm can be one of

- TSI counter value within the WINDOW and Rs ≥ minimum value
- TSI counter value out of the WINDOW and Rs = minimum value and DVOLT = maximum value

At the end of the above steps, the correct matching DVOLT value and the electrode oscillator charge and discharge current value for the current noise level is found. That is, the correct GENCS[DVOLT] value and GENCS[EXTCHRG] value are found for the current noise level. And now users can proceed with normal capacitive sense procedure by keeping both GENCS[DVOLT] and GENCS[EXTCHRG] untouched, that is, users just need switch to normal capacitive sense mode by clearing GENCS[STAT_STUP[3:2]] bits and start TSI scan.

For typical applications, the noise detection/sense algorithm shall be performed first followed by normal capacitive sense for a given channel and then alternate between noise sense and capacitive sense as shown in Figure 52-6.

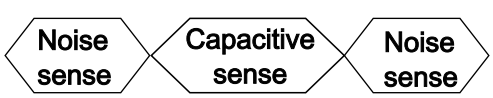


Figure 52-6. Noise detection/sense algorithm of typical application

The following flow chart shows how to detect touch with noise sense and normal capacitive sense.

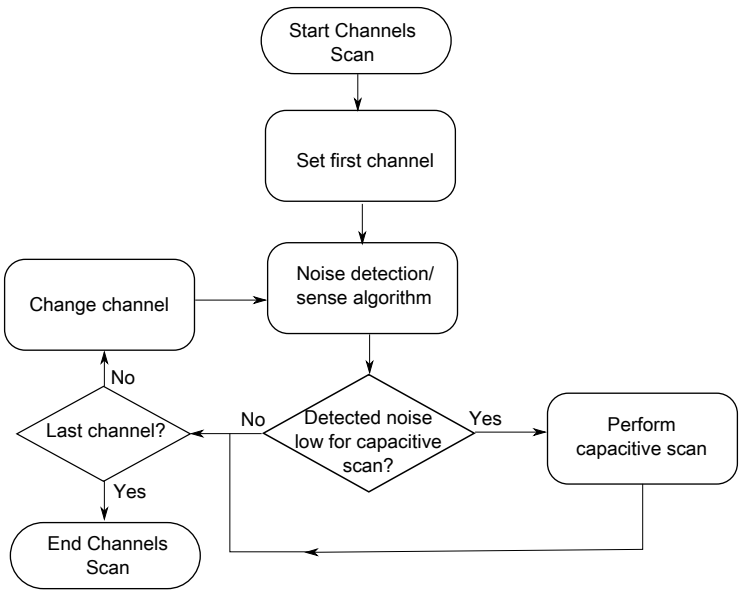


Figure 52-7. Detection of touch with noise sense and normal capacitive sense flow chart

One example of noise detection mode is shown in the following figure. in this figure the TSI is working in capacitive mode until 28 µs (T1) when it is changed to noise detection mode. In noise detection mode the selected pad is biased with 0.815V and all AC waveform in this pad is caused by a noise source external to the MCU.

It is possible to observe in the following figure that, in noise detection mode, the clkref output has the peak detection and the number of detected peaks can be counted or used by digital block. The clkext output has the internal oscillator output and can be used to set the maximum noise detection time window.

The waveform of the following figure shows two operations during noise detection mode. Again, this waveform is captured from NXP internal design simulation data, the actual useful points for noise detection should be consisted with the table provided above.

- The V(vp) and V(vm) thresholds are changed in 34.4 µs (T2).
- The Rs series resistance value is changed between 184 kΩ (GENCS[EXTCHRG]=011b), 77 kΩ (GENCS[EXTCHRG]=100) and 32 kΩ(GENCS[EXTCHRG] = 101). Because of this Rs change the amplitude of noise waveform change also.

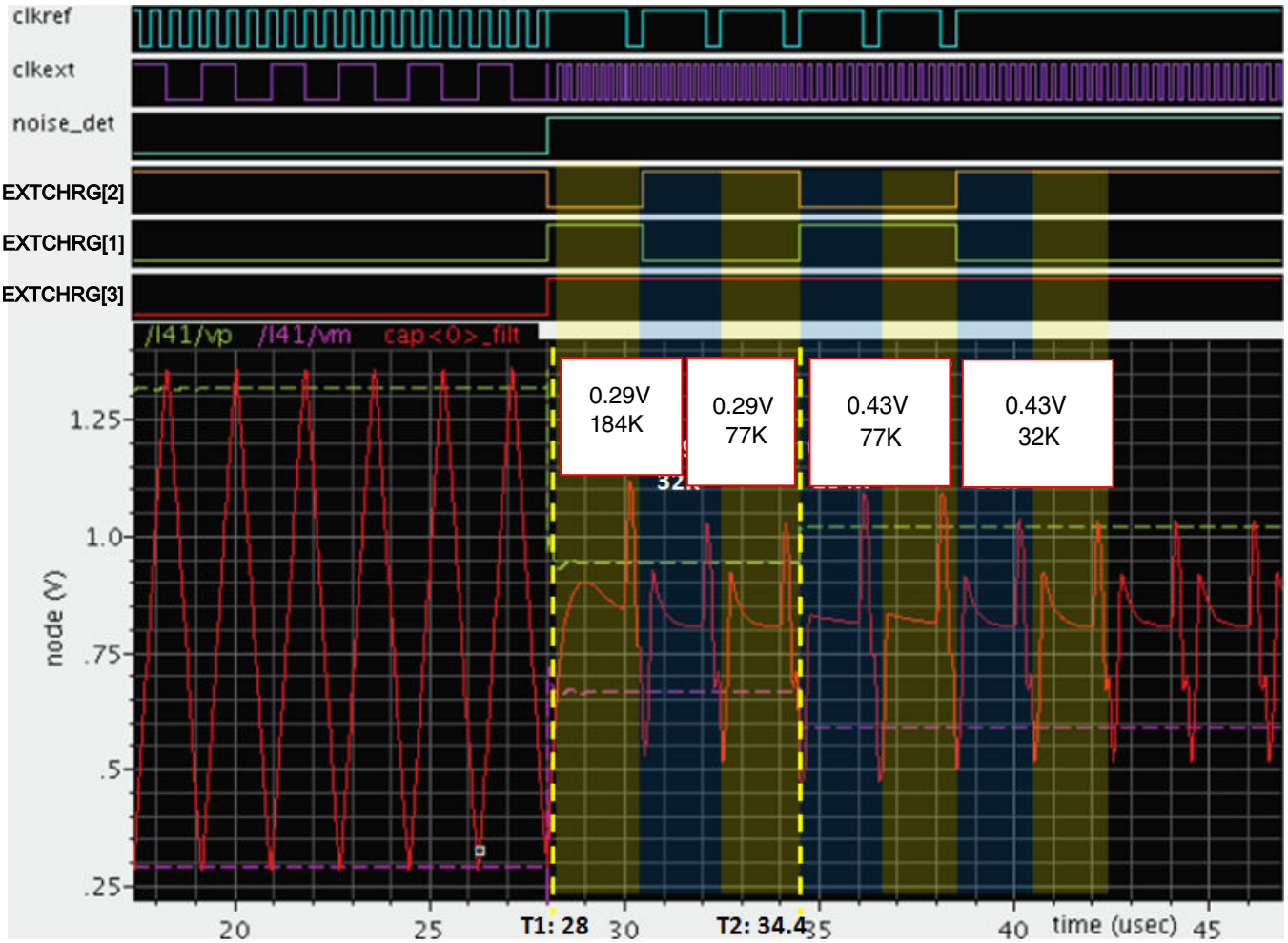


Figure 52-8. TSI noise detection mode waveform

52.5.13.1 Automatic noise mode

This mode is set by MODE[3:2] = 11 (noise mode 3). In this mode, the thresholds are incremented internally by the module until the point that there is no noise voltage trepassing the threshold.

The following diagram shows how it is done. The threshold comparator output goes to a counter and as the DVOLT control bits are increased the DVOLT thresholds are increased as well. The four bits are counted until 1111 (=15) and the counter is stop with this maximum value.

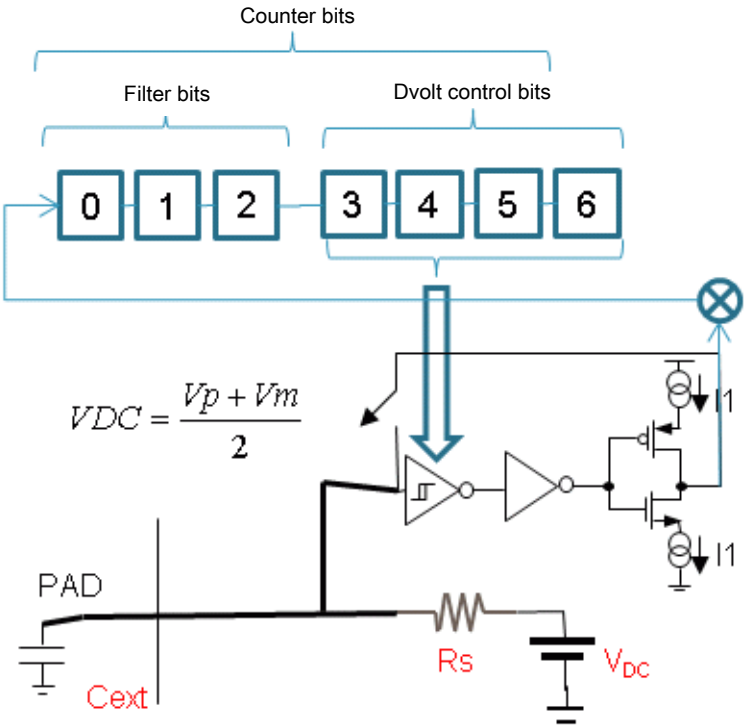


Figure 52-9. Block diagram automatic noise threshold operation

The signals that have different behavior in this noise mode (wrt capacitive mode) are shown in the following table.

Table 52-6. Signal properties in automatic noise operation mode

Name	Function	I/O type	Power Up/Reset state
MODE[3:2]	11—Noise mode operation with frequency limitation and automatic threshold counter.	I	00
EXTCHRG[2:1]	In this operation mode, these bits select the number of filter bits. 00—3 filter bits 01—2 filter bits 10—1 filter bit 11—no filter bit	I	00
EXTCHRG[0]	In this operation mode, this bit selects the series resistance. 0—uses Rs=32 kΩ 1—uses Rs=187 kΩ Independent of this bit selection, the threshold 15 is done with Rs = 5.5 kΩ	I	0

Table continues on the next page...

Table 52-6. Signal properties in automatic noise operation mode (continued)

Name	Function	I/O type	Power Up/Reset state
DVOLT[1:0]	Selects voltage rails of the internal oscillator	I	00
MODE[3:0]	DVOLT counter bits output. This field keeps 0000b if MODE[3:2] is not 11 after entering automatic noise mode.	O	0000

52.5.13.2 Single threshold noise modes

These modes are reset by MODE[3:2]=01 and 10.

In this mode, the thresholds are set by user via register bits as described in the following table.

During this mode the internal oscillator rails are set to the maximum (equivalent to DVOLT[1:0]=00).

Table 52-7. Signal properties in single noise modes (1,2)

Name	Function	I/O type	Power up / reset
MODE[3:2]	01 or 10- Single thrshold noise mode operation.	I	00
DVOLT[1:0], EXTCHRG[2:1]	In this operation mode these 4 bits are used select the noise threshold. These combinations are the maximum possible combinations, however, in real application, only the valid combinations in the above table should be used. 0000 - DVpm = 0.038 V, Vp = 0.834 V, Vm = 0.796 V 0001 - DVpm = 0.050 V, Vp = 0.830 V, Vm = 0.790 V 0010 - DVpm = 0.066 V, Vp = 0.848 V, Vm = 0.782 V 0011 - DVpm = 0.087 V, Vp = 0.858 V, Vm = 0.772 V 0100 - DVpm = 0.114 V, Vp = 0.872 V, Vm = 0.758 V 0101 - DVpm = 0.150 V, Vp = 0.890 V, Vm = 0.740 V 0110 - DVpm = 0.197 V, Vp = 0.914 V, Vm = 0.716 V 0111 - DVpm = 0.260 V, Vp = 0.945 V, Vm = 0.685 V 1000 - DVpm = 0.342 V, Vp = 0.986 V, Vm = 0.644 V 1001 - DVpm = 0.450 V, Vp = 1.040 V, Vm = 0.590 V 1010 - DVpm = 0.592 V, Vp = 1.111 V, Vm = 0.519 V 1011 - DVpm = 0.780 V, Vp = 1.205 V, Vm = 0.425 V 1100 - DVpm = 1.026 V, Vp = 1.328 V, Vm = 0.302 V 1101 - DVpm = 1.350 V, Vp = 1.490 V, Vm = 0.140 V	I	XXXX

Table continues on the next page...

Table 52-7. Signal properties in single noise modes (1,2) (continued)

Name	Function	I/O type	Power up / reset
	1110 - DVpm = 1.630 V, Vp = 1.630 V, Vm = 0 V 1111 - DVpm = 1.630 V, Vp = 1.630 V, Vm = 0 V		
EXTCHRG[0]	In this operation mode this bits selects the series resistance. 0 - uses Rs = 32 kΩ. 1- uses Rs = 187 kΩ. Independent of this bit selection the threshold 15 is done with Rs = 5.5 kΩ.	I	XX

Chapter 53
General-Purpose Input/Output (GPIO)

53.1 Chip-specific GPIO information

53.1.1 Number of GPIO signals

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

PTA4 includes a passive input filter that is enabled or disabled by PORTA_PCR4[PFE] control. This reset default is to have this function disabled.

53.1.2 Port control and interrupt module features

- 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.

Table 53-1. Ports summary

Feature	Port A	Port B	Port C	Port D	Port E
Pull select control	Yes	Yes	Yes	Yes	Yes
Pull select at reset	PTA1-PTA5=Pull up, Others=Pull down	Pull down	Pull down	Pull down	Pull down
Pull enable control	Yes	Yes	Yes	Yes	Yes

Table continues on the next page...

Table 53-1. Ports summary (continued)

Feature	Port A	Port B	Port C	Port D	Port E
Pull enable at reset	PTA0-PTA5=Enabled; Others=Disabled	Disabled	Disabled	Disabled	Disabled
Slew rate enable control	Yes	Yes	Yes	Yes	Yes
Slew rate enable at reset	Disabled	Disabled	Disabled	Disabled	Disabled
Passive filter enable control	PTA4=Yes; Others=No	No	No	No	No
Passive filter enable at reset	Disabled	Disabled	Disabled	Disabled	Disabled
Open drain enable control	Yes	Yes	Yes	Yes	Yes
Open drain enable at reset	Disabled	Disabled	Disabled	Disabled	Disabled
Pin mux control	Yes	Yes	Yes	Yes	Yes
Pin mux at reset	PTA0-PTA4=ALT7; Others=ALT0	ALT0	ALT0	ALT0	ALT0
Lock bit	Yes	Yes	Yes	Yes	Yes
Interrupt and DMA request	Yes	Yes	Yes	Yes	Yes
Digital glitch filter	No	No	No	No	No

NOTE

Port E has separate power domain to support QSPI flash

53.2 Introduction

The general-purpose input and output (GPIO) module is accessible via the peripheral bus and also 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.

53.2.1 Features

Features of the GPIO module include:

- Port Data Input register visible in all digital pin-multiplexing modes
- Port Data Output register with corresponding set/clear/toggle registers
- Port Data Direction register
- Zero wait state access to GPIO registers through IOPORT

NOTE

The GPIO module is clocked by system clock.

53.2.2 Modes of operation

The following table depicts different modes of operation and the behavior of the GPIO module in these modes.

Table 53-2. 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.

53.2.3 GPIO signal descriptions

Table 53-3. 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.

53.2.3.1 Detailed signal description

Table 53-4. GPIO interface-detailed signal descriptions

Signal	I/O	Description	
PORTA31–PORTA0 PORTB31–PORTB0 PORTC31–PORTC0 PORTD31–PORTD0 PORTE31–PORTE0	I/O	General-purpose input/output	
		State meaning	Asserted: The pin is logic 1. 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. 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.

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.

53.3 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	53.3.1/1602
400F_F004	Port Set Output Register (GPIOA_PSOR)	32	W (always reads 0)	0000_0000h	53.3.2/1603
400F_F008	Port Clear Output Register (GPIOA_PCOR)	32	W (always reads 0)	0000_0000h	53.3.3/1603
400F_F00C	Port Toggle Output Register (GPIOA_PTOR)	32	W (always reads 0)	0000_0000h	53.3.4/1604

Table continues on the next page...

GPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400F_F010	Port Data Input Register (GPIOA_PDIR)	32	R	0000_0000h	53.3.5/1604
400F_F014	Port Data Direction Register (GPIOA_PDDR)	32	R/W	0000_0000h	53.3.6/1605
400F_F040	Port Data Output Register (GPIOB_PDOR)	32	R/W	0000_0000h	53.3.1/1602
400F_F044	Port Set Output Register (GPIOB_PSOR)	32	W (always reads 0)	0000_0000h	53.3.2/1603
400F_F048	Port Clear Output Register (GPIOB_PCOR)	32	W (always reads 0)	0000_0000h	53.3.3/1603
400F_F04C	Port Toggle Output Register (GPIOB_PTOR)	32	W (always reads 0)	0000_0000h	53.3.4/1604
400F_F050	Port Data Input Register (GPIOB_PDIR)	32	R	0000_0000h	53.3.5/1604
400F_F054	Port Data Direction Register (GPIOB_PDDR)	32	R/W	0000_0000h	53.3.6/1605
400F_F080	Port Data Output Register (GPIOC_PDOR)	32	R/W	0000_0000h	53.3.1/1602
400F_F084	Port Set Output Register (GPIOC_PSOR)	32	W (always reads 0)	0000_0000h	53.3.2/1603
400F_F088	Port Clear Output Register (GPIOC_PCOR)	32	W (always reads 0)	0000_0000h	53.3.3/1603
400F_F08C	Port Toggle Output Register (GPIOC_PTOR)	32	W (always reads 0)	0000_0000h	53.3.4/1604
400F_F090	Port Data Input Register (GPIOC_PDIR)	32	R	0000_0000h	53.3.5/1604
400F_F094	Port Data Direction Register (GPIOC_PDDR)	32	R/W	0000_0000h	53.3.6/1605
400F_F0C0	Port Data Output Register (GPIOD_PDOR)	32	R/W	0000_0000h	53.3.1/1602
400F_F0C4	Port Set Output Register (GPIOD_PSOR)	32	W (always reads 0)	0000_0000h	53.3.2/1603
400F_F0C8	Port Clear Output Register (GPIOD_PCOR)	32	W (always reads 0)	0000_0000h	53.3.3/1603
400F_F0CC	Port Toggle Output Register (GPIOD_PTOR)	32	W (always reads 0)	0000_0000h	53.3.4/1604
400F_F0D0	Port Data Input Register (GPIOD_PDIR)	32	R	0000_0000h	53.3.5/1604
400F_F0D4	Port Data Direction Register (GPIOD_PDDR)	32	R/W	0000_0000h	53.3.6/1605
400F_F100	Port Data Output Register (GPIOE_PDOR)	32	R/W	0000_0000h	53.3.1/1602
400F_F104	Port Set Output Register (GPIOE_PSOR)	32	W (always reads 0)	0000_0000h	53.3.2/1603

Table continues on the next page...

GPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400F_F108	Port Clear Output Register (GPIOE_PCOR)	32	W (always reads 0)	0000_0000h	53.3.3/1603
400F_F10C	Port Toggle Output Register (GPIOE_PTOR)	32	W (always reads 0)	0000_0000h	53.3.4/1604
400F_F110	Port Data Input Register (GPIOE_PDIR)	32	R	0000_0000h	53.3.5/1604
400F_F114	Port Data Direction Register (GPIOE_PDDR)	32	R/W	0000_0000h	53.3.6/1605

53.3.1 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 unbonded 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 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

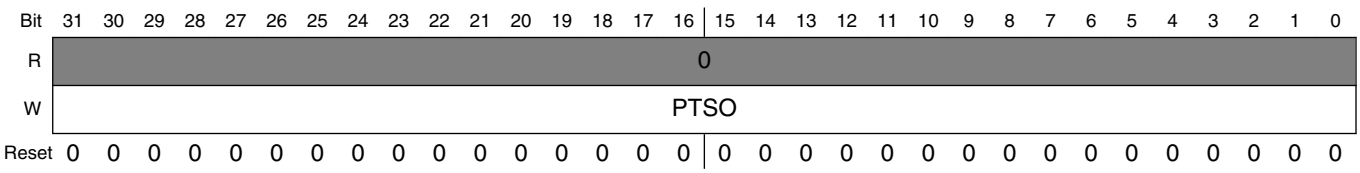
GPIOx_PDOR field descriptions

Field	Description
PDO	Port Data Output Register bits for unbonded pins return a undefined value when read. 0 Logic level 0 is driven on pin, provided pin is configured for general-purpose output. 1 Logic level 1 is driven on pin, provided pin is configured for general-purpose output.

53.3.2 Port Set Output Register (GPIOx_PSOR)

This register configures whether to set the fields of the PDOR.

Address: Base address + 4h offset



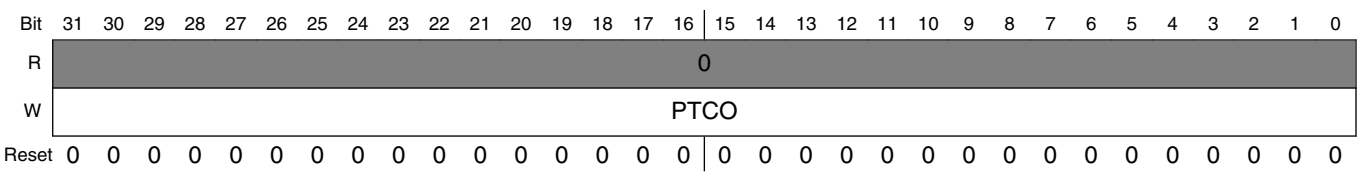
GPIOx_PSOR field descriptions

Field	Description
PTSO	Port Set Output Writing to this register will update the contents of the corresponding bit in the PDOR as follows: 0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is set to logic 1.

53.3.3 Port Clear Output Register (GPIOx_PCOR)

This register configures whether to clear the fields of PDOR.

Address: Base address + 8h offset



GPIOx_PCOR field descriptions

Field	Description
PTCO	Port Clear Output Writing to this register will update the contents of the corresponding bit in the Port Data Output Register (PDOR) as follows: 0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is cleared to logic 0.

53.3.4 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
PTTO	Port Toggle Output Writing to this register will update the contents of the corresponding bit in the PDOR as follows: 0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is set to the inverse of its existing logic state.

53.3.5 Port Data Input Register (GPIOx_PDIR)

NOTE

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded 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
PDI	Port Data Input 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. 0 Pin logic level is logic 0, or is not configured for use by digital function. 1 Pin logic level is logic 1.

53.3.6 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 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
PDD	Port Data Direction Configures individual port pins for input or output. 0 Pin is configured as general-purpose input, for the GPIO function. 1 Pin is configured as general-purpose output, for the GPIO function.

53.4 FGPIO memory map and register definition

The GPIO registers are also aliased to the IOPORT interface on the Cortex-M0+ from address 0xF800_0000.

Accesses via the IOPORT interface occur in parallel with any instruction fetches and will therefore complete in a single cycle. This aliased Fast GPIO memory map is called FGPIO.

Any read or write access to the FGPIO memory space that is outside the valid memory map results in a bus error. All register accesses complete with zero wait states, except error accesses which complete with one wait state.

FGPIO memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F800_0000	Port Data Output Register (FGPIOA_PDOR)	32	R/W	0000_0000h	53.4.1/1607
F800_0004	Port Set Output Register (FGPIOA_PSOR)	32	W (always reads 0)	0000_0000h	53.4.2/1607
F800_0008	Port Clear Output Register (FGPIOA_PCOR)	32	W (always reads 0)	0000_0000h	53.4.3/1608

Table continues on the next page...

FGPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F800_000C	Port Toggle Output Register (FGPIOA_PTOR)	32	W (always reads 0)	0000_0000h	53.4.4/1608
F800_0010	Port Data Input Register (FGPIOA_PDIR)	32	R	0000_0000h	53.4.5/1609
F800_0014	Port Data Direction Register (FGPIOA_PDDR)	32	R/W	0000_0000h	53.4.6/1609
F800_0040	Port Data Output Register (FGPIOB_PDOR)	32	R/W	0000_0000h	53.4.1/1607
F800_0044	Port Set Output Register (FGPIOB_PSOR)	32	W (always reads 0)	0000_0000h	53.4.2/1607
F800_0048	Port Clear Output Register (FGPIOB_PCOR)	32	W (always reads 0)	0000_0000h	53.4.3/1608
F800_004C	Port Toggle Output Register (FGPIOB_PTOR)	32	W (always reads 0)	0000_0000h	53.4.4/1608
F800_0050	Port Data Input Register (FGPIOB_PDIR)	32	R	0000_0000h	53.4.5/1609
F800_0054	Port Data Direction Register (FGPIOB_PDDR)	32	R/W	0000_0000h	53.4.6/1609
F800_0080	Port Data Output Register (FGPIOC_PDOR)	32	R/W	0000_0000h	53.4.1/1607
F800_0084	Port Set Output Register (FGPIOC_PSOR)	32	W (always reads 0)	0000_0000h	53.4.2/1607
F800_0088	Port Clear Output Register (FGPIOC_PCOR)	32	W (always reads 0)	0000_0000h	53.4.3/1608
F800_008C	Port Toggle Output Register (FGPIOC_PTOR)	32	W (always reads 0)	0000_0000h	53.4.4/1608
F800_0090	Port Data Input Register (FGPIOC_PDIR)	32	R	0000_0000h	53.4.5/1609
F800_0094	Port Data Direction Register (FGPIOC_PDDR)	32	R/W	0000_0000h	53.4.6/1609
F800_00C0	Port Data Output Register (FGPIOD_PDOR)	32	R/W	0000_0000h	53.4.1/1607
F800_00C4	Port Set Output Register (FGPIOD_PSOR)	32	W (always reads 0)	0000_0000h	53.4.2/1607
F800_00C8	Port Clear Output Register (FGPIOD_PCOR)	32	W (always reads 0)	0000_0000h	53.4.3/1608
F800_00CC	Port Toggle Output Register (FGPIOD_PTOR)	32	W (always reads 0)	0000_0000h	53.4.4/1608
F800_00D0	Port Data Input Register (FGPIOD_PDIR)	32	R	0000_0000h	53.4.5/1609
F800_00D4	Port Data Direction Register (FGPIOD_PDDR)	32	R/W	0000_0000h	53.4.6/1609
F800_0100	Port Data Output Register (FGPIOE_PDOR)	32	R/W	0000_0000h	53.4.1/1607

Table continues on the next page...

FGPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F800_0104	Port Set Output Register (FGPIOE_PSOR)	32	W (always reads 0)	0000_0000h	53.4.2/1607
F800_0108	Port Clear Output Register (FGPIOE_PCOR)	32	W (always reads 0)	0000_0000h	53.4.3/1608
F800_010C	Port Toggle Output Register (FGPIOE_PTOR)	32	W (always reads 0)	0000_0000h	53.4.4/1608
F800_0110	Port Data Input Register (FGPIOE_PDIR)	32	R	0000_0000h	53.4.5/1609
F800_0114	Port Data Direction Register (FGPIOE_PDDR)	32	R/W	0000_0000h	53.4.6/1609

53.4.1 Port Data Output Register (FGPIOx_PDOR)

This register configures the logic levels that are driven on each general-purpose output pins.

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 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

FGPIOx_PDOR field descriptions

Field	Description
PDO	Port Data Output Unimplemented pins for a particular device read as zero. 0 Logic level 0 is driven on pin, provided pin is configured for general-purpose output. 1 Logic level 1 is driven on pin, provided pin is configured for general-purpose output.

53.4.2 Port Set Output Register (FGPIOx_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

FGPIOx_PSOR field descriptions

Field	Description
PTSO	Port Set Output Writing to this register will update the contents of the corresponding bit in the PDOR as follows: 0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is set to logic 1.

53.4.3 Port Clear Output Register (FGPIOx_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

FGPIOx_PCOR field descriptions

Field	Description
PTCO	Port Clear Output Writing to this register will update the contents of the corresponding bit in the Port Data Output Register (PDOR) as follows: 0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is cleared to logic 0.

53.4.4 Port Toggle Output Register (FGPIOx_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

FGPIOx_PTOR field descriptions

Field	Description
PTTO	Port Toggle Output Writing to this register will update the contents of the corresponding bit in the PDOR as follows:

FGPIOx_PTOR field descriptions (continued)

Field	Description
	0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is set to the inverse of its existing logic state.

53.4.5 Port Data Input Register (FGPIOx_PDIR)

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

FGPIOx_PDIR field descriptions

Field	Description
PDI	Port Data Input 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. 0 Pin logic level is logic 0, or is not configured for use by digital function. 1 Pin logic level is logic 1.

53.4.6 Port Data Direction Register (FGPIOx_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 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

FGPIOx_PDDR field descriptions

Field	Description
PDD	Port Data Direction Configures individual port pins for input or output. 0 Pin is configured as general-purpose input, for the GPIO function. 1 Pin is configured as general-purpose output, for the GPIO function.

53.5 Functional description

53.5.1 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.

53.5.2 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.

53.5.3 IOPORT

The GPIO registers are also aliased to the IOPORT interface on the Cortex-M0+ from address 0xF800_0000. Accesses via the IOPORT interface occur in parallel with any instruction fetches and will therefore complete in a single cycle. If the DMA attempts to access the GPIO registers on the same cycle as an IOPORT access, then the DMA access will stall until any IOPORT accesses have completed.

Appendix A
Release Notes for Revision 3

A.1 General changes throughout document

- No substantial content changes

A.2 About This Document chapter changes

- No substantial content changes

A.3 Introduction chapter changes

- No substantial content changes

A.4 Core Overview chapter changes

- No substantial content changes

A.5 Memories and Memory Interfaces chapter changes

- Updated VBAT register file.

A.6 Clock Distribution using MCG chapter changes

- No substantial content changes

A.7 Reset and Boot chapter changes

- No substantial content changes

A.8 Kinetis ROM Bootloader changes

- No substantial content changes

A.9 Power Management chapter changes

- No substantial content changes

A.10 Security chapter changes

- No substantial content changes

A.11 Debug chapter changes

- No substantial content changes

A.12 Signal Multiplexing and Signal Descriptions chapter changes

- No substantial content changes

A.13 PORT changes

- No substantial content changes

A.14 System Integration Module chapter changes

- Updated SIM_SCGC6[RTC_RF] fields description.

A.15 EMVSIM changes

- No substantial content changes

A.16 RCM changes

- No substantial content changes

A.17 SMC changes

- No substantial content changes

A.18 PMC changes

- No substantial content changes

A.19 INTMUX changes

- No substantial content changes

A.20 LLWU changes

- No substantial content changes

A.21 MCM changes

- No substantial content changes

A.22 Crossbar switch module changes

- No substantial content changes

A.23 AIPS module changes

- No substantial content changes

A.24 MPU module changes

- No substantial content changes

A.25 BME configuration changes

- No substantial content changes

A.26 DMAMUX module changes

- No substantial content changes

A.27 eDMA module changes

- No substantial content changes

A.28 EWM changes

- No substantial content changes

A.29 WDOG changes

- No substantial content changes

A.30 MCG changes

- No substantial content changes

A.31 OSC changes

- No substantial content changes

A.32 RTC Oscillator changes

- No substantial content changes

A.33 MTB configuration changes

- No substantial content changes

A.34 FMC changes

- No substantial content changes

A.35 FTFA changes

- No substantial content changes

A.36 QuadSPI changes

- No substantial content changes

A.37 CRC changes

- No substantial content changes

A.38 SA-TRNG changes

- No substantial content changes

A.39 LP Trusted Cryptography chapter changes

- No substantial content changes

A.40 ADC changes

- No substantial content changes

A.41 CMP changes

- No substantial content changes

A.42 VREF changes

- No substantial content changes

A.43 DAC changes

- No substantial content changes

A.44 TPM changes

- No substantial content changes

A.45 LPTMR changes

- No substantial content changes

A.46 PIT module changes

- No substantial content changes

A.47 RTC changes

- No substantial content changes

A.48 USB full speed OTG controller changes

- No substantial content changes

A.49 SPI chapter changes

A.49.1 Chip-specific SPI information changes

- Updated SPI clocks

A.49.2 SPI module changes

- In PCS0/$\overline{SS}$—Peripheral Chip Select/Slave Select - Added note.
- In section, PCS0/$\overline{SS}$—Peripheral Chip Select/Slave Select : Updated note.

A.50 I2C changes

- No substantial content changes

A.51 LPUART changes

- No substantial content changes

A.52 FlexIO changes

- No substantial content changes

A.53 Touch sense input chapter changes

• Added new sections of Noise detection mode

A.54 GPIO changes

• No substantial content changes

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number KL82P121M72SF0RM
Revision 3, 08/2016