MCF51QW256RM
Rev.1.1, 07/2013

MCF51QW256 Reference Manual

This is the MCF51QW256 Reference Manual set consisting of the following files:

- MCF51QW256 Reference Manual Addendum, Rev 1
- MCF51QW256 Reference Manual, Rev 1

© Freescale Semiconductor, Inc., 2013. All rights reserved.

Freescale Semiconductor
Reference Manual Addendum

MCF51QW256RMAD
Rev.1, 07/2013

MCF51QW256 Reference Manual Addendum

This errata document describes corrections to the *MCF51QW256 Microcontroller Reference Manual*, order number MCF51QW256RM. For convenience, the addenda items are grouped by revision. Please check our website at http://www.freescale.com for the latest updates.

The current version available of the *MCF51QW256 Microcontroller Reference Manual* is Revision 1.0.

Table of Contents

© Freescale Semiconductor, Inc., 2013. All rights reserved.

freescale™ semiconductor

1 Addendum for Revision 1.0

Table 1. MCF51QW256RM Rev 1.0 addendum

<table>
<tr><th>Location</th><th>Description</th></tr>
<tr><td>Table 36-137, “Wait and guard time calculations”/Page 850</td><td>Updated C7816[TTYPE] = 0 value of Wait time parameter and C7816[TTYPE] = 1 value of Character wait time and Block wait time the following table.
<table>
<tr><th>Parameter</th><th>Reset value [ETU]</th><th>C7816[TTYPE] = 0 [ETU]</th><th>C7816[TTYPE] = 1 [ETU]</th></tr>
<tr><td>Wait Time</td><td>9600</td><td>((WI + 1) × 960 × (GTFD + 1)) - 1</td><td>Not used</td></tr>
<tr><td>Character wait time</td><td>Not used</td><td>Not used</td><td>$11 + 2^{(CWI - 1)}$</td></tr>
<tr><td>Block wait time</td><td>Not used</td><td>Not used</td><td>$10 + 2^{BWI} \times 960 \times (GTFD + 1)$</td></tr>
</table></td></tr>
<tr><td>Figure 21-1, “Multipurpose Clock Generator (MCG) block diagram”/ Page 366</td><td>Added the following Note below Multipurpose Clock Generator (MCG) block diagram:
Note: MCGPLLCLK is gated by LOCK signal</td></tr>
</table>

2 Revision history

Table 2 provides a revision history for this addendum.

Table 2. Revision history table

Revision	Substantive changes	Date of release
1.0	• Initial release. Updated the “Wait and guard time calculations” table in chapter 36, “Serial Communication Interface (SCI) / Universal Asynchronous Receiver/Transmitter (UART).” • Added Note below “Multipurpose Clock Generator (MCG) block diagram.”	07/2013

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/salestermsandconditions.

Freescale, the Freescale logo, AltiVec, C-5, CodeTest, CodeWarrior, ColdFire, C-Ware, Energy Efficient Solutions logo, Kinetis, mobileGT, PowerQUICC, Processor Expert, QorIQ, Qorivva, StarCore, Symphony, and VortiQa are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Airfast, BeeKit, BeeStack, ColdFire+, CoreNet, Flexis, MagniV, MXC, Platform in a Package, QorIQ Qonverge, QUICC Engine, Ready Play, SafeAssure, SMARTMOS, TurboLink, Vybrid, and Xtrinsic are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.
© 2013 Freescale Semiconductor, Inc.

Document Number: MCF51QW256RMAD
Rev.1
07/2013

freescale™
semiconductor

MCF51QW256 Reference Manual

Document Number: MCF51QW256RM
Rev. 1, 01/2013

Contents

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

Chapter 1
Preface

Chapter 2
Introduction

Chapter 3
Chip Configuration

Chapter 4
Memory Map

Chapter 5
Clock Distribution

Chapter 6
Reset and Boot

Chapter 7
Power Management

Chapter 8
Security

Chapter 9
Signal Multiplexing and Signal Descriptions

Chapter 10
Port Mux Control

Chapter 11
Core

Chapter 12
Enhanced Multiply-Accumulate Unit (EMAC)

Chapter 13
System Integration Module (SIM)

Chapter 14
Crossbar Switch

Chapter 15
Interrupt Controller (INTC)

Chapter 16
Low Leakage Wakeup Unit (LLWU)

Chapter 17
Reset Control Module (RCM)

Chapter 18
System Mode Controller (SMC)

Chapter 19
Power Management Controller (PMC)

Chapter 20
DMA Controller

Chapter 21
Multipurpose Clock Generator (MCG)

Chapter 22
Oscillator (OSC)

Chapter 23
Real Time Clock (RTC)

Chapter 24
Flash Memory Controller (FMC)

Chapter 25
Flash Memory Module (FTFL)

Chapter 26
EzPort

Chapter 27
Cryptographic Acceleration Unit (CAU)

Chapter 28
Random Number Generator Accelerator (RNGA)

Chapter 29
Cyclic Redundancy Check (CRC)

Chapter 30
Miscellaneous and Error Status Module (MESM)

Chapter 31
Modulo Timer (MTIM)

Chapter 32
Low Power Timer (LPTMR)

Chapter 33
FlexTimer (FTM)

Chapter 34
Serial Peripheral Interface (SPI)

Chapter 35
Inter-Integrated Circuit (I2C)

Chapter 36
Serial Communication Interface (SCI) / Universal Asynchronous Receiver/Transmitter (UART)

Chapter 37
Rapid GPIO (RGPIO)

Chapter 38
Enhanced GPIO (EGPIO)

Chapter 39
EGPIO Port Control

Chapter 40
External Interrupt (IRQ)

Chapter 41
Debug

Chapter 1
Preface

1.1 Overview

1.1.1 Purpose

This document describes the features, architecture, and programming model of the Freescale MCF51QW256 microcontroller.

1.1.2 Audience

This document is primarily for system architects and software application developers who are using or considering using the MCF51QW256 microcontroller in a system.

1.2 Conventions

1.2.1 Numbering systems

The following suffixes identify different numbering systems:

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

1.2.2 Typographic notation

The following typographic notation is used throughout this document:

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

1.2.3 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, or field that is either reserved for future use or for which, when written to, the module or chip behavior is unpredictable.
w1c	Write 1 to clear: Refers to a register bitfield that must be written as 1 to be "cleared."

1.2.4 Register reset

Information provided about every register includes each bit's value upon a reset event. The documented devices support multiple types of reset. The specific reset type that effects particular reset values can vary by module, by register within a module's programming model, and even by bit within a register.

For details about the reset type(s) affecting a module's registers, refer to the module's Chip Configuration information and register descriptions. When a register's details specify a reset type, in some cases other reset types do not affect the register.

For information about the various reset types, refer to the Reset details.

When a register's description does not specify a reset type, the reset type is Chip Reset (including Early Chip Reset).

Chapter 2
Introduction

2.1 MCF51QW256 Introduction

The MCF51QW256 is a member of the low-cost, low-power, high-performance Version 1 (V1) ColdFire family of 32-bit microcontroller units (MCUs). It uses the enhanced V1 Cold Fire core and is available with a variety of modules and single memory configuration. CPU clock rates on these devices can reach 50 MHz. MCF51QW256 includes 256 KB Flash, 64 KB RAM, hardware encryption support (Cryptographic Acceleration Unit and Random Number Generator).

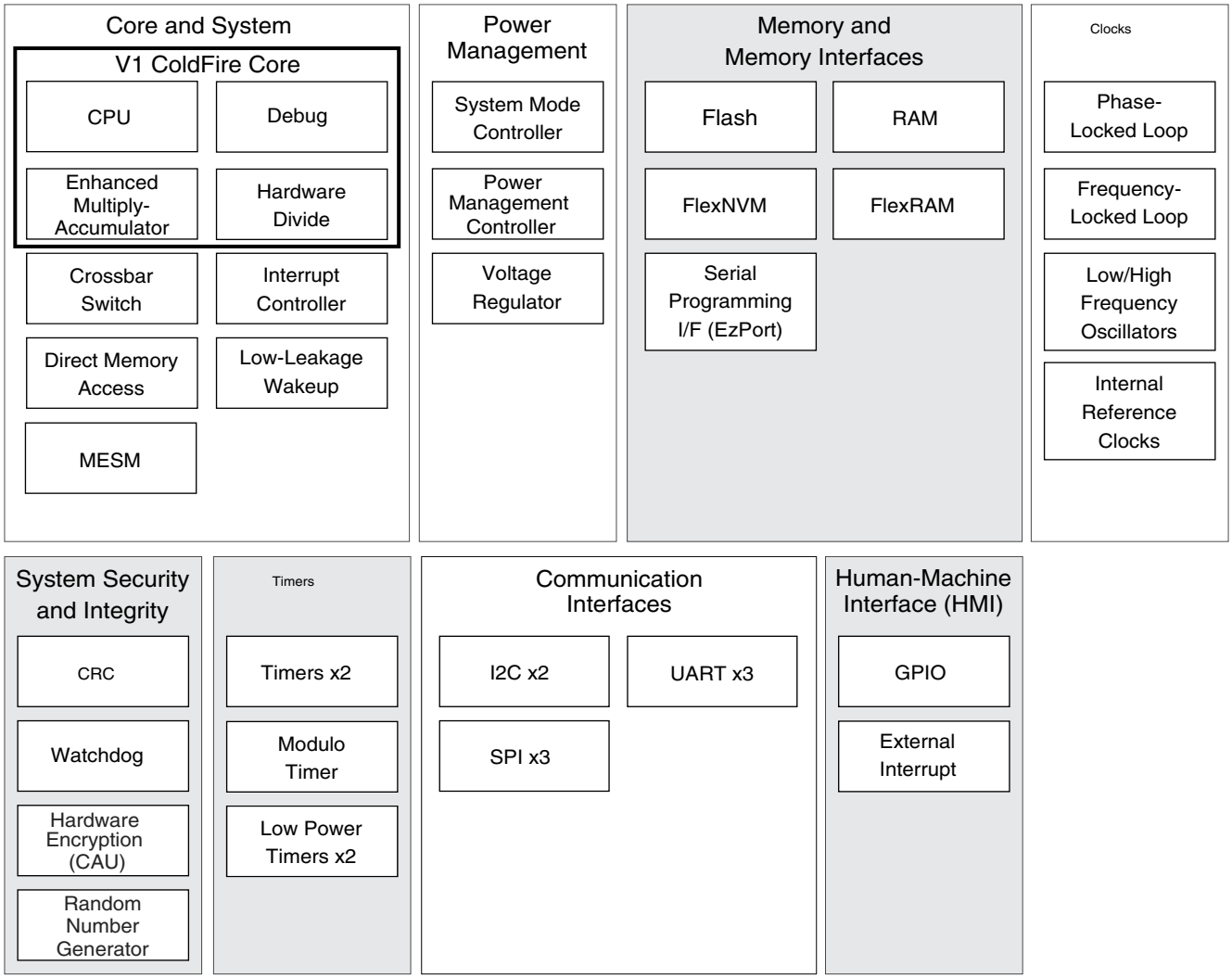


Figure 2-1. MCF51QW256 block diagram

2.2 MCF51QW256 Feature Summary

The following table lists the features integrated on MCF51QW256.

Table 2-1. MCF51QW256 Feature Summary

Feature	MCF51QW256
Hardware Characteristics	
Voltage range	1.85 V to 3.6 V
Package	44-pin MAPLGA (5 x 5 mm^2)
Temperature range, ambient (T_A)	-40 °C to 85 °C
Temperature range, junction (T_J)	-40 °C to 105 °C
Core and System	

Table continues on the next page...

Table 2-1. MCF51QW256 Feature Summary (continued)

Feature	MCF51QW256
Central processing unit (CPU)	High-performance Version 1 (V1) ColdFire core with EMAC and DIV hardware acceleration Implements instruction set revision C (ISA_C)
Maximum CPU frequency	50 MHz
Interrupt controller (INTC)	Supports 7 priority levels and software interrupt acknowledges
Direct memory access (DMA) controller	Four independently programmable channels provide the means to transfer data directly between system memory and I/O peripherals
Low-leakage wake-up unit (LLWU)	14 external wake-up pins with digital glitch filter 2 internal wake-up sources RESET pin always treated as reset wakeup in VLLS modes
Debug	Integrated ColdFire DEBUG_Rev_B+ interface with single wire BDM Real-time debug support, with six hardware breakpoints that can be configured to halt the processor or generate debug interrupt Capture of compressed processor status and debug data into trace buffer On-chip trace buffer provides programmable start/stop recording conditions
Memory and Memory Interfaces	
Total flash memory	Up to 288 KB (256 KB + 32 KB)
Flash memory	Up to 256K (Dual 128 KB Swappable Program Flash)
FlexNVM	Up to 32 KB
FlexRAM	Up to 2 KB
Total random-access memory (RAM)	Up to 66 KB (64 KB + 2 KB) Supports nibble-wide parity check code on 64 KB RAM
FlexMemory (FlexNVM plus FlexRAM) configuration examples[1]	Example 1: 32 KB data flash, 2 KB additional RAM, and no EEPROM Example 2: 2 KB EEPROM and no data Flash and no RAM Example 3: Partial data flash and EEPROM
Low-leakage standby memory	1 KB or 32 KB RAM in VLLS2 mode 32 bytes of register based storage
EzPort	Support Flash In-System Programming
Clocks	
External crystal oscillator or resonator	Low range, low power or full-swing: 32 - 40 kHz Medium range, low power or full-swing: 1 - 8 MHz High range, low power or full-swing: 8 - 32 MHz
External square wave input clock	DC to 50 MHz

Table continues on the next page...

Table 2-1. MCF51QW256 Feature Summary (continued)

Feature	MCF51QW256
Internal clock references	Two internal trimmable reference clocks: • 32 kHz • 2 MHz Internal 1 kHz low power oscillator
Phase-locked loop (PLL)	Up to 100 MHz VCO
System Security and Integrity	
Random number generator (RNGA)	Supports pseudo-random number (PRNG) generators
Cryptographic Acceleration Unit (CAU)	Providing Hardware Encryption for: • DES • AES-128, AES-192, AES-256 • SHA-1 and SHA-256 • MD5 Enables more complex algorithms such as 3DES with software encryption libraries that use the preceding basic security blocks.
CAU Performance	AES-128 Encryption or Decryption: 30.4 Mbit/s AES-128 Encryption and Decryption(text-in-ram): 15.2 Mbit/s (full duplex mode) AES-128 Encryption and Decryption(text-in-flash):14.4 Mbit/s (full duplex mode) AES-128 Key expansion + Encryption or Decryption(text-in-ram): 10.7 Mbit/s AES-128 Key expansion + Encryption or Decryption(text-in-flash):9.1 Mbit/s
Cyclic redundancy check (CRC)	User configurable 16/32-bit hardware CRC generator circuit with programmable generator polynomial Supports checksumming of any memory image
COP watchdog module	Yes
Unique Identification (ID) Number	128-bit wide
Analog	
Power management controller (PMC)	Low voltage warning and detect with selectable trip points Run and stop regulation modes to support low power MCU operation Several low power and low leakage stop modes
Timers	
16-bit Flexible timer 0 (FTM0)	2 channels
16-bit Flexible timer 1 (FTM1)	5 Channels
16-bit MTIM	16-bit modulo timer
Low-power timer (LPTMR0) [2]	1-channel, 16-bit pulse counter or periodic interrupt

Table continues on the next page...

Table 2-1. MCF51QW256 Feature Summary (continued)

Feature	MCF51QW256
Real Time Clock (RTC)	On-chip hardware calendaring and alarm functionality
Communication Interfaces	
16 bit Serial peripheral interface (SPI0)	1 with independent 8-byte transmit and receive FIFOs
16 bit Serial peripheral interface (SPI1, SPI2)	2 (without FIFO)
Inter-Integrated Circuit (I^2C)	2
Universal asynchronous receivers/transmitters (UART0)	Serial communication interface (SCI) Support for ISO 7816 protocol for interfacing with smart cards Hardware flow control Higher baud rates (CPU clock) Independent data FIFO for transmit and receive
Universal asynchronous receivers/transmitters (UART1 and UART2)	Serial communication interface (SCI) Hardware flow control(For UART1 only)
Human-Machine Interface (HMI)	
Enhanced General-purpose input/output (EGPIO)	Up to 26 Pin interrupt / DMA request capability Digital glitch filter Hysteresis and configurable pull-up/pull-down device on all input pins Configurable slew rate and drive strength on all output pins
Rapid General-purpose input/output (RGPIO)[3]	14 bits of high-speed GPIO functionality connected to the processor's local 32-bit bus
Interrupt Request Pin (IRQ)	Rising or falling edge selection Level sensitivity option Configurable internal pull-up

1. FlexNVM can be used as program flash, as data flash, or, in conjunction with FlexRAM, as high-endurance EEPROM or a combination of data flash and EEPROM.
2. Optimized for low power.
3. Shared with EGPIO pins

2.3 MCF51QW256 Feature Set

Devices within the entry level families have some or all the following features:

Table 2-2. MCF51QW256 Feature Set

Operating Characteristics	• Voltage range 1.85 V - 3.6 V • Flash write voltage 1.85 V - 3.6 V • Temperature range (T_A) -40 to 85 °C • Flexible modes of operation

Table continues on the next page...

Table 2-2. MCF51QW256 Feature Set (continued)

Core features	• Up to 50 MHz Version 1 (V1) ColdFire CPU
	• Provides Dhrystone 2.1 integer performance of:
	  • 1.10 DMIPS per MHz performance when running from internal RAM
	  • 0.99 DMIPS per MHz when running from flash
	• Implements instruction set revision C (ISA_C)
	• Enhanced Multiply Accumulate (EMAC) Unit and Hardware Divide Module

Clocks	• 1 MHz to 32 MHz crystal oscillator
	• 32 kHz crystal oscillator
	• Multi-purpose clock generator
	  • PLL and FLL operation
	  • Internal Reference (32 kHz or 2/4 MHz)
	  • 1 kHz internal LPO clock

System debug, protection and power management features	• Various stop, wait, and run modes to provide lower power based on application needs. • Peripheral clock enable register can disable clocks to unused modules, thereby reducing currents • Low voltage warning and detect with selectable trip points • Illegal opcode and illegal address detection with programmable reset or processor exception response • Hardware CRC module to support fast cyclic redundancy checks (CRC) • Cryptographic Acceleration Unit (CAU) • Random Number Generator Accelerator (RNGA) • 128-bit unique chip identifier • Hardware watchdog
Debug	• Integrated ColdFire DEBUG_Rev_B+ interface with single wire BDM connection • Real-time debug support, with six hardware breakpoints (4 PC, 1 address pair and 1 data) that can be configured into a 1- or 2-level trigger • Capture of compressed processor status and debug data into on-chip trace buffer provides program (and optional slave bus data) trace capabilities • On-chip trace buffer provides programmable start/stop recording conditions • Debug resources are accessible via single-pin BDM interface or the privileged WDEBUG • EzPort support flash In-System Programming.
DMA Controller	• Four independently programmable channels provide the capabilities to directly transfer data between system memory and I/O peripherals
Timers	• Motor control/general purpose timer (FTM) • 16-bit modulo timer (MTIM) • Low Power Timer for Pulse Counter or periodic interrupt (LPTMR0)
Real Time Clock	• Real Time Clock with Calendaring • 32 kHz operation from internal RC or external 32 kHz clock • Alarm functionality

Table continues on the next page...

Table 2-2. MCF51QW256 Feature Set (continued)

Communications	• Universal asynchronous receiver/transmitter (UART)/ Serial communications interface (SCI) • Serial peripheral interface (SPI) • Inter-Integrated Circuit (I2C)
Input/Output	• 26 GPIO pins with interrupt with selectable polarity • 14 pins with programmable glitch filter • Hysteresis and configurable pull up/down device on all input pins • Configurable slew rate and drive strength on all output pins • Up to 14 rapid general purpose I/O (RGPIO) pins connected to the processor's high speed 32-bit platform bus with faster set, clear, and toggle functionality
NVM Memory/On-Chip RAM	• Flash memory read and write down to 1.85 V • FlexMemory for additional program space or EEPROM • Flash security features and block protection • Nibble parity checking on 64 KB On-chip SRAM • Ability for software to insert single-bit errors to "check the checkers"

2.4 Part Numbers and Packaging

2.4.1 Format

Part numbers for this device have the following format:

Q CCCC DD MMM T PP

2.4.1.1 Fields

This table lists the possible values for each field in the part number (not all combinations are valid):

Field	Description	Values
Q	Qualification status	• P = Prequalification • M = Fully qualified, general market flow
CCCC	Core code	CF51 = ColdFire V1
DD	Device number	QW
MMM	Memory size	256 = 256K Flash
T	Temperature range (°C)	C = –40 to 85
PP	Package identifier	• HS = 44 pin Laminate QFN (lead-free)

2.4.2 Orderable Part Numbers

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

Table 2-3. Orderable part numbers summary

Freescale part number	Description	RAM	Flash	Package	Temperature
MCF51QW256CHS	MCF51QW256 ColdFire Microcontroller	64 KB	256 KB	44 MAPLGA	–40 to 85 °C

Chapter 3
Chip Configuration

3.1 Introduction

This chip configuration information consists of details about the individual modules that are specific to the chip. The information includes:

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

NOTE

For clock gating information that applies to modules generally, refer to Clock gating. Any additional clock gating info that is specific to a module appears in the module's dedicated chip configuration details.

3.2 Core Modules

3.2.1 Version 1 (V1) ColdFire Core Configuration

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

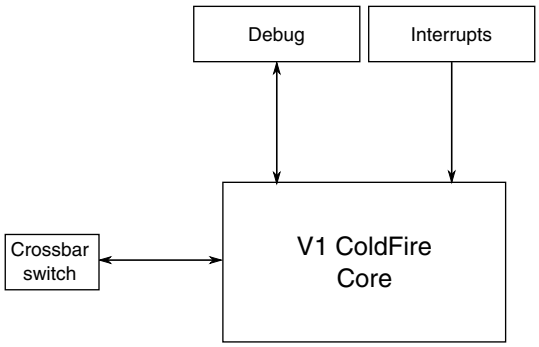


Figure 3-1. Core configuration

Table 3-1. Reference links to related information

Topic	Related module	Reference
Full description	V1 ColdFire core	Core
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Debug	Background debug mode (BDM) single-pin serial interface Real-time debug and trace	Debug
Interrupts	Interrupt controller (INTC)	INTC
System/instruction/data bus module	Crossbar switch	Crossbar switch
System/instruction module	Enhanced multiply-accumulate (EMAC) unit	EMAC

3.2.2 Debug Configuration

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

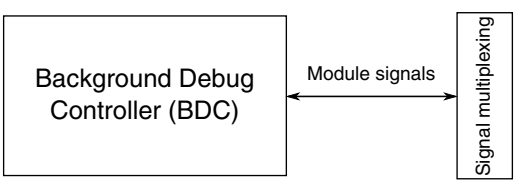


Figure 3-2. Debug configuration

Table 3-2. Reference links to related information

Topic	Related module	Reference
Full description	V1 ColdFire core debug • Background debug mode (BDM) • Background debug controller (BDC)	Debug
Signal multiplexing	Port mux control	Signal multiplexing

3.3 System Modules

3.3.1 Crossbar Switch Configuration

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

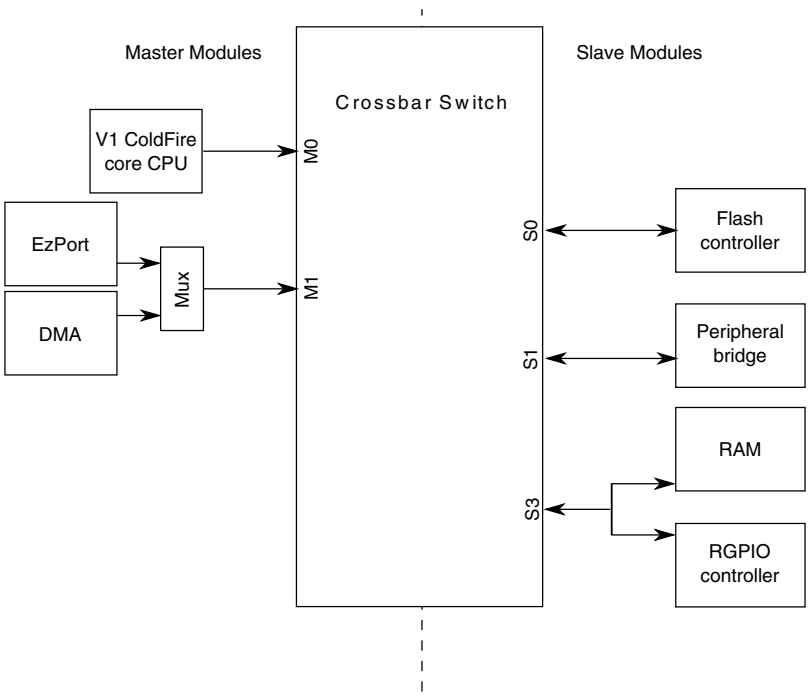


Figure 3-3. Crossbar switch integration

Table 3-3. Reference links to related information

Topic	Related module	Reference
Full description	Crossbar switch	Crossbar Switch
System memory map		System memory map
Clocking		Clock Distribution
Crossbar switch master	V1 ColdFire core CPU	Core
Crossbar switch master	DMA controller	DMA controller
Crossbar switch master	EzPort	EzPort
Crossbar switch slave	Flash memory controller	Flash memory controller
Crossbar switch slave	Peripheral bridge	Peripheral bridge
Crossbar switch slave	RAM	RAM
Crossbar switch slave	RGPIO	RGPIO

3.3.1.1 Crossbar Switch Master Assignments

This device contains the following master connections to the crossbar switch.

Master module	Master port number
CPU	0
DMA controller and EzPort (shared)	1

NOTE

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

3.3.1.2 Crossbar Switch Slave Assignments

This device contains three slave connections to the crossbar switch.

Slave module	Slave port number
Flash memory controller	0
Peripheral bridge	1
RAM and RGPIO (shared)	3

3.3.2 Peripheral Bridge Configuration

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

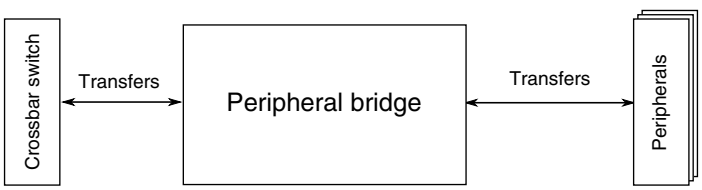


Figure 3-4. Peripheral bridge configuration

Table 3-4. Reference links to related information

Topic	Related module	Reference
System memory map		System memory map
Clocking		Clock distribution
Crossbar switch	Crossbar switch	Crossbar switch

3.3.2.1 Peripheral Bridge Interfaces

The peripheral bridge has two interfaces for transfers to and from modules.

Table 3-5. Peripheral bridge interfaces

Interface size	Connected modules
32-bit	DMA controller
8-bit	All other modules with an assigned slot in the peripheral bus memory map

3.3.2.2 Memory Map and Module Register Access

The peripheral bridge enables access to the registers of most of the modules on this device. See the memory map tables for the memory slot assignment for each module.

3.3.3 DMA Controller Configuration

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

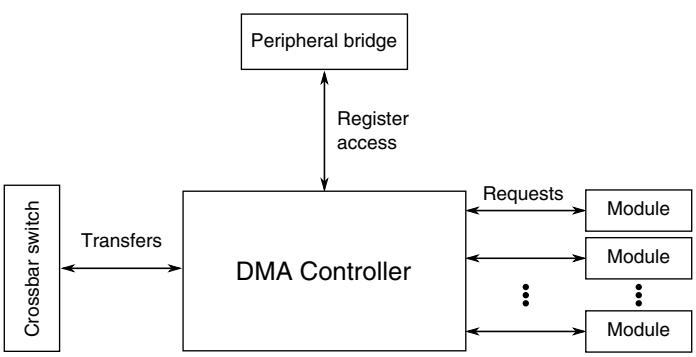


Figure 3-5. DMA Controller configuration

Table 3-6. Reference links to related information

Topic	Related module	Reference
Full description	DMA controller	DMA controller
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Requests		DMA request sources

3.3.3.1 DMA Request Sources

The DMA request sources for this device follow.

Table 3-7. DMA request sources

Request ID	Assignment to DMA Channel 0	Assignment to DMA Channel 1	Assignment to DMA Channel 2	Assignment to DMA Channel 3
0	SPI0_Tx	SPI0_Tx	SPI1_Tx	SPI1_Tx
1	SPI0_Rx	SPI0_Rx	SPI1_Rx	SPI1_Rx
2	SCI0_Tx	SCI1_Tx	SCI1_Tx	SCI0_Tx
3	SCI0_Rx	SCI1_Rx	SCI1_Rx	SCI0_Rx
4	IIC0	IIC0	SPI2_Tx	SPI2_Tx
5	IIC1	IIC1	SPI2_Rx	SPI2_Rx
6	Reserved	Reserved	SCI2_Tx	SCI2_Tx
7	Reserved	Reserved	SCI2_Rx	SCI2_Rx
8	—	—	Reserved	Reserved
9	FTM0_ch0	FTM0_ch0	FTM1_ch0	FTM1_ch0
10	FTM0_ch1	FTM0_ch1	FTM1_ch1	FTM1_ch1
11	FTM1_ch2	FTM1_ch3	FTM1_ch4	FTM1_ch5
12	FTM1_ch3	FTM1_ch4	FTM1_ch5	FTM1_ch2
13	—	—	Reserved	Reserved
14	—	—	—	—
15	PTA	PTB	PTC	PTD

3.3.4 Interrupt Controller (INTC) Configuration

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

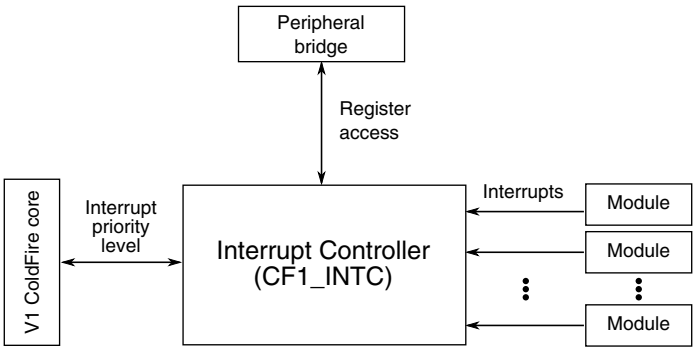


Figure 3-6. INTC configuration

Table 3-8. Reference links to related information

Topic	Related module	Reference
Full description	Interrupt Controller (INTC)	INTC
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management

3.3.4.1 Interrupt priority levels

The CF1_INTC module implements a sparsely populated 7 × 9 matrix of levels (7) and priorities within each level (9).

3.3.4.2 Interrupt Channel Assignments

The interrupt vector assignments are defined in the following table.

NOTE

Level 7 interrupts are non-maskable interrupts, while Level 6-1 are maskable interrupts. For more information, refer to the Interrupt Controller chapter.

Table 3-9. Interrupt vector assignments

Address offset	Vector Number	Level Priority[1]	Assignment
0x000	0		Initial supervisor Stack Pointer
0x004	1		Initial Program Counter
0x008 - 0x0FC	2-63		Reserved for internal CPU exceptions

Table continues on the next page...

Table 3-9. Interrupt vector assignments (continued)

Address offset	Vector Number	Level Priority[1]	Assignment
0x100	64	7(mid)	IRQ0
0x104	65	7(3)	LVD
0x108	66	7(2)	Low Leakage Wake Up[2]
0x10C	67	7(1)	MCG Loss of Lock
0x10E		6(7)	Reserved for remapped vector #1
0x10F		6(6)	Reserved for remapped vector #2
0x110	68	6(5)	Flash[3]
0x114	69	6(4)	DMA _CH0
0x118	70	6(3)	DMA _CH1
0x11C	71	6(2)	DMA _CH2
0x120	72	6(1)	DMA _CH3
0x124	73	5(7)	
0x128	74	5(6)	RNGA
0x12C	75	5(5)	
0x130	76	5(4)	FTM1 Fault+Overflow
0x134	77	5(3)	FTM1 Channel 0
0x138	78	5(2)	FTM1 Channel 1
0x13C	79	5(1)	FTM1 Channel 2
0x140	80	4(7)	FTM1 Channel 3
0x144	81	4(6)	FTM1 Channel 4
0x148	82	4(5)	FTM1 Channel 5
0x14C	83	4(4)	FTM0 Fault+Overflow
0x150	84	4(3)	FTM0 Channel 0
0x154	85	4(2)	FTM0 Channel 1
0x158	86	4(1)	SPI0
0x15C	87	3(7)	IRQ3
0x160	88	3(6)	SCI0 (Err, TX, RX) [4]
0x164	89	3(5)	RTC Interrupt
0x168	90	3(4)	IIC0
0x16C	91	3(3)	SPI2
0x170	92	3(2)	SPI1
0x174	93	3(1)	SCI1 (Err, TX, RX)[5]
0x178	94	2(7)	IRQ2
0x17C	95	2(6)	IIC1
0x180	96	2(5)	
0x184	97	2(4)	
0x188	98	2(3)	SCI2
0x18C	99	2(2)	LPT0

Table continues on the next page...

Table 3-9. Interrupt vector assignments (continued)

Address offset	Vector Number	Level Priority[1]	Assignment
0x190	100	2(1)	
0x194	101	1(7)	IRQ1
0x198	102	1(6)	MTIM
0x19C	103	7(0)	Level 7 Software Interrupt
0x1A0	104	6(0)	Level 6 Software Interrupt OR MESM Interrupt
0x1A4	105	5(0)	Level 5 Software Interrupt
0x1A8	106	4(0)	Level 4 Software Interrupt
0x1AC	107	3(0)	Level 3 Software Interrupt
0x1B0	108	2(0)	Level 2 Software Interrupt
0x1B4	109	1(0)	Level 1 Software Interrupt
0x1B8	110	1(5)	EGPIO Port A
0x1BC	111	1(4)	EGPIO Port B
0x1C0	112	1(3)	EGPIO Port D
0x1C4	113	1(2)	EGPIO Port C
0x1C8	114	1(1)	SPI_WAKEUP
0x1CC-0x3FC	115-255	-	Reserved - Unused for V1

1. x(y): were x is level priority and y is the priority within the level.
2. For wakeup from VLLSx modes:
   - One or two pending interrupts are serviced. One is serviced if the wakeup is from the LLWU wakeup pins, or two are serviced if the wakeup is from a CMP or LPTMR module.
   - Then instruction execution resumes via the reset vectors.
3. Flash interrupt is logical OR of FTFL CCIF and FTFL RDCOLERR interrupt
4. All of the SCI0 interrupt sources are joined in this single vector.
5. All of the SCI1 interrupt sources are joined in this single vector.

Table 3-10. ColdFire level, priority within level matrix interrupt assignments

Priority	Level	Priority within level								
		7	6	5	4	Midpoint	3	2	1	0
Highest	7					IRQ0[1]	LVD	Low Leakage Wake Up	MCG Loss of Clock	Level 7 SWI

Table continues on the next page...

Table 3-10. ColdFire level, priority within level matrix interrupt assignments (continued)

	6	INTC_PL 6P7	INTC_PL 6P6	Flash	DMA _CH0	x	DMA _CH1	DMA _CH2	DMA _CH3	Level 6 SWI OR MESM interrupt
	5		RNGA		FTM1 Fault +Overflow	x	FTM1 Channel 0	FTM1 Channel 1	FTM1 Channel 2	Level 5 SWI
	4	FTM1 Channel 3	FTM1 Channel 4	FTM1 Channel 5	FTM0 Fault +Overflow	x	FTM0 Channel 0	FTM0 Channel 1	SPI0	Level 4 SWI
	3	IRQ3	SCI0	RTC Interrupt	IIC0	x	SPI2	SPI1	SCI1	Level 3 SWI
	2	IRQ2	IIC1			x	SCI2	LPT0		Level 2 SWI
Lowest	1	IRQ1	MTIM	PORT A	PORT B	x	PORT D	PORT C	SPI_WAKEUP	Level 1 SWI

1. IRQ0 is non-maskable interrupt while IRQ[2,3,4] are maskable.

3.3.5 Low-Leakage Wakeup Unit (LLWU) Configuration

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

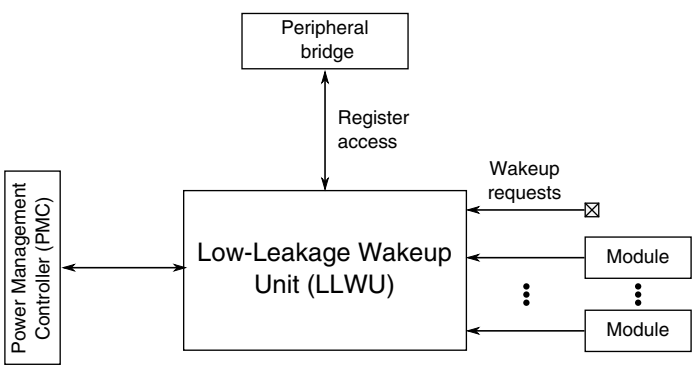


Figure 3-7. Low-Leakage Wakeup Unit configuration

Table 3-11. Reference links to related information

Topic	Related module	Reference
Full description	Low-Leakage Wakeup Unit (LLWU)	LLWU
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
	Power Management Controller (PMC)	PMC configuration
Wakeup requests		LLWU wakeup sources

3.3.5.1 LLWU Wakeup Sources

The LLWU module has the following internal and external inputs. WUP0- WUP15 are external pin inputs, and module interrupt flags (M0IF-M3IF) are internal peripheral connections.

NOTE

- The $\overline{\text{RESET}}$ pin is also a wakeup source when the LLWU's RST[LLRSTE] bit is 1 and the pin is enabled as $\overline{\text{RESET}}$ via port mux control.

Table 3-12. LLWU inputs

Wakeup Pin	Source		Wakeup Pin	Source
WUP0	Reserved		WUP10	PTC4
WUP1	PTA1		WUP11	PTC5
WUP2	PTA2		WUP12	Reserved
WUP3	PTB1		WUP13	PTC7
WUP4	PTB2		WUP14	PTD0
WUP5	PTB3		WUP15	PTD2
WUP6	PTB5		M0IF	LPTimer0 (see note)
WUP7	PTC0		M1IF	RTC Interrupt (see note)
WUP8	PTC1		M2IF	Reserved
WUP9	PTC2		M3IF	Reserved

NOTE

Requires the peripheral and the peripheral interrupt to be enabled. The internal module's WUPE bit is used to enable the internal module flag to be used as a wakeup input. After wakeup, the flags are cleared based on the peripheral clearing mechanism.

3.3.5.2 LLWU register reset

All LLWU registers are reset by Chip Reset not VLLS and by other reset types that trigger Chip Reset not VLLS. LLWU registers are unaffected by reset types that do not trigger Chip Reset not VLLS. For more information about the types of reset available on this chip, refer to the Reset details.

3.3.6 Computer Operating Properly (COP) Watchdog Configuration

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

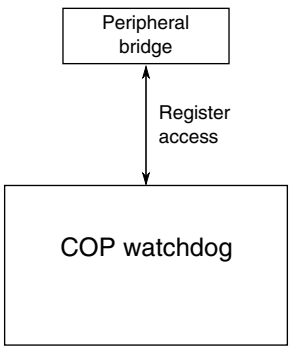


Figure 3-8. COP watchdog configuration

Table 3-13. Reference links to related information

Topic	Related module	Reference
Clocking		Clock distribution
Power management		Power management
Programming model	System Integration Module (SIM)	SIM

3.3.6.1 COP clocks

The two clock inputs for the COP are the 1 kHz clock and the bus clock.

3.3.6.2 COP Watchdog

The COP watchdog is intended to force a system reset when the application software fails to execute as expected. To prevent a system reset from the COP timer (when it is enabled), application software must reset the COP counter periodically. If the application program gets lost and fails to reset the COP counter before it times out, a system reset is generated to force the system back to a known starting point.

After any reset, the COP watchdog is enabled. If the COP watchdog is not used in an application, it can be disabled by clearing SOPT1[COPT].

The COP counter is reset by writing 0x55 and 0xAA (in that order) to the address of the SIM's Service COP (SRVCOP) register during the selected timeout period. Writes do not affect the data in the SRVCOP register. As soon as the write sequence is complete, the COP timeout period is restarted. If the program fails to perform this restart during the timeout period, the microcontroller resets. Also, if any value other than 0x55 or 0xAA is written to the SRVCOP register, the microcontroller immediately resets.

The SOPT1[COPCLKS] field selects the clock source used for the COP timer. The clock source options are either the bus clock or an internal 1 kHz clock source. With each clock source, there are three associated time-outs controlled by SOPT1[COPT]. The following table summarizes the control functions of the COPCLKS and COPT bits. The COP watchdog defaults to operation from the 1 kHz clock source and the longest time-out for that clock source (2^{10} cycles).

Control Bits		Clock Source	COP Window Opens (SOPT1[COPW]=1)	COP Overflow Count
SOPT1[COPCLKS]	SOPT1[COPT]			
N/A	00	N/A	N/A	COP is disabled
0	01	1 kHz	N/A	2^5 cycles (32 ms)
0	10	1 kHz	N/A	2^8 cycles (256 ms)
0	11	1 kHz	N/A	2^{10} cycles (1,024 ms)
1	01	Bus	6,144 cycles	2^{13} cycles
1	10	Bus	49,152 cycles	2^{16} cycles
1	11	Bus	196,608 cycles	2^{18} cycles

After the bus clock source is selected, windowed COP operation is available by setting SOPT1[COPW]. In this mode, writes to the SIM's SRVCOP register to clear the COP timer must occur in the last 25% of the selected timeout period. A premature write immediately resets the microcontroller. When the 1 kHz clock source is selected, windowed COP operation is not available.

The COP counter is initialized by the first writes to the SOPT1 register and after any system reset. Subsequent writes to SOPT1 have no effect on COP operation. Even if the application will use the reset default settings of the COPT, COPCLKS, and COPW bits, the user should write to the write-once SOPT1 register during reset initialization to lock in the settings. This will prevent accidental changes if the application program gets lost.

The write to the SIM's SRVCOP that services (clears) the COP counter should not be placed in an interrupt service routine (ISR) because the ISR could continue to be executed periodically even if the main application program fails.

If the bus clock source is selected, the COP counter does not increment while the microcontroller is in background debug mode or while the system is in stop (including VLPS) mode. The COP counter resumes when the microcontroller exits background debug mode or stop mode.

If the 1 kHz clock source is selected, the COP counter is re-initialized to zero upon entry to either background debug mode or stop (including VLPS) mode and begins from zero upon exit from background debug mode or stop mode.

Regardless of the bus selected, the COP is disabled when the chip enters a VLLSx mode. Upon a reset that wakes the chip from the VLLSx mode, the COP is re-initialized and enabled as for any reset.

3.3.7 PMC Configuration

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

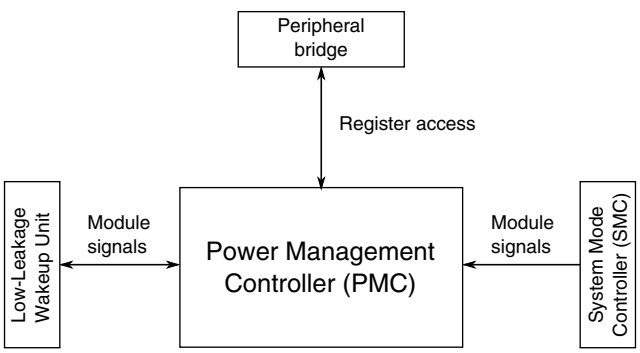


Figure 3-9. PMC configuration

Table 3-14. Reference links to related information

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

3.3.7.1 PMC register reset

Different portions of PMC registers reset on different MCU reset types. Refer to the detailed register descriptions. For information about the various reset types on this chip, refer to the Reset details.

3.3.8 System Mode Controller (SMC) Configuration

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

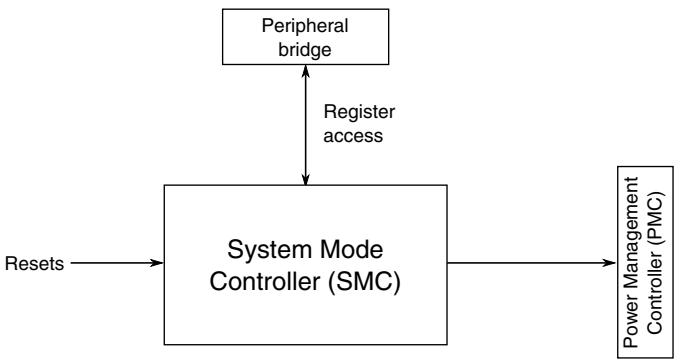


Figure 3-10. System Mode Controller configuration

Table 3-15. Reference links to related information

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

3.3.8.1 SMC register reset

Different SMC registers reset on different MCU reset types. Refer to the detailed register descriptions. For information about the various reset types on this chip, refer to the Reset details.

3.3.9 System Integration Module (SIM) Configuration

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

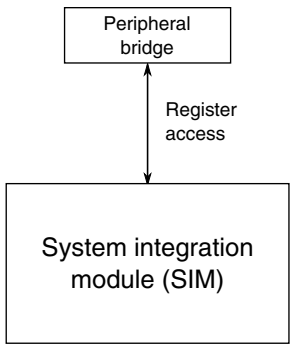


Figure 3-11. SIM configuration

Table 3-16. Reference links to related information

Topic	Related module	Reference
Full description	System Integration Module (SIM)	SIM
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management

3.3.9.1 SIM register reset

Different SIM registers reset on different MCU reset types. Refer to the detailed register descriptions. For information about the various reset types on this chip, refer to the Reset details.

3.4 Clock Modules

3.4.1 Multipurpose Clock Generator (MCG) Configuration

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

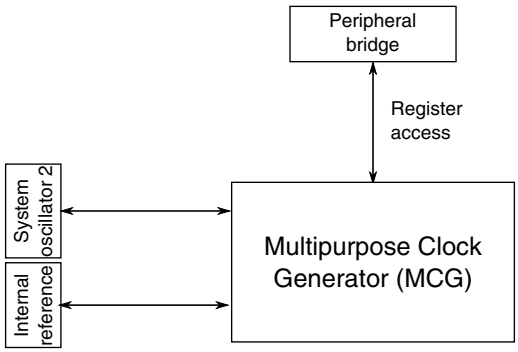


Figure 3-12. MCG configuration

Table 3-17. Reference links to related information

Topic	Related module	Reference
Full description	MCG	MCG
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management

NOTE

In MCG module, OSC2 can be used as a clock source for PLL.

3.4.1.1 MCG oscillator-frequency trim settings: factory and custom

Factory-programmed values for trimming oscillator frequency are stored in the nonvolatile information register (IFR) and are automatically loaded into the MCG's C3 and C4 registers after any reset.

A portion of the chip's program flash memory can be used to store other, custom settings for frequency trimming. These locations appear as FTRIM and TRIM in the following table.

Table 3-18. Flash memory addresses for custom oscillator-frequency trim settings

Address	Register	7	6	5	4	3	2	1	0
0x(00)00_03FD	Storage of other custom settings	—	—	—	—	—	—	—	—
0x(00)00_03FE	Storage of FTRIM	0	0	0	0	0	0	0	FTRIM
0x(00)00_03FF	Storage of TRIM	TRIM							

To override the factory-programmed settings with custom settings:

1. Using Freescale's BDM tools, users and third parties can reprogram the TRIM and FTRIM values stored in the reserved flash memory addresses.
2. User code must copy the value of FTRIM to the MCG's C4[SCFTRIM] bit and the value of TRIM to the MCG's C3[SCTRIM] field.

3.4.2 OSC Configuration

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

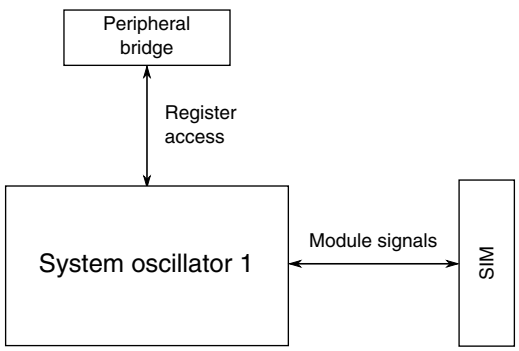


Figure 3-13. OSC1 configuration

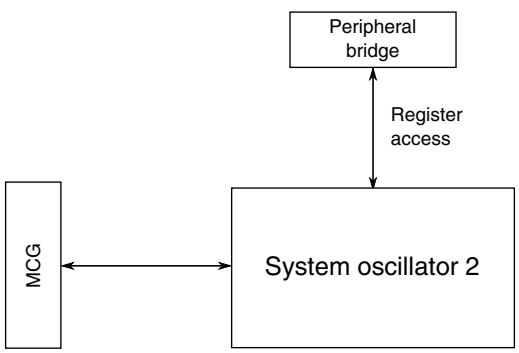


Figure 3-14. OSC2 configuration

Table 3-19. Reference links to related information

Topic	Related module	Reference
Full description	OSC	OSC
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
OSC1 configuration	System Integration Module (SIM)	SIM
OSC2 configuration	Multipurpose Clock Generation (MCG)	MCG

3.4.2.1 Real Time Clock Overview

This device includes the Real Time Clock to allow operation during all power modes. RTC includes on-chip calendaring with alarm functionality and can be clocked from the Internal 32 kHz Reference Clock (MCGIRCLK) or external 32 kHzcrystal.

NOTE

- Run the RTC from external 32 kHz oscillator when high accuracy is desired.
- RTC gets reset only on POR.
- The RTC on this device is powered by the main supply. Therefore, if the main supply is removed, the RTC contents are lost.
- This device does not support RTC in VLPS and VLLS power modes if clocked by IR clock so it is necessary to select OSC1 as RTC clock if time needs to be retained in different low power modes.
- OSC1 should be in low power mode (HGO1 = 0) to be used for RTC and LPTMR.

3.5 Memories and Memory Interfaces

3.5.1 RAM Configuration

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

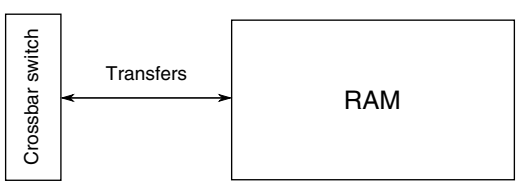


Figure 3-15. RAM configuration

Table 3-20. Reference links to related information

Topic	Related module	Reference
Description	RAM	RAM overview
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management

3.5.1.1 RAM Overview

This device includes up to 64 KB of static RAM. RAM is most efficiently accessed using the A5-relative addressing mode (address register indirect with displacement mode). Any single bit in this area can be accessed with the bit manipulation instructions (BCLR, BSET, and so on).

At power-on, the contents of RAM are uninitialized. RAM data is unaffected by any reset provided that the supply voltage does not drop below the minimum value for RAM retention (V_{RAM}).

NOTE

Since, this device includes Parity on 64 KB RAM, RAM must be written with parity before read checking can be enabled. Refer to MESM chapter for more information.

3.5.1.2 RAM sizes

The embedded RAM is tightly coupled with the V1 ColdFire core. The following table describes the amount of RAM (not counting FlexRAM) for the devices covered by this document.

Chip	RAM (KB)
MCF51256	64

The RAM has partitions that operate as a single unit:

- RAM1: 1 KB partition
- RAM2: 31 KB partition
- RAM3: 32 KB partition

3.5.1.3 RAM Retention in Low Power Modes

The RAM1, RAM2, and RAM3 partitions are retained in low power modes down to VLLS3 mode.

In VLLS2 mode: The RAM1 partition is powered, the RAM2 partition is optionally powered using the VLLSCTRL[RAM2PO] bit in system mode controller (SMC), and the RAM3 partition is not powered.

In VLLS1 mode: The RAM1, RAM2, and RAM3 partitions are not powered. However, the 32-byte register file remains available in VLLS1 mode.

3.5.1.4 RAM accesses

The RAM's interface with the crossbar switch is 32 bits wide.

3.5.2 Flash Memory Controller Configuration

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

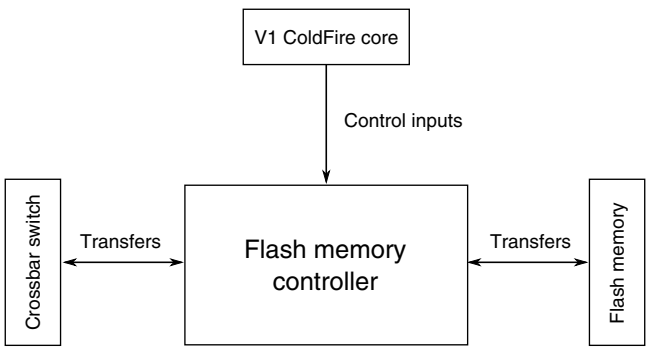


Figure 3-16. Flash memory controller configuration

Table 3-21. Reference links to related information

Topic	Related module	Reference
Full description	Flash memory controller	Flash memory controller
System memory map		System memory map
Clocking		Clock distribution
Transfers	Flash memory	Flash memory
Transfers	Crossbar switch	Crossbar switch
Control inputs	V1 ColdFire core's CPU configuration register (CPUCR)	CPUCR

3.5.3 Flash Memory Configuration

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

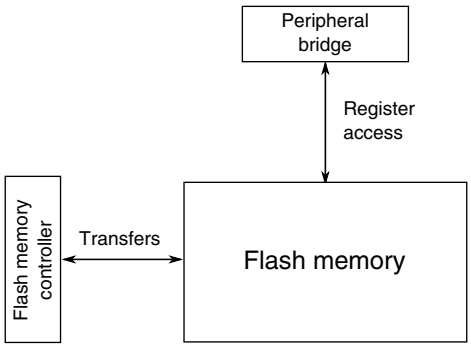


Figure 3-17. Flash memory configuration

Table 3-22. Reference links to related information

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

3.5.3.1 Flash Memory Types

This device contains multiple types of flash memory as defined below:

- Program flash: nonvolatile flash memory that can execute program code
- FlexMemory: memory block that can be configured as additional program flash, data flash, and/or EEPROM. It allows user configuration of the EEPROM and data flash sizes, as well as EEPROM endurance, to fulfill application requirements.
  - FlexNVM: nonvolatile flash memory that can execute program code, store data, or back up EEPROM data
  - FlexRAM: RAM memory that can be used as traditional RAM or as high-endurance EEPROM storage

3.5.3.2 Flash Memory Sizes

The devices covered in this document contain:

- 2 block of program flash
- 1 block of FlexNVM
- 1 block of FlexRAM

The following table describes the amounts of memory for the devices covered in this document.

Device	Program flash (KB)	FlexNVM (KB)	FlexRAM (KB)
MCF51256	256	32	2

3.5.3.3 Flash Memory Map

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

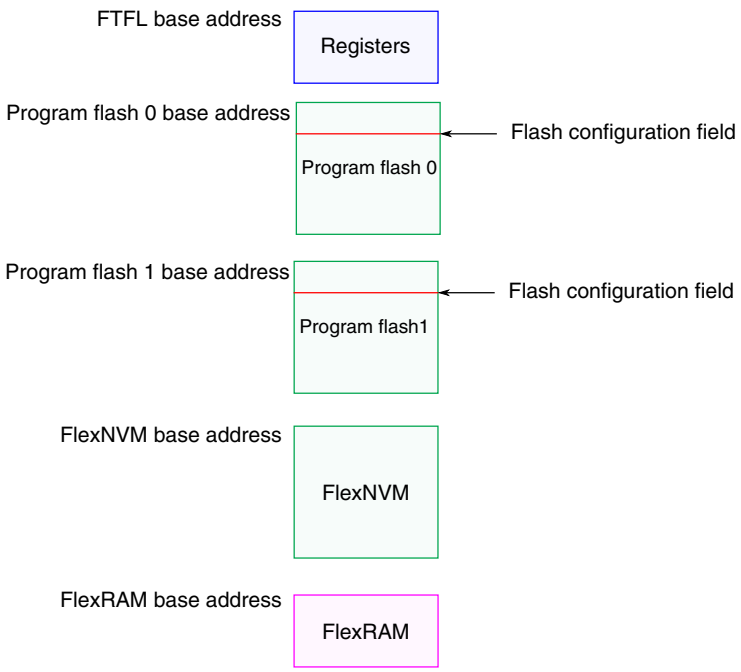


Figure 3-18. Flash memory map

The following table identifies subdivisions within the chip's program flash memory space. It also indicates where to find additional information about each area.

Table 3-23. High Level Program Flash Memory Map

Address range	Purpose	Reference
0x(00)00_0000 to 0x(00)00_03FC	Standard program flash memory, interrupt vector table	Interrupt Channel Assignments
0x(00)00_03FD to 0x(00)00_03FF	Space for custom oscillator-frequency trim settings	MCG oscillator-frequency trim settings: factory and custom
0x(00)00_0400 to 0x(00)00_040F	Flash Configuration Field	Flash Configuration Field Description
0x(00)00_0410 to upper limit	Standard program flash memory	

3.5.3.4 Flash Security

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

3.5.3.5 Flash Modes

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

3.5.3.6 Erase All Flash Contents

In addition to software, the entire flash memory may be erased external to the flash memory in two ways:

1. Via the EzPort by issuing a bulk erase (BE) command provided mask erase from flash is enabled.
2. Via background debug by using DBGCR[0]. Refer to DBGCR for details.

3.5.4 System Register File Configuration

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

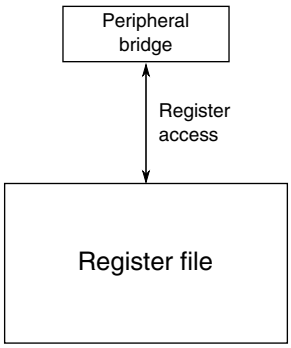


Figure 3-19. System Register file configuration

Table 3-24. Reference links to related information

Topic	Related module	Reference
Full description	Register file	Register file
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management

3.5.4.1 Register file details

The chip includes a 32-byte register file, consisting of eight 32-bit registers, that is accessible in all power modes and retains contents during low-voltage detect (LVD) events.

The register file can be accessed via 8-bit, 16-bit, and 32-bit accesses. The 16-bit and 32-bit accesses are serialized on the 8-bit peripheral bus.

The register file is reset exclusively by the POR Only reset type. It is unaffected by other reset types. For information about the various reset types on this chip, refer to the Reset details.

3.5.5 EzPort Configuration

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

Figure 3-20. EzPort configuration

Table 3-25. Reference links to related information

Topic	Related module	Reference
Full description	EzPort	EzPort
System memory map		System memory map
Clocking		Clock distribution
Transfers	Crossbar switch	Crossbar switch
Signal multiplexing	Port mux control	Signal multiplexing

3.5.5.1 EzPort and BDM

EzPort mode and active background debug mode (BDM) cannot be used at the same time. Attempts to use both simultaneously can lead to unexpected behavior.

BDM has priority over EzPort mode. For more information, refer to the detailed Boot description.

3.5.5.2 Flash Option Register (FOPT)

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

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

The FOPT register is loaded from the flash option byte. If the flash option byte is modified the new value takes effect for any subsequent resets, until the value is changed again. For FOPT register bit definitions, refer to FOPT boot options.

For more information about the FOPT register, refer to FOPT boot options.

3.5.5.3 EzPort Clocking

The EzPort module is enabled only when the device is operating in EzPort mode. The module clocks are active only in this mode. When the device is operating in normal mode, the EzPort clock is disabled.

No register bits control the ezPort module clock because the clocking is determined by operating mode.

3.6 Security Modules

3.6.1 CAU Configuration

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

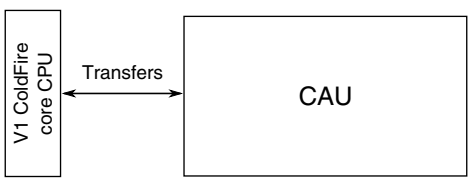


Figure 3-21. CAU configuration

Table 3-26. Reference links to related information

Topic	Related module	Reference
Full description	CAU	CAU
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Transfers	V1 ColdFire core CPU	Core

3.6.2 RNGA Configuration

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

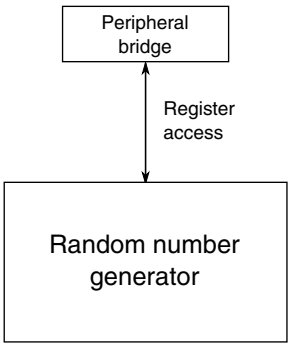


Figure 3-22. RNGA configuration

Table 3-27. Reference links to related information

Topic	Related module	Reference
Full description	RNGA	RNGA
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management

3.6.2.1 Module register width and serialization of accesses

This module's registers are 32 bits wide. Accesses via the 8-bit peripheral bus are serialized: 32-bit accesses are serialized into four 8-bit accesses.

3.6.3 CRC Configuration

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

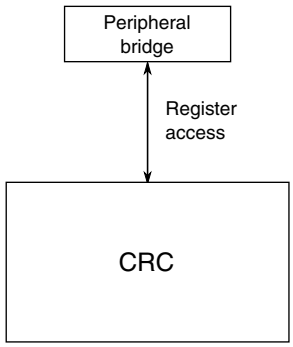


Figure 3-23. CRC configuration

Table 3-28. Reference links to related information

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

3.6.3.1 Module register width and serialization of accesses

This module's registers are 32 bits wide. Accesses via the 8-bit peripheral bus are serialized: 32-bit accesses are serialized into four 8-bit accesses.

3.7 Timers

3.7.1 FlexTimer Configuration

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

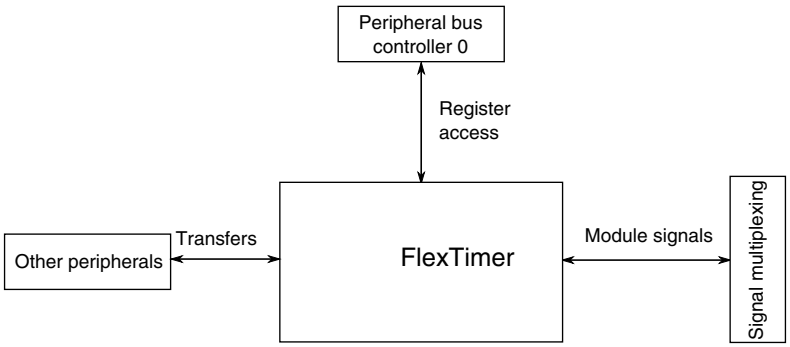


Figure 3-24. FlexTimer configuration

Table 3-29. Reference links to related information

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

3.7.1.1 Instantiation Information

The device contains two FlexTimer modules.

The following table shows how these modules are configured.

Table 3-30. FTM instantiations

FTM Instance	Number of Channels	Features/Usage
FTM1	6	General Purpose/3-phase motor
FTM0	2	general purpose

Table 3-31. FTM fault inputs source

FTMx Fault Input	FTM0 Fault Source	FTM1 Fault Source
Fault0	FTM_FLT0 bit	FTM_FLT0 bit
Fault1	FTM_FLT1 bit	FTM_FLT1 bit
Fault 2	FTM_FLT2 bit	FTM_FLT2 bit
Fault 3	FTM_FLT3 bit	FTM_FLT3 bit

Table 3-32. FTM trigger sources

Trigger Source	FTMx Synchronization
FTMxSYNC bits (in SIM)	FTM0 and FTM1 Trigger0
FTM_FLT3 bit	FTM0 and FTM1 Trigger1
FTM0CH0	FTM1 Trigger 2
FTM1CH0	FTM0 Trigger 2

Both the FTMs (FTM0 and FTM1) have flexibility to be clocked from internal as well use an external clock input (TMR_CLKIN0). Some of the FTM channels are available at multiple pins for maximum user flexiblity.

3.7.1.2 FTM External Clock Options

By default, each FTM module is clocked by the internal bus clock (the FTM refers to it as system clock). Each module has a register setting that allows the module to be clocked from an external clock instead.

3.7.1.3 FTM registers classification

Address	Registers classification
FFFF_8300 - FFFF_8316 FFFF_8340 - FFFF_8356	TPM registers
FFFF_8320 - FFFF_8332 FFFF_8360 - FFFF_8372	FTM registers

3.7.2 Low-Power Timer (LPTMR) Configuration

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

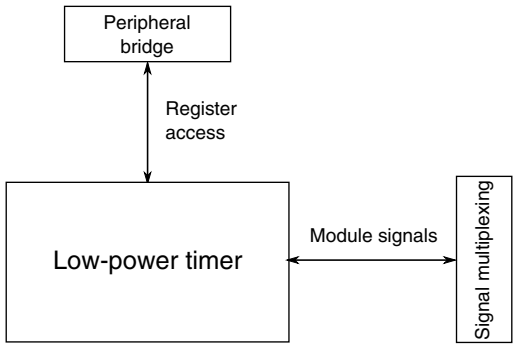


Figure 3-25. LPTMR configuration

Table 3-33. Reference links to related information

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

3.7.2.1 LPTMR Overview

This device contains one 16-bit Low Power Timer (LPTMR) module to allow operation during all power modes (VLLSx). The LPTMR is a 16-bit timer that can operate as either a Real Time Interrupt or as a Pulse Accumulator. It includes a 4-bit prescaler (Real Time Interrupt mode) or glitch filter (Pulse Accumulator mode) and can be clocked from the Internal Reference Clock, external reference clock or the internal 1 kHz LPO. An interrupt is generated (and the counter can reset) when the counter equals the value in the 16-bit compare register.

NOTE

- The Internal Reference Clock (IRC) is generated by the MCG module and can be set to be in the 30 kHz range or 2 MHz.
- The hardware trigger is not supported in this device. Please ignore all references in LPTMR chapter.

3.7.2.2 Instantiation Information

The device contains one LPTMR module with one 16-bit channel.

The LPT_ALT0 pin is connected to LPTMR0 and can be selected by the external input selection bits.

LPTMR0 module can use either the OSC1 (32.768 kHz clock) or OSC2 as an external clock based on SOPT4[LPT0CS] bit configuration in SIM module. The routing of OSC1 and LPTMR0 should be optimized for minimum power consumption in Stop modes.

Table 3-34. LPTMR pin selection in Pulse Counter mode

LPTMRx_CSR[TPS[1:0]]	Description
00	Pin LPT_ALT0 is selected
01	Reserved
10	Reserved
11	Reserved

Table 3-35. LPTMR0 clock inputs

LPTMRx_PSR[PCS[1:0]]	Description
00	Internal Reference Clock
01	Internal 1 kHz LPO
10	External 32 kHz Crystal (OSC1_32KCLK or OSC2_32KCLK[1])
11	External Reference Clock (OSC2ERCLK)

1. Depends on SOPT4[LPT0CS] bit configuration in SIM module.

3.7.2.3 LPTMR register reset

All LPTMR registers are reset by Chip POR not VLLS and by POR Only, which triggers Chip POR not VLLS. LPTMR registers are unaffected by other reset types. For information about the various reset types on this chip, refer to the Reset details.

3.7.3 Modulo Timer Configuration

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

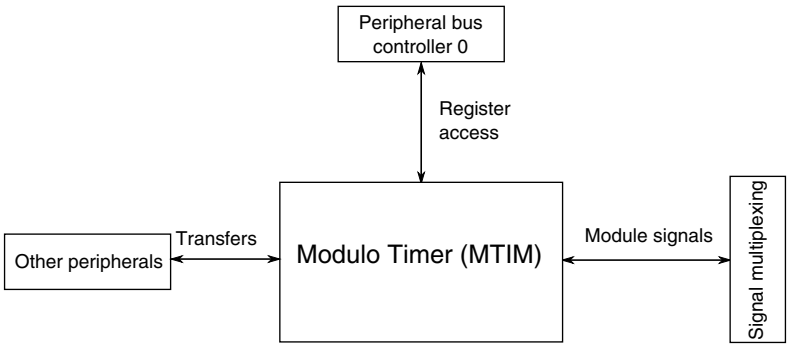


Figure 3-26. Modulo Timer configuration

Table 3-36. Reference links to related information

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

3.7.3.1 MTIM overview

The MTIM is a simple 16-bit modulo timer with several software selectable clock sources and a programmable interrupt. The chip has one MTIM module.

The MTIM shares the TMR_CLKIN pin with the FlexTimer modules.

3.8 Communication Interfaces

3.8.1 SPI Configuration

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

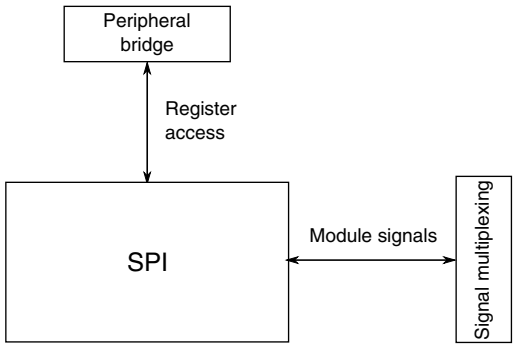


Figure 3-27. SPI configuration

Table 3-37. Reference links to related information

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

3.8.1.1 SPI Instantiation

The device contains three SPI modules. SPI0 has a 64-bit FIFO; SPI1 and SPI2 do not have a FIFO support. All three SPI module instances have DMA support.

3.8.1.2 SPI Baud Rate

The SPI is designed to run at a baud rate up to the bus clock divided by two when the module is in master mode and up to the bus clock divided by four when the module is in slave mode.

The device's maximum CPU frequency is 50 MHz with a maximum bus clock of 25 MHz. As a result, the SPI design supports a master baud rate up to 12.5 Mbit/s and a slave baud rate up to 6.25 Mbit/s .

NOTE

- For the actual maximum SPI baud rate in master and slave modes, refer to the device's Data Sheet.
- If SPI is operating in slave mode, asserting SPI_SS can wakeup the MCU from STOP and VLPS modes.

3.8.2 IIC Configuration

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

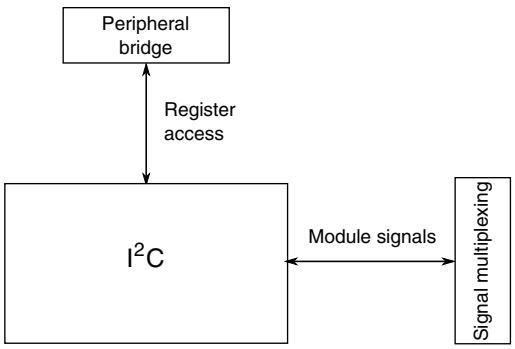


Figure 3-28. IIC configuration

Table 3-38. Reference links to related information

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

3.8.2.1 Instantiation Information

The device contains two IIC modules, IIC0 and IIC1. These modules are configured in a shared pin configuration to achieve separate master and slave interrupts. Each pin associated with an IIC module contains two instantiations of the IIC peripheral. For example, all IIC0 pins, IIC0_SDA and IIC0_SCL can be linked with IIC1 internally by setting SOPT7[IICDR0] bit in the SIM module and vice versa.

The pin mux control register in the SIM module allows IIC functionality to be selected for the individual pin. See the Signal multiplexing chapter for more information about the Pin Mux controls.

3.8.3 UART Configuration

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

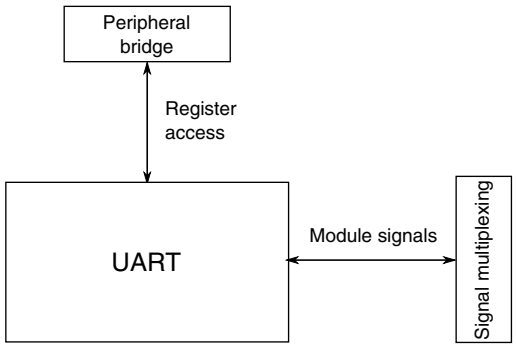


Figure 3-29. UART configuration

Table 3-39. Reference links to related information

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

3.8.3.1 Instantiation Information

The device contains three UART modules. UART0 has a 64-bit FIFO and smart card support; UART1 and UART2 do not have a FIFO and smart card support .

In the PFIFO register, the TXFIFOSIZE and RXFIFOSIZE fields both reset to 010b.

3.8.3.2 UART wakeup

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

3.8.3.3 UART support for opto-isolated interface

The PTB4 pin, to which the SCI2_TX signal can be muxed, supports high drive strength to provide an opto-isolated interface. To enable this feature, use the SOPT6[PTB4PAD] bit in the SIM.

3.9 Human-Machine Interfaces (HMI)

3.9.1 EGPIO Configuration

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

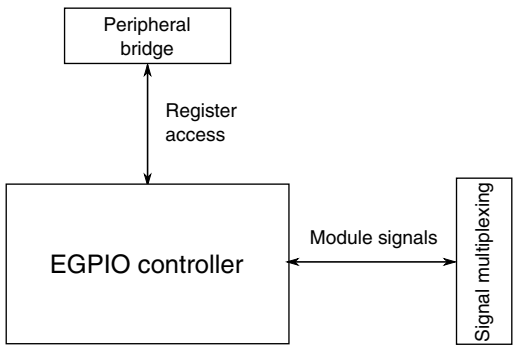


Figure 3-30. EGPIO configuration

Table 3-40. Reference links to related information

Topic	Related module	Reference
Full description	EGPIO	Parallel Input/Output Control
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port mux control	Signal multiplexing

3.9.1.1 EGPIO Overview

The EGPIO controller supports common pin input and output functions with configurable slew rate, drive strength, pull ups/downs, and passive input filters. This module combines keyboard interrupt functionality for rising/falling edges or level sensitivity. Digital glitch filter capability is available on Port B and Port C.

3.9.1.2 Instantiation Information

Smaller packages have fewer available GPIO pins. For more details on package pinouts, refer to the Signal Multiplexing chapter.

GPIO pins are configured in 8-pin ports. The peripheral bus interface for the EGPIO operates at the bus clock.

Only two EGPIO ports have the digital filter: Port B and Port C. The other ports do not have this feature.

PCTLD_PFE register has a reset value of 1Fh.

NOTE

Since EGPIO supports both Interrupt as well as GPIO capability, it is important to note that only one is active at a time. For the cases where EGPIO is being used for DMA enable request , system should disable EGPIO DMA capability by setting the "DMA enable bit" before entering STOP mode and interrupt enabled incase user wishes to support chip wakeup through EGPIO interrupt.

3.9.2 RGPIO Configuration

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

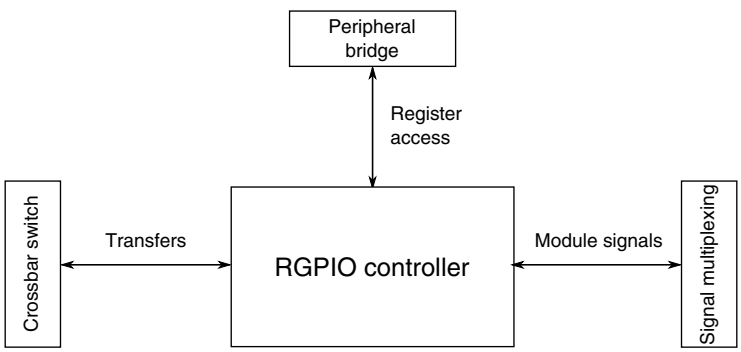


Figure 3-31. RGPIO configuration

Table 3-41. Reference links to related information

Topic	Related module	Reference
Full description	RGPIO	RGPIO
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Transfers	Crossbar switch	Crossbar switch
Signal multiplexing	Port mux control	Signal multiplexing

3.9.2.1 Instantiation Information

Smaller packages have fewer available GPIO pins. For more details on package pinouts, refer to the Signal Multiplexing chapter.

3.9.2.2 Simple Square-Wave Generation

This information replaces the information in the corresponding table of the RGPIO chapter.

Table 3-42. Square-Wave Output Performance

Loop	Peripheral Bus-mapped GPIO			RGPIO		
	Sq-Wave Frequency	Frequency @ CPU *f* = 50 MHz	Relative Speed	Sq-Wave Frequency	Frequency @ CPU *f* = 50 MHz	Relative Speed
bchg with *tpf*	(1/18) × *f* MHz	2.77 MHz	1.00x	(1/14) × *f* MHz	3.56 MHz	1.28x

bchg sample code for RGPIO:

```
move.l  #0,d0
bchg    d0,0x00C00003
tpf
bchg    d0,0x00C00003
tpf
```

Note

The square-wave frequency is measured from rising-edge to rising-edge, where the output wave has a 50% duty cycle.

The following table provides example code using which maximum frequency of 50% duty cycle square wave can be generated on RGPIO using toggle register.

Table 3-43. Example Code of 50% Duty Cycle Square-Wave

Loop	RGPIO		Code
	Sq-Wave Frequency	Frequency @ CPU *f* = 50 MHz	

Set using move instruction on toggle register (16 bit write)	(1/4) × *f* MHz	12.5 MHz	`move.l #0x00C0000E,a0`
			`move.l #1,d0`
			`nop`
			`move.w d0,(a0)`
			`tpf`
			`move.w d0,(a0)`
			`tpf`
			`move.w d0,(a0)`
			`tpf`
			`move.w d0,(a0)`
			`tpf`
			`move.w d0,(a0)`
			`tpf`
			`move.w d0,(a0)`
			`tpf`
			`move.w d0,(a0)`
			`tpf`

3.9.3 External Interrupt (IRQ) Module Configuration

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

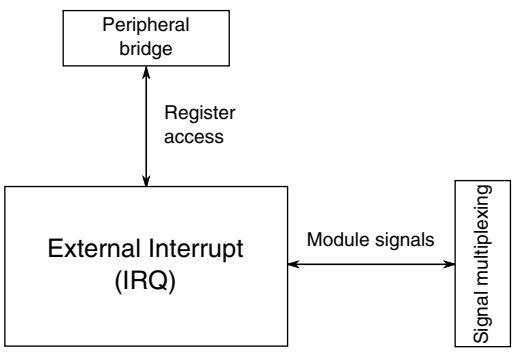


Figure 3-32. IRQ module configuration

Table 3-44. Reference links to related information

Topic	Related module	Reference
Full description	External Interrupt (IRQ)	IRQ
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port mux control	Signal multiplexing

Chapter 4
Memory Map

4.1 Introduction

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

4.2 System Memory Map

This device supports up to 256 KB of program flash and 32 KB of FlexRAM space. This configuration provides up to 288 KB of program flash and up to 2 KB of EEPROM.

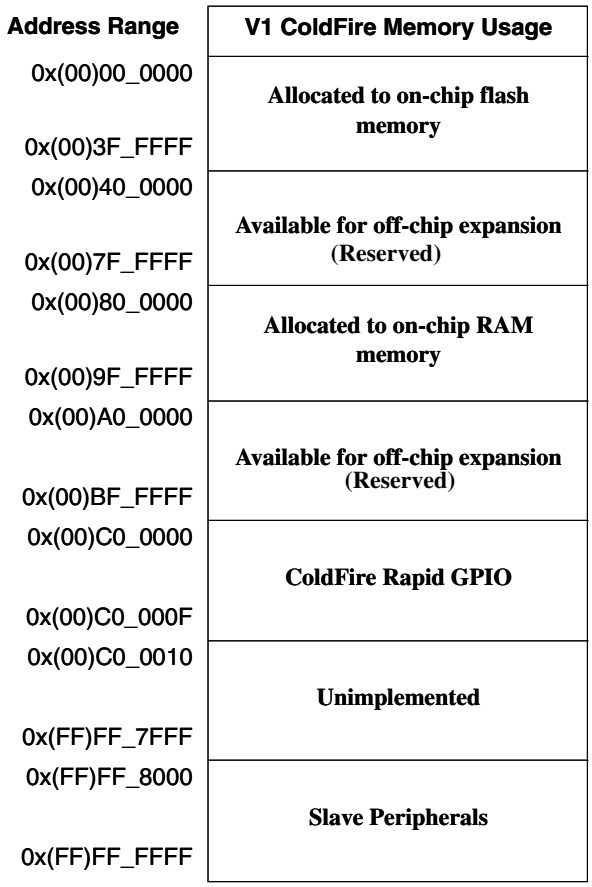


Figure 4-1. Generic V1 ColdFire Memory Map

Regions within the memory map are subject to restrictions with regard to the types of CPU accesses allowed. These are outlined in the following table. Non-supported access types terminate the bus cycle with an error (and typically generate a system reset in response to the error termination).

Table 4-1. CPU Access Type Allowed by Region

Base Address	Region	Read[1]			Write[1]		
		Byte	Word	Long	Byte	Word	Long
0x(00)00_0000	Program flash[2]	x	x	x	—	—	—
0x(00)20_0000	FlexNVM[2]	x	x	x	—	—	—
0x(00)28_0000	FlexRAM	x	x	x	x	x	x
0x(00)80_0000	RAM	x	x	x	x	x	x
0x(00)C0_0000	Rapid GPIO	x	x	x	x	x	x
0x(FF)FF_8000	8-bit Peripherals[3]	x	x	x	x	x	x

Table continues on the next page...

Table 4-1. CPU Access Type Allowed by Region (continued)

Base Address	Region	Read[1]			Write[1]		
		Byte	Word	Long	Byte	Word	Long
0x(00)40_0000/ 0x(00)A0_0000/ Reserved	Reserved/ Unimplemented	Not allowed. Any access terminated with Error					

1. Byte is 8-bit, Word 16-bit and Long 32-bit
2. Flash writes occur via a programming algorithm described in the flash memory details.
3. Allowed access types are peripheral specific. The peripheral bus bridge serializes 16 and 32-bit accesses into multiple 8-bit accesses. When using 8-bit peripherals, ensure that all accesses are properly aligned and only desired 8-bit locations are accessed.

The slave peripherals section of the memory map is further subdivided as shown in the following table.

Table 4-2. High Level System Memory Map

Memory	Description	Comment
0x(00)00_0000 - 0x(00)03_FFFF	Program Flash Space	Smaller area for any derivatives
0x(00)04_0000 - 0x(00)1F_FFFF	Unimplemented	Bus Error - Illegal address
0x(00)20_0000 - 0x(00)20_7FFF	FlexNVM Space	can be used as Program Flash if no Data Flash and EEPROM is used. Shared memory with EEPROM.
0x(00)20_8000 - 0x(00)27_FFFF	Unimplemented	Bus Error - Illegal address
0x(00)28_0000 - 0x(00)28_07FF	FlexRAM space	2K RAM available if no EEPROM is used
0x(00)28_8000 - 0x(00)7F_FFFF	Unimplemented	Bus Error -illegal address
0x(00)80_0000 - 0x(00)80_FFFF	64K RAM	On-chip 64K RAM
0x(00)81_0000 - 0x(00)BF_FFFF	Unimplemented	Bus Error -illegal address
0x(00)C0_0000 - 0x(00)C0_000F	RGPIO	16 RGPIO pins
0x(00)C0_000f - 0x(FF)FF_7FFF	Unimplemented	Bus Error - Illegal address
0x(FF)FF_8000 - 0x(FF)FF_FFFF	Peripheral Bus	See next table for info

Table 4-3. High Level Peripheral Memory Map

Peripheral	Description	Instance Name	BaseAddress	Size(Bytes)
Port I/O Module	Enhanced General Purpose I/O	PTA	0x(FF)FF_8000-0x(FF)FF_800F	16
Port I/O Module	Enhanced General Purpose I/O	PTB	0x(FF)FF_8010-0x(FF)FF_801F	16
Port I/O Module	Enhanced General Purpose I/O	PTC	0x(FF)FF_8020-0x(FF)FF_802F	16
Port I/O Module	Enhanced General Purpose I/O	PTD	0x(FF)FF_8030-0x(FF)FF_803F	16

Table continues on the next page...

Table 4-3. High Level Peripheral Memory Map (continued)

Peripheral	Description	Instance Name	BaseAddress	Size(Bytes)
Register File	Register File (32 Bytes)	RF	0x(FF)FF_8040-0x(FF)FF_805F	32
Port Mux Controls	Port Mux Controls	MXC	0x(FF)FF_8060-0x(FF)FF_807F	32
RCM	Reset Control Module(RCM)	RCM	0x(FF)FF_8080-0x(FF)FF_808F	16
LLWU	Low Leakage Wake Up Module	LLWU	0x(FF)FF_8090 - 0x(FF)FF_809F	16
PMC	Power Management Controller	PMC	0x(FF)FF_80A0- 0x(FF)FF_80AF	16
SMC	System Mode Controller	MC	0x(FF)FF_80B0 - 0x(FF)FF_80BF	16
SIM	System Integration Module	SIM	0x(FF)FF_80C0-0x(FF)FF_80FF	64
OSC	OSC Control Register	OSC1	0x(FF)FF_8100-0x(FF)FF_810F	16
OSC	OSC Control Register	OSC2	0x(FF)FF_8110-0x(FF)FF_811F	16
SCI	Serial Communications Interface (w/ FIFO)	SCI0	0x(FF)FF_8120-0x(FF)FF_813F	32
SCI	Serial Communications Interface	SCI1	0x(FF)FF_8140-0x(FF)FF_815F	32
SCI	Serial Communications Interface	SCI2	0x(FF)FF_8160-0x(FF)FF_817F	32
RESERVED			0x(FF)FF_8180-0x(FF)FF_818F	16
SPI	Serial Peripheral Interface (w/ FIFO)	SPI0	0x(FF)FF_8190-0x(FF)FF_819F	16
SPI	Serial Peripheral Interface	SPI1	0x(FF)FF_81A0-0x(FF)FF_81AF	16
SPI	Serial Peripheral Interface	SPI2	0x(FF)FF_81B0-0x(FF)FF_81BF	16
IIC	Inter-Integrated IC	IIC0	0x(FF)FF_81C0-0x(FF)FF_81CF	16
IIC	Inter-Integrated IC	IIC1	0x(FF)FF_81D0-0x(FF)FF_81DF	16
MCG	Multipurpose Clock Generator	MCG	0x(FF)FF_81E0-0x(FF)FF_81EF	16
MTIM16	16-Bit Modulo Timer	MTIM	0x(FF)FF_81F0-0x(FF)FF_81FF	16
RESERVED			0x(FF)FF_8200-0x(FF)FF_82FF	256
2-channel FTM	2-channel Flex Timer / PWM Module	FTM0	0x(FF)FF_8300-0x(FF)FF_833F	64
6-channel FTM	6-channel Flex Timer / PWM Module	FTM1	0x(FF)FF_8340-0x(FF)FF_837F	64

Table continues on the next page...

Table 4-3. High Level Peripheral Memory Map (continued)

Peripheral	Description	Instance Name	BaseAddress	Size(Bytes)
FLASH	Flash Wrapper	FTFL	0x(FF)FF_8380-0x(FF)FF_839F	32
LPTMR	Low Power Timer[1]	LPTMR0	0x(FF)FF_83A0-0x(FF)FF_83AF	16
RESERVED			0x(FF)FF_83B0-0x(FF)FF_83BF	16
CRC	Cyclic Redundancy Check Generator	CRC	0x(FF)FF_83C0-0x(FF)FF_83FF	64
Port Control	Port I/O Control Module	PCTLA	0x(FF)FF_8620-0x(FF)FF_862F	16
Port Control	Port I/O Control Module	PCTLB	0x(FF)FF_8630-0x(FF)FF_863F	16
Port Control	Port I/O Control Module	PCTLC	0x(FF)FF_8640-0x(FF)FF_864F	16
Port Control	Port I/O Control Module	PCTLD	0x(FF)FF_8650-0x(FF)FF_865F	16
RESERVED			0x(FF)FF_8660-0x(FF)FF_866F	16
RNGA	Random Number Generator Block	RNGA	0x(FF)FF_8680-0x(FF)FF_86AF	48
RESERVED			0x(FF)FF_86B0-0x(FF)FF_86BF	16
TE	FSL Test registers (non cust)	TE	0x(FF)FF_86C0-0x(FF)FF_86CF	16
RESERVED			0x(FF)FF_86D0-0x(FF)FF_87FF	304
IRQ0	External Interrupt 0	IRQ0	0x(FF)FF_8800-0x(FF)FF_880F	16
IRQ1	External Interrupt 1	IRQ1	0x(FF)FF_8810-0x(FF)FF_881F	16
IRQ2	External Interrupt 2	IRQ2	0x(FF)FF_8820-0x(FF)FF_882F	16
IRQ3	External Interrupt 3	IRQ3	0x(FF)FF_8830-0x(FF)FF_883F	16
RTC	Real Time Clock	RTC	0x(FF)FF_8840-0x(FF)FF_886F	48
RESERVED			0x(FF)FF_8870-0x(FF)FF_89FF	400
RESERVED			0x(FF)FF_8C00-0x(FF)FF_E3FF	22528
DMA	Direct Memory Access Controller	DMA	0x(FF)FF_E400-0x(FF)FF_E5FF	512
RESERVED			0x(FF)FF_E600-0x(FF)FF_E7FF	512
MESM	Miscellaneous and error status module	MESM	0x(FF)FF_E800 - 0x(FF)FF_E81F	32

Table continues on the next page...

Table 4-3. High Level Peripheral Memory Map (continued)

Peripheral	Description	Instance Name	BaseAddress	Size(Bytes)
RESERVED			0x(FF)FF_E820-0x(FF)FF_ECFF	1248
RESERVED			0x(FF)FF_EE00-0x(FF)FF_FFBF	4544
INTC	V1 ColdFire Interrupt Controller	INTC	0x(FF)FF_FFC0-0x(FF)FF_FFFF	64

1. Optimized for Low Power

This device supports a 32-bit peripheral bus for the DMA, MESM, and 8-bit peripheral bus for rest of the modules.

Table 4-4. Peripheral address Mapping Summary

Base Address Range	Peripheral Mapping
0x(FF)FF_8000 - 0x(FF)FF_DFFF	8-bit off-platform IPS peripherals
0x(FF)FF_E000 - 0x(FF)FF_ECFF	32-bit on-platform IPS peripherals
0x(FF)FF_ED00 - 0x(FF)FF_EFFF	32-bit off-platform AHB slaves
0x(FF)FF_F000 - 0x(FF)FF_FFFF	8-bit off-platform IPS peripherals

The bus bridge from the ColdFire system bus to the peripheral bus is capable of serializing 16-bit accesses into two 8-bit accesses and 32-bit access into four 8-bit accesses. This can be used to speed access to properly aligned peripheral registers. However, not all peripheral registers are aligned to take advantage of this feature.

CPU accesses to parts of the memory map marked as unimplemented result in an illegal address reset if CPUCR[ARD] = 0 or an access error exception if CPUCR[ARD] = 1.

The lower 32 KB of flash memory and slave peripherals section of the memory map are most efficiently accessed using the ColdFire absolute short addressing mode. RAM is most efficiently accessed using the A5-relative addressing mode (address register indirect with displacement mode).

NOTE

Access to the memory areas marked as reserved does not result in a bus error but could cause an unexpected behavior.

4.3 Read-after-write sequence and required serialization of memory operations

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

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

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

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

NOTE

One factor contributing to these situations is processor write buffering. The processor architecture has a programmable configuration bit to disable write buffering: CPUCR[BWD]. However, disabling buffered writes is likely to degrade system performance much more than simply performing the required memory serialization for the situations that truly require it.

Chapter 5
Clock Distribution

5.1 Introduction

The multipurpose clock generator (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.

5.2 Clock distribution

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 tradeoffs between performance and power dissipation.

Various modules have module-specific clocks that can be generated from the MCGPLLCLK 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.

5.3 High-Level Device Clocking Diagram

The high-level clock diagram is shown below.

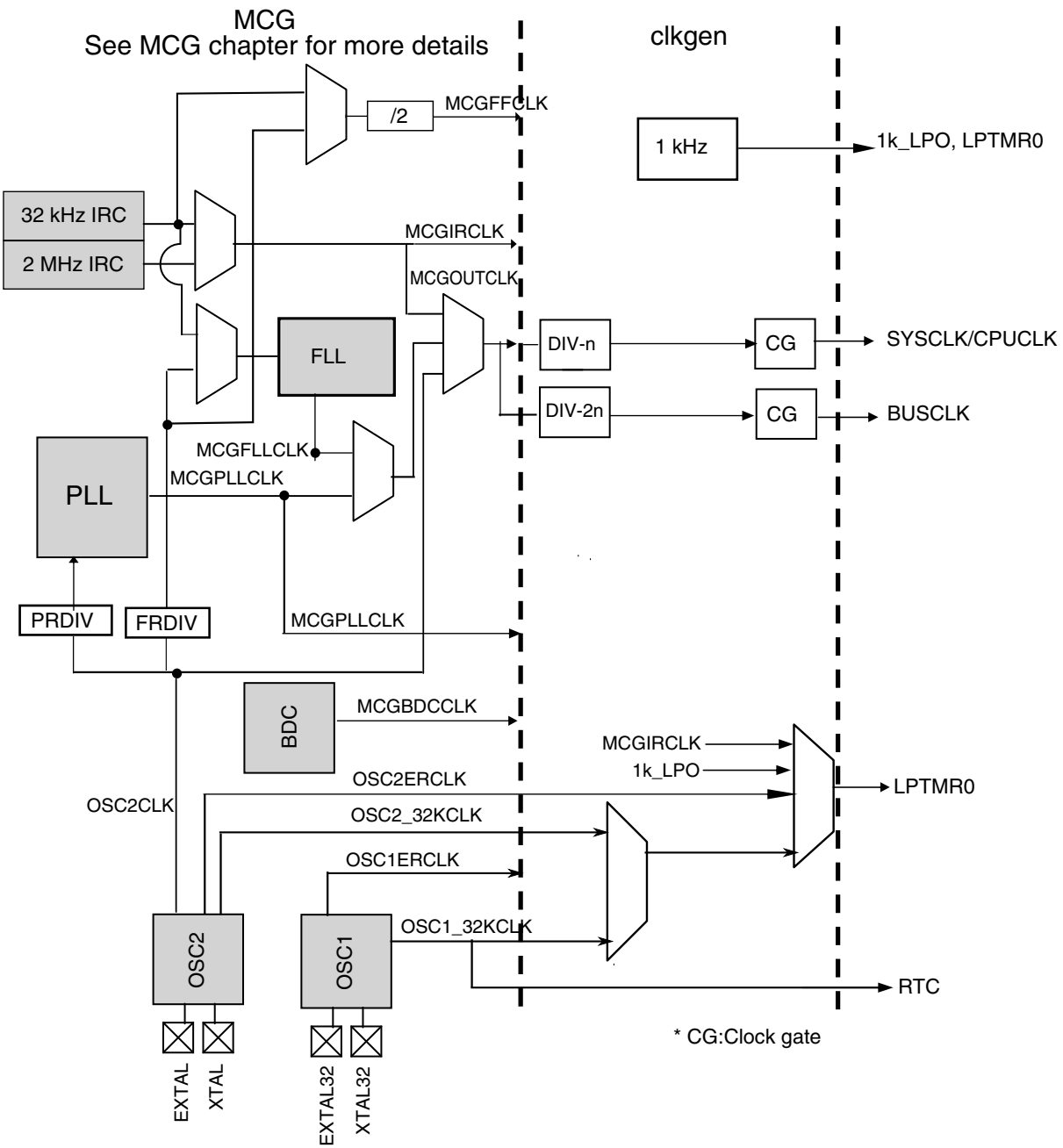


Figure 5-1. Clocking diagram

NOTE

- 1 kHz clock source resides in PMC.
- Another source for the RTC is MCGIRCLK that can be configured to use 32 kHz IRC clock.
- Source of BDC clock (not shown in the figure above can be either 4 MHz IRC or core clock.

The following table correlates the chip-level OSC signal names with the signal names used in the module's chapter.

Table 5-1. OSC clocks description

Module clock	chip clock
OSCxERCLK	OSCxERCLK
OSCx32KCLK	OSCx_32KCLK

5.4 Device Clock Summary

The following table summarizes the on-chip clocks.

Table 5-2. Input clocks for the MCG

MCG external input clock options	Description
OSC2CLK	OSC2 output, crystal, or external clock
IRC (high and low)	Internal reference clock of the MCG

Table 5-3. Device clock summary

Clock name	Run mode clock frequency	VLPR mode clock frequency	Clock source	Clock is disabled when…
MCGOUTCLK	Up to 100 MHz	Up to 2 MHz	MCG	In all stop modes
Core clock (CPUCLK)	Up to 50 MHz	Up to 2 MHz	System clock	In wait and all stop modes
System clock (SYSCLK)	Up to 50 MHz	Up to 2 MHz	MCGOUTCLK clock divider	In all stop modes
Bus clock (BUSCLK)	Up to 25 MHz	Up to 1 MHz	MCGOUTCLK clock divider	In all stop modes
Fixed fequency clock (MCGFFCLK)	Up to 6.25 MHz	Up to 0.25 MHz	MCGIRCCLK or OSC2CLK	In all stop modes
FLL output clock (MCGFLLCLK)	Up to 96 MHz	N/A	MCG	In all stop modes
PLL output clock (MCGPLLCLK)	Up to 100 MHz	N/A	MCG	PLL output clock optionally ON in Stop modes if C5[PLLSTEN] bit in MCG module is set.
Background Debug Controller clock (MCGBDCCLK)	Up to 50 MHz	N/A	MCG	In all stop modes

Table continues on the next page...

Table 5-3. Device clock summary (continued)

Clock name	Run mode clock frequency	VLPR mode clock frequency	Clock source	Clock is disabled when…
Internal reference (MCGIRCLK)	30-40 kHz or 2 MHz	2 MHz only	Internal Source	MCG IRCLKEN disabled, Stop mode and MCG IRSTEN disabled, or VLPS/VLLS mode
External reference (OSC1ERCLK)	Up to 32 kHz (bypass) 30-40 kHz (low-range crystal)	Up to 32 kHz (bypass) 30-40 kHz (low-range crystal)	OSC1	OSC1_CR[ERCLKEN] is disabled, or In all stop modes when both OSC1_CR[EREFSTEN] and SIM_OSC1[OSC1EN] are cleared
External reference (OSC1_32KCLK)	30-40 kHz (low-range crystal)	30-40 kHz (low-range crystal)	OSC1	When OSC1_CR[ERCLKEN] is cleared and SIM_OSC1[OSC1EN] is cleared, or In all stop modes when both OSC1_CR[EREFSTEN] and SIM_OSC1[OSC1EN] are cleared
External reference (OSC2ERCLK)	Up to 50 MHz (bypass) 30-40 kHz (low-range crystal) 1-32 MHz (high-range crystal)	Up to 4 MHz (bypass) 30-40 kHz (low-range crystal) Up to 4 MHz (high-range crystal)	OSC2	OSC2_CR[ERCLKEN] is disabled, or In all stop modes when OSC2_CR[EREFSTEN] is cleared
External reference (OSC2CLK, OSC2_32KCLK)	Up to 50 MHz (bypass) 30-40 kHz (low-range crystal) 1-32 MHz (high-range crystal)	Up to 4 MHz (bypass) 30-40 kHz (low-range crystal) Up to 4 MHz (high-range crystal)	OSC2	In all stop modes where OSC2ERCLK is disabled
LPO	1 kHz	1 kHz	PMC	Never

5.5 Architecture

Various clocks are generated by dividing the MCGOUTCLK:

- Core clock – Clocks the ColdFire V1 core.

- System clock – Clocks the crossbar switch and bus masters directly connected to the crossbar. In addition, this clock is used for UART0.
- Bus clock – Clocks the bus slaves and peripheral (excluding memories).

5.6 Clock divider requirements

The flash memory's LPBOOT bit of the option byte loaded to the FTFL_FOPT register controls the reset value of the core clock, system clock, and bus clock dividers. For a definition of LPBOOT and other bits of the option byte, refer to FOPT boot options.

Table 5-4. LPBOOT's effect on clock dividers

LPBOOT	SIM_CLKDIV0[OUTDIV]	Core/system clock	Bus clock	Description
0	0xF	Divide by 16	Divide by 32	Low power boot
1	0x0	Divide by 1	Divide by 2	Fast clock boot

The LPBOOT bit provides the flexibility to select a lower frequency, low power boot option. The flash erased state defaults to fast clocking mode, because where the low power boot (LPBOOT) bit resides in flash is logic 1 in the flash erased state.

To enable the low power boot option, program LPBOOT (in the option byte of the FTFL module's flash configuration field) to 0. During the reset sequence, if LPBOOT is 0, the system is in a slow clock configuration. Upon any system reset, the clock dividers return to this configurable reset state.

5.7 Clock gating

The clock to each module can be individually gated on and off using the SIM'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, disable the module or place the module in its default configuration.

Turning off the clock for an enabled, working module can produce unexpected behavior.

5.8 Module Clocks

The following table summarizes the clocks associated with each module.

Table 5-5. Module clock options

Description	Module	Input clock options
System Peripherals, Core, and Platform	CPU	CPUCLK
	CAU	CPUCLK
	RNGA	BUSCLK
	DBG	CPUCLK
	BDM	CPUCLK, MCGBDCCLK
	DMA	SYSCLK
	RAM	SYSCLK
	FLASH (and FlexMemory)	BUSCLK
External Memory Interface	CRC	BUSCLK
Timers	MTIM	BUSCLK, MCGFFCLK, TMR_CLKIN0, FTM0_CH1, FTM1_CH1
	LPTMR0	1k_LPO, OSC2ERCLK, OSC1_32KCLK, OSC2_32KCLK, MCGIRCLK
	COP	BUSCLK, 1k_LPO
	FTM0, FTM1	BUSCLK, MCGFFCLK, SYSCLK, TMR_CLKIN0
	RTC	BUSCLK, OSC1_32KCLK, MCGIRCLK
Communications	SPI0, SPI1, SPI2	BUSCLK
	UART0	SYSCLK
	UART1, UART2	BUSCLK
	IIC0, IIC1	BUSCLK, SYSCLK (Filter)
Human Machine Interface	EGPIO	BUSCLK
	RGPIO	SYSCLK

5.8.1 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 clocks are less than or equal to 2 MHz.

5.8.2 UART Clocking

UART0 module operate from the core clock, which provides higher performance level for these modules. UART1 and UART2 operates from bus clock.

5.8.3 PMC 1 kHz Clock

The PMC generates a 1 kHz clock that is alive in all modes of operation, including all low power modes (whenever PMC is powered). This 1 kHz source is commonly referred to as LPO clock or 1 kHz LPO clock.

This clock feeds the following destinations:

- LPTMR0
- EGPIO Filter (1.2 V)
- COP
- RCM reset filter
- LLWU wakeup filter

Chapter 6
Reset and Boot

6.1 Reset

This section discusses the Reset Control Module (RCM), basic reset mechanisms, and the various sources of reset on the chip. See the Reset Control Module for RCM register details.

Some modules that cause resets can be configured to cause interrupts instead. Consult the individual peripherals' chapters for more information.

6.1.1 MCU reset sources

Resetting the MCU provides a way to start processing from a known set of initial conditions.

When the ColdFire processor exits reset, it fetches initial 32-bit values for the supervisor stack pointer and program counter from locations 0x(00)00_0000 and 0x(00)00_0004, respectively.

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

During and following a reset, the input pin associated with the BKGD pin is configured with pullup enabled.

This series of devices has the following sources for reset:

- Power-on reset (POR)
- External $\overline{\text{RESET}}$ pin (PIN)
- COP watchdog (WDOG) timer
- Illegal opcode detect (ILOP)
- Illegal address detect (ILAD)
- Multipurpose Clock Generator loss of clock (LOC)

- Low-voltage detect (LVD)
- Low leakage wakeup (WAKEUP)
- Stop mode acknowledge error (SACKERR)
- EzPort RESET command (EZPT)
- Background debug forced reset (BDFR)

Each of these sources has an associated bit in the system reset status (SRS) registers.

6.1.1.1 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 (VPOR), the POR circuit causes a POR reset condition.

As the supply voltage rises, the LVD circuit holds the MCU in reset until the supply rises above the LVD low threshold (VLVDL). The RCM's SRS0[POR] and SRS0[LVD] bits are both set following a POR.

6.1.1.2 External Reset Pin (PIN)

On this device, $\overline{\text{RESET}}$ is a dedicated pin. By default, the RESET function on this pin is always enabled. This pin is open drain and has an internal pullup device and an optional digital filter. Asserting $\overline{\text{RESET}}$ wakes the device from any mode. During a pin reset, the SRS0[PIN] bit in RCM is set.

6.1.1.2.1 Reset Pin Filter

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

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

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

The 1 kHz filter has a fixed filter value of 3. Due to a synchronizer on the input data, there is also some associated latency (2 cycles). As a result, 5 cycles are required to complete a transition from low to high or high to low.

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

6.1.1.3 COP watchdog (WDOG) timer

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

6.1.1.4 Illegal opcode detect (ILOP)

By default, the V1 ColdFire core generates a MCU reset when attempting to execute an illegal instruction (except for the ILLEGAL opcode), illegal line-A instruction, illegal line-F instruction, or a supervisor instruction while in user mode (privilege violation). You may set the core's CPUCR[IRD] bit to generate the appropriate exception instead of forcing a reset.

NOTE

The attempted execution of the STOP instruction is treated as an illegal instruction if entry to stop or wait mode is disabled.

NOTE

The attempted execution of the HALT instruction is treated as an illegal instruction if the core's XCSR[ENBDM] bit is cleared.

6.1.1.5 Illegal address detect (ILAD)

By default, the V1 ColdFire core generates an MCU reset when detecting an address error, bus error termination, RTE format error, or fault-on-fault condition. If the core's CPUCR[ARD] bit is set, the processor generates the appropriate exception instead of forcing a reset, or simply halts the processor in response to the fault-on-fault condition.

6.1.1.6 Multipurpose Clock Generator loss of clock (LOC)

The MCG module supports an external reference clock.

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

NOTE

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

6.1.1.7 Low voltage detect (LVD)

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

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

6.1.1.8 Low leakage wakeup (WAKEUP)

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 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

During a reset sequence initiated via the LLWU, if neither BDM nor EzPort mode is latched, the CPU immediately fetches the LLWU interrupt service routine (ISR). Upon completion of the ISR, the CPU resumes the reset sequence.

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 specifications for more information.

6.1.1.9 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.

6.1.1.10 Background debug forced reset (BDFR)

A host debug system, when connected to the MCU via the BKGD pin, can force a background debug reset by setting the CSR2[BDFR] bit in the ColdFire debug register set. To show that this event was the source of a reset, the RCM's SRS1[BDFR] bit is 1 after the reset.

6.1.2 MCU Resets

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

6.1.2.1 POR Only

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

The POR Only reset also causes all other reset types to occur.

6.1.2.2 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).

6.1.2.3 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.

6.1.2.4 Chip Reset not VLLS

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

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

6.1.2.5 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).

6.1.2.6 Chip Reset

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

6.2 Boot

The boot description includes information about boot sources and boot options available in the MCU.

6.2.1 Boot sources

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

6.2.2 Boot options

The device's functional mode is controlled by the two mode select (MS) pins, which have the following relative priority:

1. The BKGD/MS pin is used to select between active background debug mode (BDM) and single chip modes.
2. The $\overline{\text{EZP\_MS}}$ pin selects between EzPort serial flash programming mode and single chip mode.

The mode select functionality on these pins does not depend on the pin muxing setting of the pins. Upon power on reset, these pins are analyzed and the different modes are entered depending on the level provided on these pins.

By default, the device comes out of reset in functional mode. In functional mode, the device will be in single chip (default) mode. While in single chip mode only, the device can be in run mode or various low power modes described in Modes of operation.

Table 6-1. BKGD/MS mode select decoding

Mode select (BKGD/MS)	Description
0	Active background debug mode (BDM)
1	Single chip (default)

Table 6-2. $\overline{\text{EZP\_MS}}$ mode select decoding

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

6.2.3 FOPT boot options

The flash option (FOPT) register in the flash memory module (FTFL) 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 (OPT field), refer to the detailed description of the flash memory module.

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

Table 6-3. Flash Option Register (FTFL_FOPT) Bit Definitions

Bit Num	Field	Value	Definition
7-2	Reserved		Reserved for future expansion.
1	EZPORT_EN	0	EzPort operation is disabled. If the device boots with the $\overline{\text{EZP\_MS}}$ signal low, then the device boots in normal mode. This mode might be desired if the $\overline{\text{EZP\_MS}}$/IRQ pin is used for its IRQ function.
		1	EzPort operation is enabled. The state of the $\overline{\text{EZP\_MS}}$ pin during reset determines whether the device enters EzPort mode.
0	LPBOOT	0	Low-power boot: At reset exit, the OUTDIV field's value in the SIM_CLKDIV0 register is auto-configured to 0xF for higher divide values (divide by 16) to produce lower power consumption at reset exit.
		1	Normal boot: At reset exit, the OUTDIV field's value in the SIM_CLKDIV0 register is auto-configured to 0x0 for lower divide values (divide by 1) to produce faster operating frequencies at reset exit.

6.2.4 Boot sequence

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

1. A system reset is held on internal logic, the 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, and any Bus Clocks that do not have clock gate control.
3. The system reset on internal logic continues to be held, but the Flash Controller is released from reset and begins initialization operation while the Mode Control logic continues to drive the RESET pin out low for a count of ~128 Bus Clock cycles.
4. The RESET pin and system reset on internal logic continues to be asserted while the Flash Controller continues initialization.
   a. Active background debug mode (BDM) will be selected instead of normal CPU execution if BKGD/MS is pulled low at the end of the RESET drive out low.
   b. If BDM is not selected, EzPort mode will be selected instead of normal CPU execution if EZP_MS is pulled low at the end of the RESET drive out low. EzPort mode can be disabled if the EZPORT_DIS bit of the option byte loaded to the FTFL_FOPT register is configured for the disabled setting.
5. During flash initialization, clocking is switched to a slow clock if the LPBOOT bit of the option byte loaded to the FTFL_FOPT register is configured for low power boot.
6. When flash initialization completes, the RESET pin is released. After the RESET pin is released, if RESET continues to be asserted (an indication of a slow rise time on the RESET pin or external drive in low), the system continues to be held in reset. After the RESET pin is detected high, the system is released from reset.
7. When the system exits reset, the processor fetches initial 32-bit values for the supervisor stack pointer and program counter (PC) from the locations 0x(00)00_0000 and 0x(00)00_0004, respectively. The CPU begins execution at the PC location, but at this time:
   a. If active background debug mode was latched during the sequence, BDM is entered instead of normal CPU execution.
   b. If BDM was not latched but EzPort mode was latched during the sequence, EzPort mode is entered instead of normal CPU execution.
   c. If neither BDM nor EzPort mode was latched, and if the reset event was caused by a low leakage wakeup interrupt, the CPU immediately fetches the LLWU interrupt service routine (ISR). Upon completion of the ISR, the CPU resumes execution at the PC location.
8. If FlexNVM is enabled, the Flash Controller continues to restore the FlexNVM data. This data is not available immediately out of reset, and the system should not access

this data until the Flash Controller completes this initialization step as indicated by the EEERDY flag.

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

Chapter 7
Power Management

7.1 Introduction

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

7.2 Power modes

The V1 ColdFire CPU has two primary modes of operation, Run and Stop. The STOP instruction is used to invoke both Stop and Wait modes. The CPU does not differentiate between Stop and Wait modes. Stop, Wait, and Run are augmented in a number of ways to provide a lower power MCU based on application needs.

The System Mode Controller (SMC) in this family provides the user with multiple power options. The Very Low Power Run (VLPR) operating mode can reduce run time power when maximum bus frequency is not required. Corresponding Wait and Stop modes are the Very Low Power Wait (VLPW) and Very Low Power Stop (VLPS) modes.

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.

Table 7-1. MCU power modes

Power mode	Description	Normal recovery method
Normal Run	Allows maximum performance of MCU.	-
Normal Wait	Allows peripherals to function while allowing CPU to go to sleep, reducing power.	Interrupt
Normal Stop	Places MCU in static state. Lowest power mode that retains all registers while maintaining LVD protection.	Interrupt

Table continues on the next page...

Table 7-1. MCU power modes (continued)

Power mode	Description	Normal recovery method
Very Low Power Run (VLPR)	Reduced frequency Run mode, limits CPU operation to 2 MHz (1 MHz bus, Flash access) so peripherals and CPU can continue to operate at a reduced RIDD. Regulator in low power mode, LVD off, Internal oscillator provides low power 1 MHz source for core and peripherals.	Interrupt
Very Low Power Wait (VLPW)	Similar to VLPR, with CPU in sleep to further reduce power.	Interrupt
Very Low Power Stop (VLPS)	Places MCU in static state, with LVD operation off. Lowest power mode with pin interrupts functional. LPTMR, RTC functional.	Interrupt
Very Low Leakage Stop3 (VLLS3)	Lowest power mode where all SRAM memory is retained. LLWU, LPTMR, RTC functional.	Wakeup reset
Very Low Leakage Stop2 (VLLS2)	Lowest power mode where SRAM memory subset is retained. LLWU, LPTMR, RTC functional.	Wakeup reset
Very Low Leakage Stop1 (VLLS1)	Lowest power mode where no SRAM is retained. LLWU, LPTMR, RTC functional.	Wakeup reset

For additional details, refer to Power Mode Transitions.

7.3 Module Operation in Low Power Modes

The following table illustrates the available functionality of each module while the MCU is in each of the low power modes (Stop, VLPx, and VLLSx). The module control is configured in its corresponding control registers

NOTE

In Wait mode, RNGA is static.

Table 7-2. Module operation in low power modes

Modules	Stop	VLPR	VLPW	VLPS	VLLSx
System peripherals					
CPU clock	OFF	2 MHz maximum	OFF	OFF	OFF
Bus clock	OFF	1 MHz maximum	1 MHz maximum	OFF	OFF
LLWU[1]	Static	Static	Static	Static	FF
Register File	Powered	Powered	Powered	Powered	Powered
DMA	Static	FF	FF	Static	OFF
Power management					
PMC/SMC	FF	FF	FF	FF	FF
LVD	ON	OFF	OFF	OFF	OFF
Regulator	ON	Low power	Low power	Low power	Low power
Memory and memory interfaces					

Table continues on the next page...

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

Modules	Stop	VLPR	VLPW	VLPS	VLLSx
Flash	Powered	1 MHz maximum access; no programming	Low power	Low power	OFF
RAM1 1 KB and periphery	Powered	Powered	Powered	Powered	Powered in VLLS3,2
RAM2 63 KB	Powered	Powered	Powered	Powered	Powered in VLLS3
FlexMemory	Low power	Low power[2]	Low power	Low power	OFF[3]
EzPort	OFF	OFF	OFF	OFF	OFF
Clocks					
MCG	Static: IRC optional; PLL optionally on but gated	2 MHz IRC[4]	2 MHz IRC	Static - IRCLK optional	OFF
OSCx (high range)	ERCLK optional	ERCLK limited to 4 MHz crystal	ERCLK limited to 4 MHz crystal	ERCLK limited to 4 MHz crystal	Limited to low range/low power
OSCx (32 kHz OSC)	FF	FF	FF	FF	FF
1 kHz LPO	ON	ON	ON	ON	ON
System security and integrity					
CRC	Static	FF	FF	Static	OFF
RNGA/CAU	Static	FF	Static	Static	OFF
COP	Static	FF	FF	Static	OFF
Timers					
FTM	Static	FF	FF	Static	OFF
MTIM	Static	FF	FF	Static	OFF
LPTMR0	FF	FF	FF	FF	FF
RTC	FF	FF	FF	FF	FF
Communication interfaces					
UART/SCI	Static, wakeup on edge	125 kbit/s	125 kbit/s	Static, wakeup on edge	OFF
SPI	Static	0.5 Mbit/s	0.5 Mbit/s	Static	OFF
I^2C	Static, address match wakeup	100 kbit/s	100 kbit/s	Static, address match wakeup	OFF
Human-machine interface (HMI)					
EGPIO	Wakeup	FF	FF	Wakeup	OFF, pins latched
RGPIO	Static	FF	Static	Static	OFF
IRQ(s)	Wakeup	FF	FF	Wakeup	OFF, pins latched

1. Using the LLWU module, the external pins available for this MCU do not require the associated peripheral function to be enabled. The only requirement is for the function controlling the pin (GPIO or peripheral) to be configured as an input to allow a transition to occur to the LLWU.
2. In VLPR mode, FlexRAM enabled as EEPROM is not writable (writes are ignored) but can be read. There are no access restrictions in VLPR mode for FlexRAM configured as traditional RAM.
3. FlexRAM is always powered off in VLLSx modes.
4. Before executing an entry to VLPR mode, the MCG must be in one of two of its operating modes, each with a particular clock source selected:

- Either the MCG must be in its BLPE operating mode with only the low gain oscillator selected, or
- The MCG must be in its BLPI operating mode with only the 2 MHz IRC selected.

- *ON* means the module is operational by default in the designated power mode.
- *FF* means "full functionality." The user has the option to enable the module's operation in the designated power mode. In VLPR and VLPW modes, the system frequency might limit some modules.
- *Static* means the digital modules' register states and associated memories are held.
- *Powered* means memory is powered to retain contents.
- *Low power* means flash has a low power state that retains configuration registers to support faster wakeup.
- *Wakeup* means the module can serve as a wakeup source for the chip. For more information, refer to the LLWU module's dedicated chapter and to its Chip Configuration details.
- *OFF* means the module is powered off and is in a reset state upon wakeup.

Chapter 8
Security

8.1 Introduction

The device implements security based on the mode selected from the flash module. The following sections provides an overview of flash security and effects of security on non-flash modules.

8.2 Flash Security

The flash module provides security information to the MCU based on the state held by the flash memory (FTFL) module's 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. For more information, refer to the Flash Configuration Field Description.

NOTE

The security features apply only to external accesses via debug and EzPort. CPU accesses to the flash are not affected by the status of FSEC.

In the unsecured state, all flash commands are available to the programming interfaces (BDM and EzPort) and so is user code execution of Flash Controller commands. When the flash is secured, programmer interfaces have no access to memory locations and are allowed only to launch mass erase operations.

Further information regarding the flash security options and enabling/disabling flash security is available in the detailed description of the flash memory module. Further information regarding the flash security options and enabling/disabling flash security in BDM is available in the detailed description of V1 ColdFire debug.

8.3 Flash Security Options

In addition to enabling and disabling security the flash security byte configures various security options.

8.3.1 Backdoor Key Access

The FSEC[KEYEN] field's setting allows flash security to be temporarily disabled by entering an 8-byte key value.

Table 8-1. Flash Key Enable States

KEYEN[1:0]	Status of Backdoor Key Access
00	Disabled
01	Disabled (preferred setting to disable backdoor key access)
10	Enabled
11	Disabled

8.3.2 Freescale Factory Access

The FSEC[FSLACC] field's setting enables or disables access to 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 testing must begin with a full erase operation 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 memory contents. The state of the FSLACC bits is only relevant when the SEC bits are set to secure. When the SEC field is set to unsecure, the FSLACC setting does not matter.

Table 8-2. Freescale Factory Access States

FSLACC[1:0]	Freescale Factory Access
00	Granted
01	Denied
10	Denied
11	Granted

8.3.3 Mass Erase Disable

Setting the FSEC[MEEN] field to 10 disables the Mass Erase Command.

This setting only takes effect when the flash is in a secure state. In secure mode, ERSALL and RD1ALL are the only commands available, they are both disabled when the FSEC[MEEN] field is 10.

CAUTION

Disabling the mass erase capability might be irreversible. Simultaneously setting MEEN to disabled, KEYEN to disabled, and FSLACC to denied effectively locks the current security state permanently and cannot be undone.

Table 8-3. Mass Erase Enable States

MEEN[1:0]	Mass Erase Capability
00	Enabled
01	Enabled
10	Disabled
11	Enabled

8.4 Enabling Flash Security

Flash security can be enabled by programming the security byte of the flash configuration field. This assumes that flash is currently unsecured and that the region of the P-Flash containing the Flash Configuration field is unprotected. If the Flash security byte is successfully programmed, its new value takes affect after the next MCU reset.

NOTE

Once enabled, the security options cannot be modified without disabling security first. The backdoor key access enable, 8-byte backdoor key, and Freescale access bits should be programmed to the desired settings before enabling security.

8.5 Unsecuring the MCU using Backdoor Key Access

The MCU may 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. If the KEYEN[1:0] bits are in the enabled state, the Verify Backdoor

Access Key command can be run. This command allows the user to present prospective keys for comparison to the stored keys. If the keys match, the SEC bits in the FSEC register change to unsecure the MCU until the next reset.

NOTE

The entire 8-byte key cannot be 0000_0000_0000_0000h or FFFF_FFFF_FFFF_FFFFh. These values 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[1:0] bits are in the enabled state, the MCU can be unsecured by this backdoor key access sequence:

1. Execute the Verify Backdoor Access Key command with the 8-byte key value loaded into the flash memory module's FCCOB0 to FCCOB7 registers.
2. If the Verify Backdoor Access Key command is successful, the MCU is unsecured and the SEC[1:0] bits in the FSEC register are forced to the unsecure state.
3. If the backdoor keys do not match, security is not released, all future attempts to execute the Verify Backdoor Access Key command are immediately aborted, and the ACCERR bit in FSTAT is (again) set to 1 until a power down (cold) reset occurs.

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. After the next reset of the MCU, the security state of the flash memory module reverts 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 in the flash protection (FPROT) registers.

If the Backdoor Keys successfully match, the unsecured MCU has full control of the contents of the Flash Configuration Field. The MCU may erase the sector containing the Flash Configuration Field, reprogram the Flash security byte to the unsecure state, and change the Backdoor keys to any desired value.

8.6 Unsecuring the MCU using the Erase All Blocks Command

The Erase All Blocks operation erases entirety of the program flash memory, FlexNVM, and FlexRAM used as EEPROM; verifies the erase; and then releases MCU security.

When the chip is in BDM, the functionality of the Erase All Blocks command is also available in an uncommanded fashion using the DGBCR[0] bit. When invoked, this function erases all program flash memory, FlexNVM, and FlexRAM used as EEPROM regardless of the protection settings. This function, like the normal command activation, also initializes all locations in the FlexRAM used as EEPROM to the address FFFFh. If the Flash Erase Verify passes, the routine then releases security by setting the FSEC[SEC] field to the unsecure state.

8.7 Security Interactions with other Modules

The flash security settings are used by the MCU to determine what resources are available. The following sections describe the interactions between modules and the flash security settings or the impact that the flash security has on non-flash modules.

8.8 Security Interactions with ezPort

When flash memory security is active, the MCU can still boot in EzPort mode. The EzPort module can execute an erase all blocks command (mass erase). The mass erase can be used to disable flash memory security, but all of the flash memory's contents are lost in the process.

8.9 Security Interactions with Debug

When flash memory security is active, the BDM port cannot be used to access the MCU's internal flash memory. Scan chain operations work, but debugging capabilities are disabled so that the debug port cannot read out flash contents.

Although most debug functions are disabled when flash memory security is active, the debug port can be used to execute an erase all blocks command (mass erase).

Chapter 9
Signal Multiplexing and Signal Descriptions

9.1 Introduction

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

The Port Mux Control block controls which signal is present on the external pin. Refer to that chapter to find which register controls the operation of a specific pin.

9.2 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.

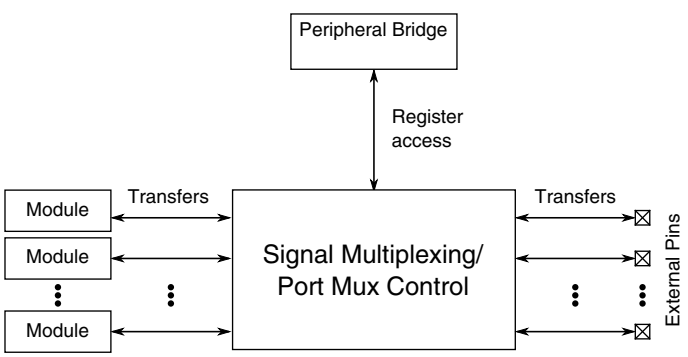


Figure 9-1. Signal multiplexing integration

Table 9-1. Reference links to related information

Topic	Related module	Reference
Full description	Port control	Port Mux Control
System memory map		System memory map

Table continues on the next page...

Table 9-1. Reference links to related information (continued)

Topic	Related module	Reference
Clocking		Clock distribution
Register access	Peripheral Bridge	

9.3 Port Mux Control Features

Not all pins described in the port mux are available on every package in the device family. Do not change the port mux control field for pins that do not appear on the device in use. Changing the port mux control field for such unavailable pins can generate unnecessary current leakage.

Table 9-2. Port mux control registers overview

Address	Peripheral	Register	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8060	MXC	PTAPF0	A7				A6			
0x(FF)FF_8061	MXC	PTAPF1	A5				A4			
0x(FF)FF_8062	MXC	PTAPF2	A3				A2			
0x(FF)FF_8063	MXC	PTAPF3	A1				A0			
0x(FF)FF_8064	MXC	PTBPF0	B7				B6			
0x(FF)FF_8065	MXC	PTBPF1	B5				B4			
0x(FF)FF_8066	MXC	PTBPF2	B3				B2			
0x(FF)FF_8067	MXC	PTBPF3	B1				B0			
0x(FF)FF_8068	MXC	PTCPF0	C7				C6			
0x(FF)FF_8069	MXC	PTCPF1	C5				C4			
0x(FF)FF_806A	MXC	PTCPF2	C3				C2			
0x(FF)FF_806B	MXC	PTCPF3	C1				C0			
0x(FF)FF_806D	MXC	PTDPF1	Reserved				D4			
0x(FF)FF_806E	MXC	PTDPF2	D3				D2			

Table continues on the next page...

Table 9-2. Port mux control registers overview (continued)

Address	Peripheral	Register	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_806F	MXC	PTDPF3	D1				D0			

9.4 Signal Multiplexing and Pin Assignments

44 MAPLGA	Pin Name	DEFAULT	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	EZPORT
1	VDD	VDD	VDD						
2	NC	NC	NC						
3	NC	NC	NC						
4	NC	NC	NC						
5	NC	NC	NC						
6	BKGD/ MS/ PTD4	BKGD/ MS	Disabled	PTD4	BKGD	MS			
7	EXTAL32	EXTAL32	EXTAL32						
8	XTAL32	XTAL32	XTAL32						
9	PTA1/ FTM1_CH0/ IIC1_SCL/ RGPIO1	Disabled	Disabled	PTA1	FTM1_CH0		IIC1_SCL	RGPIO1	
10	PTA2/ FTM1_CH1/ IIC1_SDA/ RGPIO2	Disabled	Disabled	PTA2	FTM1_CH1		IIC1_SDA	RGPIO2	
11	VDD	VDD	VDD						
12	VSS	VSS	VSS						
13	PTA3/ FTM1_CH2/ CLKOUT/ RGPIO3	Disabled	Disabled	PTA3	FTM1_CH2	CLKOUT		RGPIO3	
14	PTA5/ SCI2_CTS_b/ FTM1_CH4/ RGPIO5	Disabled	Disabled	PTA5	SCI2_CTS_b	FTM1_CH4		RGPIO5	
15	PTA6/ RGPIO6/ EZP_DO	Disabled	Disabled	PTA6				RGPIO6	EZP_DO
16	PTA7/ LPT_ALT0/ RGPIO7/ EZP_DI	Disabled	Disabled	PTA7	LPT_ALT0			RGPIO7	EZP_DI

44 MAPLGA	Pin Name	DEFAULT	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	EZPORT
17	PTB0/ RGPIO8/ EZP_CLK	Disabled	Disabled	PTB0				RGPIO8	EZP_CLK
18	IRQ0/ PTB1/ EZP_MS_b/ RGPIO9/ EZP_CS_b	IRQ0	IRQ0	PTB1	EZP_MS_b			RGPIO9	EZP_CS_b
19	PTB2/ SCI0_RTS_b/ FTM0_CH1/ IIC0_SCL/ RGPIO10	Disabled	Disabled	PTB2	SCI0_RTS_b	FTM0_CH1	IIC0_SCL	RGPIO10	
20	PTB3/ SCI0_CTS_b/ FTM0_CH0/ IIC0_SDA/ RGPIO11	Disabled	Disabled	PTB3	SCI0_CTS_b	FTM0_CH0	IIC0_SDA	RGPIO11	
21	PTB4/ SCI2_TX/ SPI2_MOSI/ RGPIO12	Disabled	Disabled	PTB4	SCI2_TX	SPI2_MOSI		RGPIO12	
22	VSS	VSS	VSS						
23	VDD	VDD	VDD						
24	PTB5/ SCI2_RX/ SPI2_MISO/ RGPIO13	Disabled	Disabled	PTB5	SCI2_RX	SPI2_MISO		RGPIO13	
25	PTB6/ SPI2_CLK/ RGPIO14	Disabled	Disabled	PTB6		SPI2_CLK		RGPIO14	
26	PTB7/ SCI1_TX/ SPI1_MOSI/ RGPIO15	Disabled	Disabled	PTB7	SCI1_TX	SPI1_MOSI		RGPIO15	
27	PTC0/ SPI2_SS	Disabled	Disabled	PTC0		SPI2_SS		Disabled	
28	IRQ3/ PTC1/ FTM1_CH5/ CLKOUT	IRQ3	IRQ3	PTC1	FTM1_CH5	CLKOUT			
29	PTC2/ SCI1_RX/ SPI1_MISO	Disabled	Disabled	PTC2	SCI1_RX	SPI1_MISO			
30	PTC3/ SPI1_CLK/	Disabled	Disabled	PTC3		SPI1_CLK			
31	IRQ1/ PTC4/ SCI1_RTS_b	IRQ1	IRQ1	PTC4	SCI1_RTS_b				
32	IRQ2/ PTC5/	IRQ2	IRQ2	PTC5	SCI1_CTS_b	TMR_CLKIN0			

44 MAPLGA	Pin Name	DEFAULT	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	EZPORT
	SCI1_CTS_b/ TMR_CLKIN0								
33	VDD	VDD	VDD						
34	VSS	VSS	VSS						
35	NC	NC	NC						
36	RESET_B	RESET_B	RESET_B						
37	EXTAL	EXTAL	EXTAL						
38	XTAL	XTAL	XTAL						
39	PTC7/ SPI1_SS	Disabled	Disabled	PTC7		SPI1_SS			
40	PTD0/ SPI0_SS	Disabled	Disabled	PTD0	SPI0_SS				
41	PTD1/ SCI0_TX/ SPI0_MOSI	Disabled	Disabled	PTD1	SCI0_TX	SPI0_MOSI			
42	PTD2/ SCI0_RX/ SPI0_MISO	Disabled	Disabled	PTD2	SCI0_RX	SPI0_MISO			
43	PTD3/ SPI0_CLK	Disabled	Disabled	PTD3		SPI0_CLK			
44	VSS	VSS	VSS						

9.5 Pinout Diagram

The below figure shows the pinout diagram for the devices supported by this document. Many signals may be multiplexed onto a single pin.

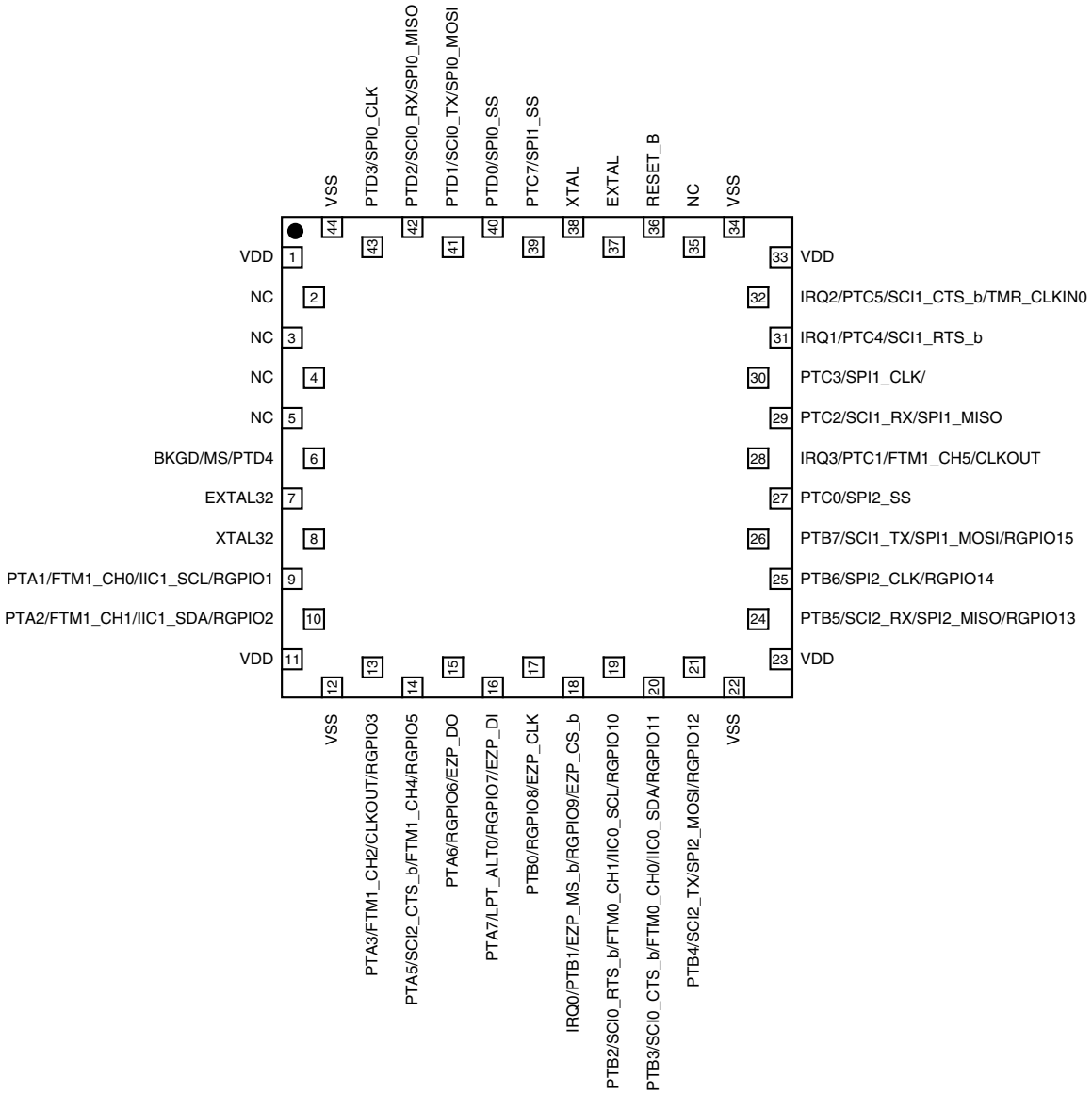


Figure 9-2. MCF51QW256 44-pin MAPLGA Pinout Diagram

NOTE

PTD4 is adjacent to EXTAL32 and may inject coupling noise on EXTAL32 when both PTD4 as well as EXTAL32 is being used. So it is highly recommended that PTD4 be only used for the cases where EXTAL32 is not used at all and system relies on internal RC clock or external EXTAL clock.

9.6 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.

9.6.1 Core Modules

Table 9-3. Background Debug Signal Descriptions

Chip signal name	Module signal name	Description	I/O
BKGD	BKGD	Single-wire background debug interface pin	I/O

9.6.2 System Modules

Table 9-4. System Signal Descriptions

Chip signal name	Module signal name	Description	I/O
$\overline{\text{RESET}}$	—	Reset input signal	I
VDD	—	MCU power	I
VSS	—	MCU ground	I
CLKOUT	—	External clock reference	O

9.6.3 Clock Modules

Table 9-5. OSC1 (MHz OSC) Signal Descriptions

Chip signal name	Module signal name	Description	I/O
EXTAL	EXTAL	External clock/Oscillator input	I
XTAL	XTAL	Oscillator output	O

Table 9-6. OSC2 (RTC OSC) Signal Descriptions

Chip signal name	Module signal name	Description	I/O
EXTAL32	EXTAL32	External clock/Oscillator input	I
XTAL32	XTAL32	Oscillator output	O

9.6.4 Memories and Memory Interfaces

Table 9-7. EzPort Signal Descriptions

Chip signal name	Module signal name	Description	I/O
EZP_CLK	EZP_CK	EzPort Clock	Input
EZP_CS_b	$\overline{\text{EZP\_CS}}$	EzPort Chip Select	Input
EZP_DI	EZP_D	EzPort Serial Data In	Input
EZP_DO	EZP_Q	EzPort Serial Data Out	Output
EZP_MS_b	—	Selects between EzPort serial flash programming mode and single chip mode	I

9.6.5 Timer Modules

Table 9-8. FTM0 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
TMR_CLKIN[0]	EXTCLK	External clock – FTM external clock can be selected to drive the FTM counter.	I
FTM0_CH[1:0]	CHn[1]	Channel (n) – I/O pin associated with FTM channel (n).	I/O

1. n = channel number (0 to 7)

Table 9-9. FTM1 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
TMR_CLKIN[0]	EXTCLK	External clock – FTM external clock can be selected to drive the FTM counter.	I
FTM1_CH[5:0]	CHn[1]	Channel (n) – I/O pin associated with FTM channel (n).	I/O

1. n = channel number (0 to 7)

Table 9-10. MTIM Signal Descriptions

Chip signal name	Module signal name	Description	I/O
TMR_CLKIN[0]	TCLK	External clock source input into MTIM16	I

Table 9-11. LPTMR Signal Descriptions

Chip signal name	Module signal name	Description	I/O
LPT_ALT0	LPTMR_ALT*n*	Pulse Counter Input pin	I

9.6.6 Communication Interfaces

Table 9-12. SPI0 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
SPI0_MISO	MISO	Master Data In, Slave Data Out	I/O
SPI0_MOSI	MOSI	Master Data Out, Slave Data In	I/O
SPI0_SCLK	SPSCK	SPI Serial Clock	I/O
SPI0_SS	$\overline{SS}$	Slave Select	I/O

Table 9-13. SPI1 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
SPI1_MISO	MISO	Master Data In, Slave Data Out	I/O
SPI1_MOSI	MOSI	Master Data Out, Slave Data In	I/O
SPI1_SCLK	SPSCK	SPI Serial Clock	I/O
SPI1_SS	$\overline{SS}$	Slave Select	I/O

Table 9-14. SPI2 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
SPI2_MISO	MISO	Master Data In, Slave Data Out	I/O
SPI2_MOSI	MOSI	Master Data Out, Slave Data In	I/O
SPI2_SCLK	SPSCK	SPI Serial Clock	I/O
SPI2_SS	$\overline{SS}$	Slave Select	I/O

Table 9-15. I^2C0 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
I2C0_SCL	SCL	Bidirectional serial clock line of the I^2C system.	I/O
I2C0_SDA	SDA	Bidirectional serial data line of the I^2C system.	I/O

Table 9-16. I^2C1 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
I2C1_SCL	SCL	Bidirectional serial clock line of the I^2C system.	I/O
I2C1_SDA	SDA	Bidirectional serial data line of the I^2C system.	I/O

Table 9-17. SCI0/UART0 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
SCI0_CTS_b	CTS	Clear to send	I
SCI0_RTS_b	$\overline{\text{RTS}}$	Request to send	O
SCI0_TX	TXD	Transmit data	O
SCI0_RX	RXD	Receive data	I

Table 9-18. SCI1/UART1 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
SCI1_CTS_b	CTS	Clear to send	I
SCI1_RTS_b	$\overline{\text{RTS}}$	Request to send	O
SCI1_TX	TXD	Transmit data	O
SCI1_RX	RXD	Receive data	I

Table 9-19. SCI2/UART2 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
SCI2_CTS_b	CTS	Clear to send	I
SCI2_TX	TXD	Transmit data	O
SCI2_RX	RXD	Receive data	I

9.6.7 Human-Machine Interfaces (HMI)

Table 9-20. GPIO Signal Descriptions

Chip signal name	Module signal name	Description	I/O
PTA[7:5],PTA[3:1][1]	PTA[7:5],PTA[3:1]	General purpose input/output	I/O
PTB[7:0][1]	PTB[7:0]	General purpose input/output	I/O
PTC[7] PTC[5:0][1]	PTC[7] PTC[5:0]	General purpose input/output	I/O
PTD[4:0][1]	PTD[4:0]	General purpose input/output	I/O

1. The available GPIO pins depend on the specific package. See the signal multiplexing details for which exact GPIO signals are available.

Table 9-21. RGPIO Signal Descriptions

Chip signal name	Module signal name	Description	I/O
RGPIO[15:0]	RGPIO[15:0]	Data Input/Output. When configured as an input, the state of this signal is reflected in the read data register. When configured as an output, this signal is the output of the write data register.	I/O

Table 9-22. IRQ0 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
IRQ0	IRQ	External interrupt pin	I

Table 9-23. IRQ1 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
IRQ1	IRQ	External interrupt pin	I

Table 9-24. IRQ2 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
IRQ2	IRQ	External interrupt pin	I

Table 9-25. IRQ3 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
IRQ3	IRQ	External interrupt pin	I

Chapter 10
Port Mux Control

10.1 Pin Mux Control

Package pins can be programmed for up to sixteen different functions using the mux control registers. Controls are organized by GPIO port. Each GPIO port has four mux control registers, consisting of 4 bits per package pin. Alternate values for each function match the column number in which that function occurs in the pin summary table. That is, default functions are assigned value 00h, ALT1 functions 01h, and so on. Setting the RESERVED setting in the port muxing results in the DEFAULT function for the pin.

CAUTION

For any GPIO pin that is not available in a package, do not change the port mux control field for that pin. If you change the port mux control field for such a pin, the pin is enabled and can generate unnecessary current leakage.

10.2 Memory Map and Registers

MXC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_8060	Port A Pin Function 0 Register (MXC_PTAPF0)	8	R/W	00h	10.2.1/150
FFFF_8061	Port A Pin Function 1 Register (MXC_PTAPF1)	8	R/W	00h	10.2.2/151
FFFF_8062	Port A Pin Function 2 Register (MXC_PTAPF2)	8	R/W	00h	10.2.3/151
FFFF_8063	Port A Pin Function 3 Register (MXC_PTAPF3)	8	R/W	00h	10.2.4/152
FFFF_8064	Port B Pin Function 0 Register (MXC_PTBPF0)	8	R/W	00h	10.2.5/153
FFFF_8065	Port B Pin Function 1 Register (MXC_PTBPF1)	8	R/W	00h	10.2.6/153
FFFF_8066	Port B Pin Function 2 Register (MXC_PTBPF2)	8	R/W	00h	10.2.7/154
FFFF_8067	Port B Pin Function 3 Register (MXC_PTBPF3)	8	R/W	00h	10.2.8/155

Table continues on the next page...

MXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_8068	Port C Pin Function 0 Register (MXC_PTCPF0)	8	R/W	00h	10.2.9/156
FFFF_8069	Port C Pin Function 1 Register (MXC_PTCPF1)	8	R/W	00h	10.2.10/156
FFFF_806A	Port C Pin Function 2 Register (MXC_PTCPF2)	8	R/W	00h	10.2.11/157
FFFF_806B	Port C Pin Function 3 Register (MXC_PTCPF3)	8	R/W	00h	10.2.12/158
FFFF_806D	Port D Pin Function 1 Register (MXC_PTDPF1)	8	R/W	02h	10.2.13/158
FFFF_806E	Port D Pin Function 2 Register (MXC_PTDPF2)	8	R/W	00h	10.2.14/159
FFFF_806F	Port D Pin Function 3 Register (MXC_PTDPF3)	8	R/W	00h	10.2.15/160

10.2.1 Port A Pin Function 0 Register (MXC_PTAPF0)

Address: FFFF_8060h base + 0h offset = FFFF_8060h

Bit	7	6	5	4	3	2	1	0
Read Write	A7				A6			
Reset	0	0	0	0	0	0	0	0

MXC_PTAPF0 field descriptions

Field	Description
7–4 A7	Port A7 Pin Mux Controls 0000 Disabled 0001 PTA7 0010 LPT_ALT0 0011 Reserved 0100 Reserved 0101 RGPIO7 0110 Reserved 0111 Reserved 1000-1111 Reserved
3–0 A6	Port A6 Pin Mux Controls 0000 Disabled 0001 PTA6 0010 Reserved 0011 Reserved 0100 Reserved 0101 RGPIO6 0110 Reserved 0111 Reserved 1000-1111 Reserved

10.2.2 Port A Pin Function 1 Register (MXC_PTAPF1)

Address: FFFF_8060h base + 1h offset = FFFF_8061h

Bit	7	6	5	4	3	2	1	0
Read	A5				0			
Write								
Reset	0	0	0	0	0	0	0	0

MXC_PTAPF1 field descriptions

Field	Description
7–4 A5	Port A5 Pin Mux Controls 0000 Disabled 0001 PTA5 0010 SCI2_CTS_b 0011 FTM1_CH4 0100 Reserved 0101 RGPIO5 0110 Reserved 0111 Reserved 1000-1111 Reserved
3–0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

10.2.3 Port A Pin Function 2 Register (MXC_PTAPF2)

Address: FFFF_8060h base + 2h offset = FFFF_8062h

Bit	7	6	5	4	3	2	1	0
Read	A3				A2			
Write								
Reset	0	0	0	0	0	0	0	0

MXC_PTAPF2 field descriptions

Field	Description
7–4 A3	Port A3 Pin Mux Controls 0000 Disabled 0001 PTA3 0010 FTM1_CH2 0011 CLKOUT 0100 Reserved 0101 RGPIO3 0110 Reserved

Table continues on the next page...

MXC_PTAPF2 field descriptions (continued)

Field	Description
	0111 Reserved 1000-1111 Reserved
3–0 A2	Port A2 Pin Mux Controls 0000 Disabled 0001 PTA2 0010 FTM1_CH1 0011 Reserved 0100 IIC1_SDA 0101 RGPIO2 0110 Reserved 0111 Reserved 1000-1111 Reserved

10.2.4 Port A Pin Function 3 Register (MXC_PTAPF3)

Address: FFFF_8060h base + 3h offset = FFFF_8063h

Bit	7	6	5	4	3	2	1	0
Read	A1				0			
Write								
Reset	0	0	0	0	0	0	0	0

MXC_PTAPF3 field descriptions

Field	Description
7–4 A1	Port A1 Pin Mux Controls 0000 Disabled 0001 PTA1 0010 FTM1_CH0 0011 Reserved 0100 IIC1_SCL 0101 RGPIO1 0110 Reserved 0111 Reserved 1000-1111 Reserved
3–0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

10.2.5 Port B Pin Function 0 Register (MXC_PTBPF0)

Address: FFFF_8060h base + 4h offset = FFFF_8064h

Bit	7	6	5	4	3	2	1	0
Read Write	B7				B6			
Reset	0	0	0	0	0	0	0	0

MXC_PTBPF0 field descriptions

Field	Description
7–4 B7	Port B7 Pin Mux Controls 0000 Disabled 0001 PTB7 0010 SCI1_TX 0011 SPI1_MOSI 0100 Reserved 0101 RGPIO15 0110 Reserved 0111 Reserved 1000-1111 Reserved
3–0 B6	Port B6 Pin Mux Controls 0000 Disabled 0001 PTB6 0010 Reserved 0011 SPI2_CLK 0100 Reserved 0101 RGPIO14 0110 Reserved 0111 Reserved 1000-1111 Reserved

10.2.6 Port B Pin Function 1 Register (MXC_PTBPF1)

Address: FFFF_8060h base + 5h offset = FFFF_8065h

Bit	7	6	5	4	3	2	1	0
Read Write	B5				B4			
Reset	0	0	0	0	0	0	0	0

MXC_PTBPF1 field descriptions

Field	Description
7–4 B5	Port B5 Pin Mux Controls 0000 Disabled

Table continues on the next page...

MXC_PTBPF1 field descriptions (continued)

Field	Description
	0001 PTB5 0010 SCI2_RX 0011 SPI2_MISO 0100 Reserved 0101 RGPIO13 0110 Reserved 0111 Reserved 1000-1111 Reserved
3–0 B4	Port B4 Pin Mux Controls 0000 Disabled 0001 PTB4 0010 SCI2_TX 0011 SPI2_MOSI 0100 Reserved 0101 RGPIO12 0110 Reserved 0111 Reserved 1000-1111 Reserved

10.2.7 Port B Pin Function 2 Register (MXC_PTBPF2)

Address: FFFF_8060h base + 6h offset = FFFF_8066h

Bit	7	6	5	4	3	2	1	0
Read Write	B3				B2			
Reset	0	0	0	0	0	0	0	0

MXC_PTBPF2 field descriptions

Field	Description
7–4 B3	Port B3 Pin Mux Controls 0000 Disabled 0001 PTB3 0010 SCI0_CTS_b 0011 FTM0_CH0 0100 IIC0_SDA 0101 RGPIO11 0110 Reserved 0111 Reserved 1000-1111 Reserved
3–0 B2	Port B2 Pin Mux Controls 0000 Disabled

Table continues on the next page...

MXC_PTBPF2 field descriptions (continued)

Field	Description
	0001 PTB2 0010 SCI0_RTS_b 0011 FTM0_CH1 0100 IIC0_SCL 0101 RGPIO10 0110 Reserved 0111 Reserved 1000-1111 Reserved

10.2.8 Port B Pin Function 3 Register (MXC_PTBPF3)

Address: FFFF_8060h base + 7h offset = FFFF_8067h

Bit	7	6	5	4	3	2	1	0
Read Write	B1				B0			
Reset	0	0	0	0	0	0	0	0

MXC_PTBPF3 field descriptions

Field	Description
7–4 B1	Port B1 Pin Mux Controls 0000 IRQ0 0001 PTB1 0010 Reserved 0011 Reserved 0100 Reserved 0101 RGPIO9 0110 Reserved 0111 Reserved 1000-1111 Reserved
3–0 B0	Port B0 Pin Mux Controls 0000 Disabled 0001 PTB0 0010 Reserved 0011 Reserved 0100 Reserved 0101 RGPIO8 0110 Reserved 0111 Reserved 1000-1111 Reserved

10.2.9 Port C Pin Function 0 Register (MXC_PTCPF0)

Address: FFFF_8060h base + 8h offset = FFFF_8068h

Bit	7	6	5	4	3	2	1	0
Read	C7				0			
Write								
Reset	0	0	0	0	0	0	0	0

MXC_PTCPF0 field descriptions

Field	Description
7–4 C7	Port C7 Pin Mux Controls 0000 Disabled 0001 PTC7 0010 Reserved 0011 SPI1_SS 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000-1111 Reserved
3–0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

10.2.10 Port C Pin Function 1 Register (MXC_PTCPF1)

Address: FFFF_8060h base + 9h offset = FFFF_8069h

Bit	7	6	5	4	3	2	1	0
Read	C5				C4			
Write								
Reset	0	0	0	0	0	0	0	0

MXC_PTCPF1 field descriptions

Field	Description
7–4 C5	Port C5 Pin Mux Controls 0000 IRQ2 0001 PTC5 0010 SCI1_CTS_b 0011 TMR_CLKIN0 0100 Reserved 0101 Reserved 0110 Reserved

Table continues on the next page...

MXC_PTCPF1 field descriptions (continued)

Field	Description
	0111 Reserved 1000-1111 Reserved
3–0 C4	Port C4 Pin Mux Controls 0000 IRQ1 0001 PTC4 0010 SCI1_RTS_b 0011 Reserved 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000-1111 Reserved

10.2.11 Port C Pin Function 2 Register (MXC_PTCPF2)

Address: FFFF_8060h base + Ah offset = FFFF_806Ah

Bit	7	6	5	4	3	2	1	0
Read Write	C3				C2			
Reset	0	0	0	0	0	0	0	0

MXC_PTCPF2 field descriptions

Field	Description
7–4 C3	Port C3 Pin Mux Controls 0000 Disabled 0001 PTC3 0010 Reserved 0011 SPI1_CLK 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000-1111 Reserved
3–0 C2	Port C2 Pin Mux Controls 0000 Disabled 0001 PTC2 0010 SCI1_RX 0011 SPI1_MISO 0100 Reserved 0101 Reserved 0110 Reserved

Table continues on the next page...

MXC_PTCPF2 field descriptions (continued)

Field	Description
	0111 Reserved 1000-1111 Reserved

10.2.12 Port C Pin Function 3 Register (MXC_PTCPF3)

Address: FFFF_8060h base + Bh offset = FFFF_806Bh

Bit	7	6	5	4	3	2	1	0
Read Write	C1				C0			
Reset	0	0	0	0	0	0	0	0

MXC_PTCPF3 field descriptions

Field	Description
7–4 C1	Port C1 Pin Mux Controls 0000 IRQ3 0001 PTC1 0010 FTM1_CH5 0011 CLKOUT 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000-1111 Reserved
3–0 C0	Port C0 Pin Mux Controls 0000 Disabled 0001 PTC0 0010 Reserved 0011 SPI2_SS 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000-1111 Reserved

10.2.13 Port D Pin Function 1 Register (MXC_PTDPF1)

Address: FFFF_8060h base + Dh offset = FFFF_806Dh

Bit	7	6	5	4	3	2	1	0
Read	0				D4			
Write								
Reset	0	0	0	0	0	0	1	0

MXC_PTDPF1 field descriptions

Field	Description
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–0 D4	Port D4 Pin Mux Controls 0000 Disabled 0001 PTD4 0010 BKGD 0011 MS 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000-1111 Reserved

10.2.14 Port D Pin Function 2 Register (MXC_PTDPF2)

Address: FFFF_8060h base + Eh offset = FFFF_806Eh

Bit	7	6	5	4	3	2	1	0
Read Write	D3				D2			
Reset	0	0	0	0	0	0	0	0

MXC_PTDPF2 field descriptions

Field	Description
7–4 D3	Port D3 Pin Mux Controls 0000 Disabled 0001 PTD3 0010 Reserved 0011 SPI0_CLK 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000-1111 Reserved
3–0 D2	Port D2 Pin Mux Controls 0000 Disabled 0001 PTD2 0010 SCI0_RX 0011 SPI0_MISO 0100 Reserved 0101 Reserved 0110 Reserved

Table continues on the next page...

MXC_PTDPF2 field descriptions (continued)

Field	Description
	0111 Reserved 1000-1111 Reserved

10.2.15 Port D Pin Function 3 Register (MXC_PTDPF3)

Address: FFFF_8060h base + Fh offset = FFFF_806Fh

Bit	7	6	5	4	3	2	1	0
Read Write	D1				D0			
Reset	0	0	0	0	0	0	0	0

MXC_PTDPF3 field descriptions

Field	Description
7–4 D1	Port D1 Pin Mux Controls 0000 Disabled 0001 PTD1 0010 SCI0_TX 0011 SPI0_MOSI 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000-1111 Reserved
3–0 D0	Port D0 Pin Mux Controls 0000 Disabled 0001 PTD0 0010 SPI0_SS 0011 Reserved 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000-1111 Reserved

Chapter 11
Core

11.1 Introduction

This section describes the organization of the Version 1 (V1) ColdFire processor core and an overview of the program-visible registers. For detailed information on instructions, see the ISA_ C definition in the *ColdFire Family Programmer's Reference Manual*.

11.1.1 Overview

As with all ColdFire cores, the V1 ColdFire core consists of two separate pipelines decoupled by an instruction buffer.

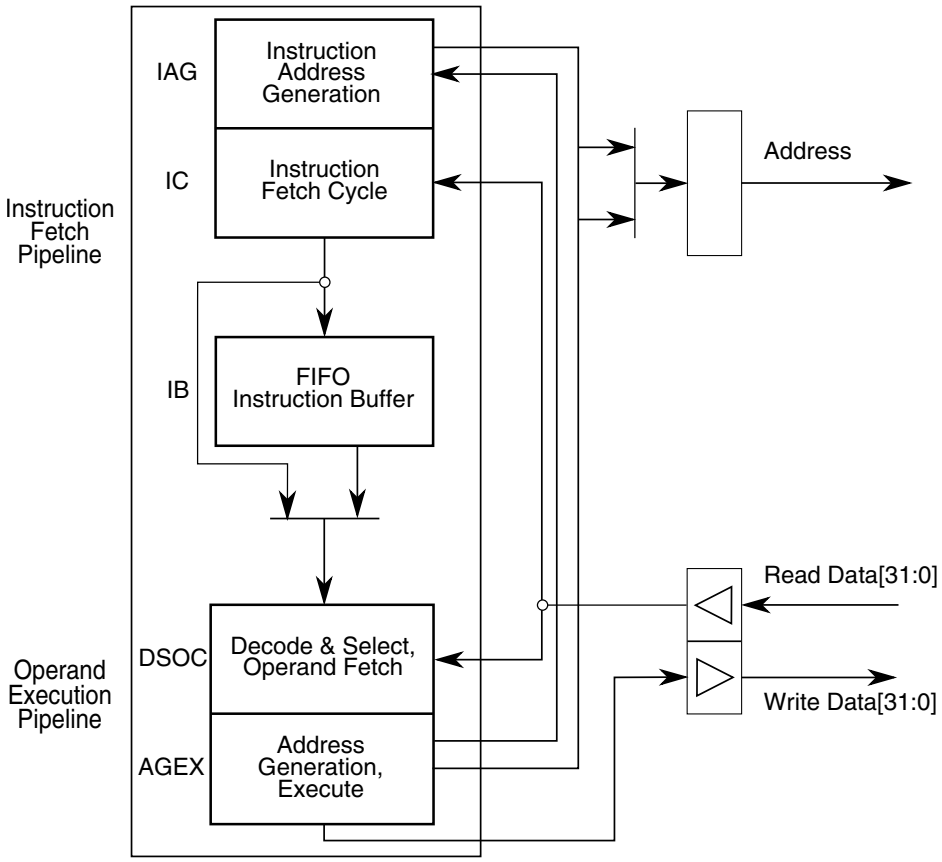


Figure 11-1. V1 ColdFire Core Pipelines

The instruction fetch pipeline (IFP) is a two-stage pipeline for prefetching instructions. The prefetched instruction stream is then gated into the two-stage operand execution pipeline (OEP), that decodes the instruction, fetches the required operands, and then executes the required function. Because the IFP and OEP pipelines are decoupled by an instruction buffer serving as a FIFO queue, the IFP is able to prefetch instructions in advance of their actual use by the OEP thereby minimizing time stalled waiting for instructions.

The V1 ColdFire core pipeline stages include the following:

- Two-stage instruction fetch pipeline (IFP) (plus optional instruction buffer stage)
  - Instruction address generation (IAG) — Calculates the next prefetch address
  - Instruction fetch cycle (IC) — Initiates prefetch on the processor's local bus
  - Instruction buffer (IB) — Optional buffer stage minimizes fetch latency effects using FIFO queue
- Two-stage operand execution pipeline (OEP)

- Decode and select/operand fetch cycle (DSOC) — Decodes instructions and fetches the required components for effective address calculation, or the operand fetch cycle
- Address generation/execute cycle (AGEX) — Calculates operand address or executes the instruction

When the instruction buffer is empty, opcodes are loaded directly from the IC cycle into the operand execution pipeline. If the buffer is not empty, the IFP stores the contents of the fetched instruction in the IB until it is required by the OEP. The instruction buffer on the V1 core contains three longwords of storage.

For register-to-register and register-to-memory store operations, the instruction passes through both OEP stages once. For memory-to-register and read-modify-write memory operations, an instruction is effectively staged through the OEP twice; the first time to calculate the effective address and initiate the operand fetch on the processor's local bus, and the second time to complete the operand reference and perform the required function defined by the instruction.

The resulting pipeline and local bus structure allow the V1 ColdFire core to deliver sustained high performance across a variety of demanding embedded applications.

11.2 Memory Map/Register Description

The following sections describe the processor registers in the user and supervisor programming models. The programming model is selected based on the processor privilege level (user mode or supervisor mode) as defined by the S bit of the status register (SR).

The user-programming model consists of the following registers:

- 16 general-purpose 32-bit registers (D0–D7, A0–A7)
- 32-bit program counter (PC)
- 8-bit condition code register (CCR)
- EMAC registers (refer to EMAC description)
  - Four 48-bit accumulator registers partitioned as follows:

- Four 32-bit accumulators (ACC0–ACC3)
- Eight 8-bit accumulator extension bytes (two per accumulator). These are grouped into two 32-bit values for load and store operations (ACCEXT01 and ACCEXT23).

Accumulators and extension bytes can be loaded, copied, and stored, and results from EMAC arithmetic operations generally affect the entire 48-bit destination.

- One 16-bit mask register (MASK)
- One 32-bit status register (MACSR) including four indicator bits signaling product or accumulation overflow (one for each accumulator: PAV0–PAV3)

The supervisor programming model is to be used only by system control software to implement restricted operating system functions, I/O control, and memory management. All accesses that affect the control features of ColdFire processors are in the supervisor programming model, that consists of registers available in user mode as well as the following control registers:

- 16-bit status register (SR)
- 32-bit supervisor stack pointer (SSP)
- 32-bit vector base register (VBR)
- 32-bit CPU configuration register (CPUCR)

Table 11-1. ColdFire core programming model

BDM command[1]	Register[2]	Width (bits)	Access	Reset value	Written with MOVEC[3]
Supervisor/user access registers					
Load: 0x60 Store: 0x40	Data register 0 (D0) see Data registers (D0–D7)	32	R/W	See Reset Exception	No
Load: 0x61 Store: 0x41	Data register 1 (D1) see Data registers (D0–D7)	32	R/W	See Reset Exception	No
Load: 0x62-7 Store: 0x42-7	Data registers 2-7 (D2-7) see Data registers (D0–D7)	32	R/W	POR: Undefined Else: Unaffected	No
Load: 0x68-E Store: 0x48-E	Address registers (A0–A6)	32	R/W	POR: Undefined Else: Unaffected	No
Load: 0x6F Store: 0x4F	User stack pointer see Supervisor/user stack pointers (A7 and OTHER_A7)	32	R/W	POR: Undefined Else: Unaffected	No

Table continues on the next page...

Table 11-1. ColdFire core programming model (continued)

BDM command[1]	Register[2]	Width (bits)	Access	Reset value	Written with MOVEC[3]
Load: 0xE4 Store: 0xC4	MAC status register (MACSR)	32	R/W	0x0000_0000	No
Load: 0xE5 Store: 0xC5	MAC address mask register (MASK)	16	R/W	0xFFFF	No
Load: 0xE6 Store: 0xC6	MAC accumulator (ACC)	32	R/W	POR: Undefined Else: Unaffected	No
Load: 0xE7 Store: 0xC7	MAC accumulator 0,1 extension bytes (ACCext01)	32	R/W	POR: Undefined Else: Unaffected	No
Load: 0xE8 Store: 0xC8	MAC accumulator 2,3 extension bytes (ACCext23)	32	R/W	POR: Undefined Else: Unaffected	No
Load: 0xEE Store: 0xCE	Condition code register (CCR) (LSB of Status register (SR))	8	R/W	POR: Undefined Else: Unaffected	No
Load: 0xEF Store: 0xCF	Program counter (PC)	32	R/W	Contents of location 0x(00)00_0004	No
Supervisor access only registers					
Load: 0xE0 Store: 0xC0	Supervisor stack pointer see Supervisor/user stack pointers (A7 and OTHER_A7)	32	R/W	Contents of location 0x(00)00_0000	No
Load: 0xE1 Store: 0xC1	Vector base register (VBR)	32	R/W	See register's description	Yes Rc = 0x801
Load: 0xE2 Store: 0xC2	CPU configuration register (CPUCR)	32	W	See register's description	Yes Rc = 0x802
Load: 0xEE Store: 0xCE	Status register (SR)	16	R/W	0x27--	No

1. The values listed in this column represent the 8-bit BDM command code used when accessing the core registers via the 1-pin BDM port. For details, see information about ColdFire debug operation. (These BDM commands are not similar to those of non-V1 ColdFire processors.)
2. The EMAC registers are available only if the EMAC module is present on the device.
3. If the given register is written using the MOVEC instruction, the 12-bit control register address (Rc) is also specified.

11.2.1 Data registers (D0–D7)

These registers are for bit (1-bit), byte (8-bit), word (16-bit) and longword (32-bit) operations; they can also be used as index registers.

NOTE

The D0 and D1 registers contain hardware configuration details after reset. See Reset Exception for more details.

Table 11-2. Data registers (D0–D7)

BDM:	Load: 0x60 + *n*; *n* = 0–7 (D*n*) Store: 0x40 + *n*; *n* = 0–7 (D*n*)												Access: User read/write BDM read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Data															
Reset (D2–D7)	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset (D0, D1)	See Reset Exception															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Data															
Reset (D2–D7)	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset (D0, D1)	See Reset Exception															

11.2.2 Address registers (A0–A6)

These registers can be used as software stack pointers, index registers, or base address registers. They can also be used for word and longword operations.

Table 11-3. Address registers (A0–A6)

BDM:	Load: 0x68 + *n*; *n* = 0–6 (A*n*) Store: 0x48 + *n*; *n* = 0–6 (A*n*)												Access: User read/write BDM read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Address															
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Address															
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

11.2.3 Supervisor/user stack pointers (A7 and OTHER_A7)

This ColdFire architecture supports two independent stack pointer (A7) registers: the supervisor stack pointer (SSP) and the user stack pointer (USP). The hardware implementation of these two program-visible 32-bit registers does not identify one as the SSP and the other as the USP. Instead, the hardware uses one 32-bit register as the active A7 and the other as OTHER_A7. Thus, the register contents are a function of the processor operation mode, as shown in the following:

```
if SR[S] = 1
          then          A7 = Supervisor Stack Pointer
                        OTHER_A7 = User Stack Pointer
          else          A7 = User Stack Pointer
                        OTHER_A7 = Supervisor Stack Pointer
```

The BDM programming model supports direct reads and writes to A7 and OTHER_A7. It is the responsibility of the external development system to determine, based on the setting of SR[S], the mapping of A7 and OTHER_A7 to the two program-visible definitions (SSP and USP).

To support dual stack pointers, the following two supervisor instructions are included in the ColdFire instruction set architecture to load/store the USP:

```
move.l Ay,USP;move to USP
move.l USP,Ax;move from USP
```

The *ColdFire Family Programmer's Reference Manual* describes these instructions. All other instruction references to the stack pointer, explicit or implicit, access the active A7 register.

NOTE

The USP must be initialized using the

```
move.l Ay,USP
```

instruction before any entry into user mode.

The SSP is loaded during reset exception processing with the contents of location 0x(00)00_0000.

Table 11-4. Stack pointer registers (A7 and OTHER_A7)

BDM:	Load: 0x6F (A7) Store: 0x4F (A7) Load: 0xE0 (OTHER_A7) Store: 0xC0 (OTHER_A7)	Access: A7: User or BDM read/write OTHER_A7: Supervisor or BDM read/write

Table continues on the next page...

Table 11-4. Stack pointer registers (A7 and OTHER_A7) (continued)

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

11.2.4 Condition code register (CCR)

The CCR is the LSB of the processor status register (SR). Bits 4–0 act as indicator flags for results generated by processor operations. The extend bit (X) is also an input operand during multiprecision arithmetic computations. The CCR register must be explicitly loaded after reset and before any CMP, Bcc, or Scc instructions are executed.

Table 11-5. Condition code register (CCR)

BDM:	LSB of status register (SR) Load: 0xEE (SR) Store: 0xCE (SR)						Access: User read/write BDM read/write	
	7	6	5	4	3	2	1	0
R	0	0	0	X	N	Z	V	C
W								
Reset	0	0	0	–	–	–	–	–

Table 11-6. CCR field descriptions

Field	Description
7–5	Reserved; must be cleared.
4 X	Extend condition code This bit is set to the C bit's value for arithmetic operations; otherwise not affected or set to a specified result.
3 N	Negative condition code This bit is set if the most significant bit of the result is set; otherwise cleared.
2 Z	Zero condition code This bit is set if result equals zero; otherwise cleared.

Table continues on the next page...

Table 11-6. CCR field descriptions (continued)

Field	Description
1 V	Overflow condition code This bit is set if an arithmetic overflow occurs implying the result cannot be represented in operand size; otherwise cleared.
0 C	Carry condition code This bit is set if a carry out of the operand msb occurs for an addition or if a borrow occurs in a subtraction; otherwise cleared.

11.2.5 Program counter (PC)

The PC contains the currently executing instruction address. During instruction execution and exception processing, the processor automatically increments PC contents or places a new value in the PC. The PC is a base address for PC-relative operand addressing.

The PC is initially loaded during reset exception processing with the contents at location 0x(00)00_0004.

Table 11-7. Program counter (PC) register

BDM:	Load: 0xEF (PC) Store: 0xCF (PC)												Access: User read/write BDM read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	Address							
W																
Reset	0	0	0	0	0	0	0	0	–	–	–	–	–	–	–	–
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Address															
W																
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

11.2.6 Vector base register (VBR)

The VBR contains the base address of the exception vector table in the memory. To access the vector table, the displacement of an exception vector is added to the value in VBR. The lower 20 bits of the VBR are not implemented by ColdFire processors. They are assumed to be zero, forcing the table to be aligned on a 1 MB boundary.

In addition, because the V1 ColdFire core supports a 16 MB address space, the upper byte of the VBR is also forced to zero. The VBR can be used to relocate the exception vector table from its default position in the flash memory (address 0x(00)00_0000) to the base of the RAM (address 0x(00)80_0000) if needed.

Table 11-8. Vector base register (VBR)

BDM:	0x801 (VBR) Load: 0xE1 (VBR) Store: 0xC1 (VBR)												Access: Supervisor read/write BDM read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	Base Address				–	–	–	–
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	–	–	–	–
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
W																
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

11.2.7 CPU configuration register (CPUCR)

This register provides supervisor mode configurability of specific core functionality. Particular hardware features can be enabled/disabled individually based on the state of the CPUCR.

NOTE

Program the Flash Memory Controller's configuration and control settings only while the Flash Memory Controller is idle. Changing settings while a flash access is in progress can lead to non-deterministic behavior.

NOTE

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

Table 11-9. CPU configuration register (CPUCR)

BDM:	0x802 (CPUCR) Load: 0xE2 (CPUCR) Store: 0xC2 (CPUCR)												Access: Supervisor read/ write BDM read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ARD	IRD	IAE	IME	BWD	HAE	FSD	CBR R	FHP	FCDI S	FDC EN	FICDI S	0	–	–	–
W													FCCL R			
Reset	0	0	0	0	0	0	0	0	–	0	0	0	0	–	–	–
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
W																
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 11-10. CPUCR field descriptions

Field	Description	
31 ARD	Address-related reset disable Used to disable the generation of a reset event in response to a processor exception caused by an address error, a bus error, an RTE format error, or a fault-on-fault halt condition.	
	0	The detection of these types of exception conditions or the fault-on-fault halt condition generate a reset event.
	1	No reset is generated in response to these exception conditions.
30 IRD	Instruction-related reset disable Used to disable the generation of a reset event in response to a processor exception caused by the attempted execution of an illegal instruction (except for the ILLEGAL opcode), illegal line A, illegal line F instructions, or a privilege violation.	
	0	The detection of these types of exception conditions generate a reset event.
	1	No reset is generated in response to these exception conditions.
29 IAE	Interrupt acknowledge (IACK) enable Forces the processor to generate an IACK read cycle from the interrupt controller during exception processing to retrieve the vector number of the interrupt request being acknowledged. The processor's execution time for an interrupt exception is slightly improved when this bit is cleared.	
	0	The processor uses the vector number provided by the interrupt controller at the time the request is signaled.
	1	IACK read cycle from the interrupt controller is generated.
28 IME	Interrupt mask enable Forces the processor to raise the interrupt level mask (SR[I]) to 7 during every interrupt exception.	
	0	As part of an interrupt exception, the processor sets SR[I] to the level of the interrupt being serviced.
	1	As part of an interrupt exception, the processor sets SR[I] to 7. This disables all level 1-6 interrupt requests but allows recognition of the edge-sensitive level 7 requests.

Table continues on the next page...

Table 11-10. CPUCR field descriptions (continued)

Field	Description
27 BWD	Buffered write disable The ColdFire core is capable of marking processor memory writes as bufferable or non-bufferable. **NOTE:** If buffered writes are enabled (BWD is 0), any error status is lost as the immediate termination of the data transfer assumes an error-free completion. 0 — Writes are buffered and the bus cycle is terminated immediately with zero wait states. 1 — Disable the buffering of writes. In this configuration, the write transfer is terminated based on the response time of the addressed destination memory device.
26 HAE	Crossbar high priority arbitration enable Elevates the processor's fixed crossbar arbitration from lowest to highest during the processing of an interrupt service routine. 0 — Do not enable the processor's fixed crossbar arbitration from lowest to highest during the processing of an interrupt service routine. 1 — Enable the processor's fixed crossbar arbitration from lowest to highest during the processing of an interrupt service routine.
25 FSD	Flash speculation disable This bit controls whether prefetching by the speculation buffer is enabled. When enabled, prefetching occurs only for program flash accesses. Disabling prefetching also clears the current prefetch buffer. 0 — Prefetching is enabled. 1 — Prefetching is disabled.
24 CBRR	Crossbar round-robin arbitration enable Configures the crossbar slave ports to fixed-priority or round-robin arbitration. 0 — Fixed-priority arbitration 1 — Round-robin arbitration
23 FHP	Crossbar force high priority arbitration. Elevates the processor's fixed crossbar arbitration from lowest to highest. 0 — Do not force the elevation of the processor's fixed crossbar arbitration from lowest to highest. 1 — Force the elevation of the processor's fixed crossbar arbitration from lowest to highest.
22 FCDIS	Flash controller cache disable Disables the caching of the flash read data. This bit overrides the instruction and data cache enables. 0 — The flash controller's cache is enabled. Use the instruction and data cache enable bits to decide which accesses are cached. 1 — The flash controller's cache is disabled. (lower performance)
21 FDCEN	Flash data caching enable Enables the caching of operand fetches from the flash memory controller. 0 — Data accesses via the flash controller cache are disabled. 1 — Data accesses via the flash controller cache are enabled.

Table continues on the next page...

Table 11-10. CPUCR field descriptions (continued)

Field	Description
20 FICDIS	Flash instruction caching disable Disables the caching of instruction fetches from the flash memory controller. 0 Instruction fetches via the flash controller cache are enabled. 1 Instruction fetches via the flash controller cache are disabled.
19 FCCLR	Clear flash controller cache Setting this bit to 1 clears (invalidates) the cache immediately. This bit always reads as 0. The cache invalidate function does not depend on the flash cache being enabled. The cache is invalidated by system reset. System software is required to maintain memory coherency when any segment of the flash memory is programmed or erased. Accordingly, cache invalidations must occur after a programming or erase event is completed and before the new memory image is accessed.
18–0	Reserved; must be cleared.

11.2.8 Status register (SR)

This register stores the processor status and includes the CCR, the interrupt priority mask, and other control bits. In supervisor mode, software can access the entire SR. In user mode, only the lower 8 bits (the CCR) are accessible. The control bits indicate the following states for the processor: trace mode (T bit), supervisor or user mode (S bit), and master or interrupt state (M bit). All defined bits in the SR have read/write access when in supervisor mode. The lower byte of the SR (the CCR) must be loaded explicitly after reset and before any compare (CMP), Bcc, or Scc instructions execute.

Table 11-11. Status register (SR)

BDM:	Load: 0xEE (SR) Store: 0xCE (SR)												Access: Supervisor read/write BDM read/write			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	T	0	S	M	0	I			0	0	0	X	N	Z	V	C
W																
Reset	0	0	1	0	0	1	1	1	0	0	0	–	–	–	–	–

Table 11-12. SR field descriptions

Field	Description
15 T	Trace enable When this bit is set, the processor performs a trace exception after every instruction.
14	Reserved; must be cleared.

Table continues on the next page...

Table 11-12. SR field descriptions (continued)

Field	Description
13 S	Supervisor/user state 0 User mode 1 Supervisor mode
12 M	Master/interrupt state This bit is cleared by an interrupt exception, and software can set it during execution of the RTE or move to SR instructions.
11	Reserved; must be cleared.
10–8 I	Interrupt level mask Defines current interrupt level. Interrupt requests are inhibited for all priority levels less than or equal to current level, except edge-sensitive level 7 requests, which cannot be masked.
7–0	Refer to Condition code register (CCR)

11.3 Functional Description

11.3.1 Instruction Set Architecture

The original ColdFire instruction set architecture (ISA_A) was derived from the M68000 family opcodes based on extensive analysis of embedded application code. The ISA was optimized for code compiled from high-level languages where the dominant operand size was the 32-bit integer declaration. This approach minimized processor complexity and cost, while providing excellent performance for compiled applications.

After the initial ColdFire compilers were created, developers noted there were certain ISA additions that would enhance code density and overall performance. Additionally, as users implemented ColdFire-based designs into a wide range of embedded systems, they found certain frequently-used instruction sequences that could be improved by the creation of additional instructions.

The original ISA definition minimized support for instructions referencing byte- and word-sized operands. Full support for the move byte and move word instructions was provided, but the only other opcodes supporting these data types are CLR (clear) and TST (test). A set of instruction enhancements has been implemented in subsequent ISA revisions, ISA_B and ISA_C. The new opcodes primarily addressed three areas:

1. Enhanced support for byte and word-sized operands
2. Enhanced support for position-independent code
3. Miscellaneous instruction additions to address new functionality

The following table summarizes the instructions added to revision ISA_A to form revision ISA_ C. For more details see the *ColdFire Family Programmer's Reference Manual*.

Table 11-13. Instruction Enhancements over Revision ISA_A

Instruction	Description
BITREV	The contents of the destination data register are bit-reversed; that is, new Dn[31] equals old Dn[0], new Dn[30] equals old Dn[1], ..., new Dn[0] equals old Dn[31].
BYTEREV	The contents of the destination data register are byte-reversed; that is, new Dn[31:24] equals old Dn[7:0], ..., new Dn[7:0] equals old Dn[31:24].
FF1	The data register, Dn, is scanned, beginning from the most-significant bit (Dn[31]) and ending with the least-significant bit (Dn[0]), searching for the first set bit. The data register is then loaded with the offset count from bit 31 where the first set bit appears.
MOV3Q.L	Moves 3-bit immediate data to the destination location.
Move from USP	User Stack Pointer → Destination register
Move to USP	Source register → User Stack Pointer
MVS.{B,W}	Sign-extends source operand and moves it to destination register.
MVZ.{B,W}	Zero-fills source operand and moves it to destination register.
SATS.L	Performs saturation operation for signed arithmetic and updates destination register, depending on CCR[V] and bit 31 of the register.
TAS.B	Performs indivisible read-modify-write cycle to test and set addressed memory byte.
Bcc.L	Branch conditionally, longword
BSR.L	Branch to sub-routine, longword
CMP.{B,W}	Compare, byte and word
CMPA.W	Compare address, word
CMPI.{B,W}	Compare immediate, byte and word
MOVEI	Move immediate, byte and word to memory using Ax with displacement
STLDSR	Pushes the contents of the status register onto the stack and then reloads the status register with the immediate data value.

11.3.2 Exception Processing Overview

Exception processing for ColdFire processors is streamlined for performance. The ColdFire processors differ from the M68000 family because they include:

- A simplified exception vector table
- Reduced relocation capabilities using the vector-base register
- A single exception stack frame format
- Use of separate system stack pointers for user and supervisor modes.

All ColdFire processors use an instruction restart exception model.

Exception processing includes all actions from fault condition detection to the initiation of fetch for first handler instruction. Exception processing is comprised of four major steps:

1. The processor makes an internal copy of the SR and then enters supervisor mode by setting the S bit and disabling trace mode by clearing the T bit. The interrupt exception also forces the M bit to be cleared and the interrupt priority mask to set to current interrupt request level.
2. The processor determines the exception vector number. For all faults except interrupts, the processor performs this calculation based on exception type. For interrupts, the processor performs an interrupt-acknowledge (IACK) bus cycle to obtain the vector number from the interrupt controller if CPUCR[IAE] is set. The IACK cycle is mapped to special locations within the interrupt controller's address space with the interrupt level encoded in the address. If CPUCR[IAE] is cleared, the processor uses the vector number supplied by the interrupt controller at the time the request was signaled for improved performance.
3. The processor saves the current context by creating an exception stack frame on the system stack. The exception stack frame is created at a 0-modulo-4 address on top of the system stack pointed to by the supervisor stack pointer (SSP). As shown in Figure 11-2 , the processor uses a simplified fixed-length stack frame for all exceptions. The exception type determines whether the program counter placed in the exception stack frame defines the location of the faulting instruction (fault) or the address of the next instruction to be executed (next).
4. The processor calculates the address of the first instruction of the exception handler. By definition, the exception vector table is aligned on a 1 MB boundary. This instruction address is generated by fetching an exception vector from the table located at the address defined in the vector base register. The index into the exception table is calculated as (4 × vector number). After the exception vector has been fetched, the vector contents determine the address of the first instruction of the desired handler. After the instruction fetch for the first opcode of the handler has initiated, exception processing terminates and normal instruction processing continues in the handler.

All ColdFire processors support a 1024-byte vector table aligned on any 1 MB address boundary (see Table 11-14). For the V1 ColdFire core, the only practical locations for the vector table are based at 0x(00)00_0000 in the flash or 0x(00)80_0000 in the internal SRAM.

The table contains 256 exception vectors; the first 64 are defined for the core and the remaining 192 are device-specific peripheral interrupt vectors. See the interrupt chapter for details on the device-specific interrupt sources.

For the V1 ColdFire core, the table is partially populated with the first 64 reserved for internal processor exceptions, while vectors 64-102 are reserved for the peripheral I/O requests and the seven software interrupts. Vectors 103–255 are unused and reserved.

Table 11-14. Exception Vector Assignments

Vector Number(s)	Vector Offset (Hex)	Stacked Program Counter[1]	Assignment
0	0x000	—	Initial supervisor stack pointer
1	0x004	—	Initial program counter
2	0x008	Fault	Access error
3	0x00C	Fault	Address error
4	0x010	Fault	Illegal instruction
5	0x014	Fault	Divide by zero
6–7	0x018–0x01C	—	Reserved
8	0x020	Fault	Privilege violation
9	0x024	Next	Trace
10	0x028	Fault	Unimplemented line-A opcode
11	0x02C	Fault	Unimplemented line-F opcode
12	0x030	Next	Debug interrupt
13	0x034	—	Reserved
14	0x038	Fault	Format error
15–23	0x03C–0x05C	—	Reserved
24	0x060	Next	Spurious interrupt
25–31	0x064–0x07C	—	Reserved
32–47	0x080–0x0BC	Next	Trap # 0-15 instructions
48–60	0x0C0–0x0F0	—	Reserved
61	0x0F4	Fault	Unsupported instruction
62–63	0x0F8–0x0FC	—	Reserved
64–102	0x100–0x198	Next	Device-specific interrupts
103–255	0x19C–0x3FC	—	Reserved

1. Fault refers to the PC of the instruction that caused the exception.

   Next refers to the PC of the instruction that follows the instruction that caused the fault.

All ColdFire processors inhibit interrupt sampling during the first instruction of all exception handlers. This allows any handler to disable interrupts effectively, if necessary, by raising the interrupt mask level contained in the status register. In addition, the ISA_C architecture includes an instruction (STLDSR) that stores the current interrupt mask level and loads a value into the SR. This instruction is specifically intended for use as the first instruction of an interrupt service routine that services multiple interrupt requests with different interrupt levels. Finally, the V1 ColdFire core includes the CPUCR[IME] bit

that forces the processor to automatically raise the mask level to 7 during the interrupt exception, removing the need for any explicit instruction in the service routine to perform this function. For more details, see *ColdFire Family Programmer's Reference Manual*.

11.3.2.1 Exception Stack Frame Definition

The following figures shows the exception stack frame. The first longword contains the 16-bit format/vector word (F/V) and the 16-bit status register, and the second longword contains the 32-bit program counter address.

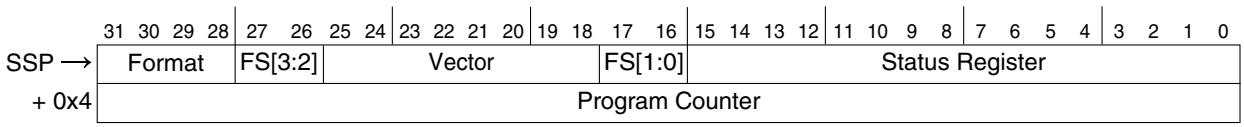


Figure 11-2. Exception Stack Frame Form

The 16-bit format/vector word contains three unique fields:

- A 4-bit format field at the top of the system stack is always written with a value of 4, 5, 6, or 7 by the processor, indicating a two-longword frame format. See Table 11-15 .

Table 11-15. Format Field Encodings

Original SSP @ Time of Exception, Bits 1:0	SSP @ 1st Instruction of Handler	Format Field
00	Original SSP - 8	0100
01	Original SSP - 9	0101
10	Original SSP - 10	0110
11	Original SSP - 11	0111

- There is a 4-bit fault status field, FS[3:0], at the top of the system stack. This field is defined for access and address errors only and written as zeros for all other exceptions. See Table 11-16 .

Table 11-16. Fault Status Encodings

FS[3:0]	Definition
00xx	Reserved
0100	Error on instruction fetch

Table continues on the next page...

Table 11-16. Fault Status Encodings (continued)

FS[3:0]	Definition
0101	Reserved
011x	Reserved
1000	Error on operand write
1001	Reserved
101x	Reserved
1100	Error on operand read
1101	Reserved
111x	Reserved

- The 8-bit vector number, vector[7:0], defines the exception type and is calculated by the processor for all internal faults and represents the value supplied by the interrupt controller in case of an interrupt . See Table 11-14 .

11.3.2.2 S08 and ColdFire Exception Processing Comparison

This section presents a brief summary comparing the exception processing differences between the S08 and V1 ColdFire processor families.

Table 11-17. Exception Processing Comparison

Attribute	S08	V1 ColdFire
Exception Vector Table	32, 2-byte entries, fixed location at upper end of memory	103, 4-byte entries, located at lower end of memory at reset, relocatable with the VBR
More on Vectors	2 for CPU + 30 for IRQs, reset at upper address	64 for CPU + 39 for IRQs, reset at lowest address
Exception Stack Frame	5-byte frame: CCR, A, X, PC	8-byte frame: F/V, SR, PC; General-purpose registers (An, Dn) must be saved/ restored by the ISR
Interrupt Levels	1 = *f*(CCR[I])	7 = *f*(SR[I]) with automatic hardware support for nesting
Non-Maskable IRQ Support	No	Yes, with level 7 interrupts
Core-enforced IRQ Sensitivity	No	Level 7 is edge sensitive, else level sensitive
INTC Vectoring	Fixed priorities and vector assignments	Fixed priorities and vector assignments, plus any 2 IRQs can be remapped as the highest priority level 6 requests
Software IACK	No	Yes
Exit Instruction from ISR	RTI	RTE

The notion of a software IACK refers to the ability to query the interrupt controller near the end of an interrupt service routine (after the current interrupt request has been cleared) to determine if there are any pending (but currently masked) interrupt requests. If the response to the software IACK's byte operand read is non-zero, the service routine uses the value as the vector number of the highest pending interrupt request and passes control to the appropriate new handler. This process avoids the overhead of a context restore and RTE instruction execution followed immediately by another interrupt exception and context save. In system environments with high rates of interrupt activity, this mechanism can improve overall performance noticeably.

Emulation of the S08's 1-level IRQ processing can easily be managed by software convention within the ColdFire interrupt service routines. For this type of operation, only two of the seven interrupt levels are used:

- SR[I] equals 0 indicates interrupts are enabled
- SR[I] equals 7 indicates interrupts are disabled

Recall that ColdFire treats true level 7 interrupts as edge-sensitive, non-maskable requests. Typically, only the IRQ input pin and a low-voltage detect are assigned as level 7 requests. All the remaining interrupt requests (levels 1-6) are masked when SR[I] equals 7. In any case, all ColdFire processors guarantee that the first instruction of any exception handler is executed before interrupt sampling resumes. By making the first instruction of the ISR a store/load status register (STLDSR #0x2700) or a move-to-SR (MOVE.W #2700,SR) instruction, interrupts can be safely disabled until the service routine is exited with an RTE instruction that lowers the SR[I] back to level 0. The same functionality can also be provided without an explicit instruction by setting CPUCR[IME] because this forces the processor to load SR[I] with 7 on each interrupt exception.

11.3.3 Processor Exceptions

11.3.3.1 Access Error Exception

The default operation of the V1 ColdFire processor is the generation of an illegal address reset event if an access error (also known as a bus error) is detected. If CPUCR[ARD] is set, the reset is disabled and a processor exception is generated as detailed below.

The exact processor response to an access error depends on the memory reference being performed. For an instruction fetch, the processor postpones the error reporting until the faulted reference is needed by an instruction for execution. Therefore, faults during

instruction prefetches followed by a change of instruction flow do not generate an exception. When the processor attempts to execute an instruction with a faulted opword and/or extension words, the access error is signaled and the instruction is aborted. For this type of exception, the programming model has not been altered by the instruction generating the access error.

If the access error occurs on an operand read, the processor immediately aborts the current instruction's execution and initiates exception processing. In this situation, any address register updates attributable to the auto-addressing modes, (for example, (An)+,-(An)), have already been performed, so the programming model contains the updated An value. In addition, if an access error occurs during a MOVEM instruction loading from memory, any registers already updated before the fault occurs contain the operands from memory.

The V1 ColdFire processor uses an imprecise reporting mechanism for access errors on operand writes. Because the actual write cycle may be decoupled from the processor's issuing of the operation, the signaling of an access error appears to be decoupled from the instruction that generated the write. Accordingly, the PC contained in the exception stack frame merely represents the location in the program when the access error was signaled. All programming model updates associated with the write instruction are completed. The NOP instruction can collect access errors for writes. This instruction delays its execution until all previous operations, including all pending write operations, are complete. If any previous write terminates with an access error, it is guaranteed to be reported on the NOP instruction.

11.3.3.2 Address Error Exception

The default operation of the V1 ColdFire processor is the generation of an illegal address reset event if an address error is detected. If CPUCR[ARD] equals 1, then the reset is disabled and a processor exception is generated as detailed below.

Any attempted execution transferring control to an odd instruction address (if bit 0 of the target address is set) results in an address error exception.

Any attempted use of a word-sized index register (Xn.w) or a scale factor of eight on an indexed effective addressing mode generates an address error, as does an attempted execution of a full-format indexed addressing mode, which is defined by bit 8 of extension word 1 being set.

If an address error occurs on an RTS instruction, the Version 1 ColdFire processor overwrites the faulting return PC with the address error stack frame.

11.3.3.3 Illegal Instruction Exception

The default operation of the V1 ColdFire processor is the generation of an illegal opcode reset event if an illegal instruction is detected. If CPUCR[IRD] is set, the reset is disabled and a processor exception is generated as detailed below. There is one special case involving the ILLEGAL opcode (0x4AFC); attempted execution of this instruction always generates an illegal instruction exception, regardless of the state of the CPUCR[IRD] bit.

The ColdFire variable-length instruction set architecture supports three instruction sizes: 16, 32, or 48 bits. The first instruction word is known as the operation word (or opword), while the optional words are known as extension word 1 and extension word 2. The opword is further subdivided into three sections: the upper four bits segment the entire ISA into 16 instruction lines, the next 6 bits define the operation mode (opmode), and the low-order 6 bits define the effective address. The opword line definition is shown below.

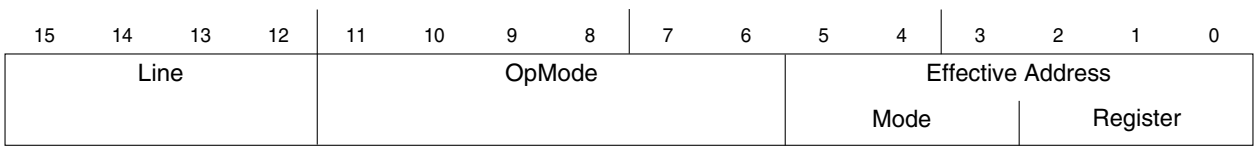


Figure 11-3. ColdFire Instruction Operation Word (Opword) Format

Table 11-18. ColdFire Opword Line Definition

Opword[Line]	Instruction Class
0x0	Bit manipulation, Arithmetic and Logical Immediate
0x1	Move Byte
0x2	Move Long
0x3	Move Word
0x4	Miscellaneous
0x5	Add (ADDQ) and Subtract Quick (SUBQ), Set according to Condition Codes (Scc)
0x6	PC-relative change-of-flow instructions Conditional (Bcc) and unconditional (BRA) branches, subroutine calls (BSR)
0x7	Move Quick (MOVEQ), Move with sign extension (MVS) and zero fill (MVZ)
0x8	Logical OR (OR)
0x9	Subtract (SUB), Subtract Extended (SUBX)
0xA	EMAC, Move 3-bit Quick (MOV3Q)
0xB	Compare (CMP), Exclusive-OR (EOR)
0xC	Logical AND (AND), Multiply Word (MUL)
0xD	Add (ADD), Add Extended (ADDX)
0xE	Arithmetic and logical shifts (ASL, ASR, LSL, LSR)
0xF	Write DDATA (WDDATA), Write Debug (WDEBUG)

In the original M68000 ISA definition, lines A and F were effectively reserved for user-defined operations (line A) and co-processor instructions (line F). Accordingly, there are two unique exception vectors associated with illegal opwords in these two lines.

Any attempted execution of an illegal 16-bit opcode (except for line-A and line-F opcodes) generates an illegal instruction exception (vector 4). Additionally, any attempted execution of any non-MAC line-A and most line-F opcodes generate their unique exception types, vector numbers 10 and 11, respectively. ColdFire cores do not provide illegal instruction detection on the extension words on any instruction, including MOVEC.

The V1 ColdFire processor also detects two special cases involving illegal instruction conditions:

1. If execution of the stop instruction is attempted and neither low-power stop nor wait modes are enabled, the processor signals an illegal instruction.
2. If execution of the halt instruction is attempted and BDM is not enabled (XCSR[ENBDM] equals 0), the processor signals an illegal instruction.

In both cases, the processor response is then dependent on the state of CPUCR[IRD]— a reset event or a processor exception.

11.3.3.4 Divide-By-Zero

Attempting to divide by zero causes an exception (vector 5, offset equal 0x014).

11.3.3.5 Privilege Violation

The default operation of the V1 ColdFire processor is the generation of an illegal opcode reset event if a privilege violation is detected. If CPUCR[IRD] is set, the reset is disabled and a processor exception is generated as detailed below.

The attempted execution of a supervisor mode instruction while in user mode generates a privilege violation exception. See *ColdFire Programmer's Reference Manual* for a list of supervisor-mode instructions.

There is one special case involving the HALT instruction. Normally, this opcode is a supervisor mode instruction, but if the debug module's CSR[UHE] is set, then this instruction can be also be executed in user mode for debugging purposes.

11.3.3.6 Trace Exception

To aid in program development, all ColdFire processors provide an instruction-by-instruction tracing capability. While in trace mode, indicated by setting of the SR[T] bit, the completion of an instruction execution (for all but the stop instruction) signals a trace exception. This functionality allows a debugger to monitor program execution.

The stop instruction has the following effects:

1. The instruction before the stop executes and then generates a trace exception. In the exception stack frame, the PC points to the stop opcode.
2. When the trace handler is exited, the stop instruction executes, loading the SR with the immediate operand from the instruction.
3. The processor then generates a trace exception. The PC in the exception stack frame points to the instruction after the stop, and the SR reflects the value loaded in the previous step.

If the processor is not in trace mode and executes a stop instruction where the immediate operand sets SR[T], hardware loads the SR and generates a trace exception. The PC in the exception stack frame points to the instruction after the stop, and the SR reflects the value loaded in step 2.

Because ColdFire processors do not support any hardware stacking of multiple exceptions, it is the responsibility of the operating system to check for trace mode after processing other exception types. As an example, consider a TRAP instruction execution while in trace mode. The processor initiates the trap exception and then passes control to the corresponding handler. If the system requires that a trace exception be processed, it is the responsibility of the trap exception handler to check for this condition (SR[T] in the exception stack frame set) and pass control to the trace handler before returning from the original exception.

11.3.3.7 Unimplemented Line-A Opcode

The default operation of the V1 ColdFire processor is the generation of an illegal opcode reset event if an unimplemented line-A opcode is detected. If CPUCR[IRD] is set, the reset is disabled and a processor exception is generated as detailed below.

A line-A opcode is defined when bits 15-12 of the opword are 0b1010. This exception is generated by the attempted execution of an undefined line-A opcode.

11.3.3.8 Unimplemented Line-F Opcode

The default operation of the V1 ColdFire processor is the generation of an illegal opcode reset event if an unimplemented line-F opcode is detected. If CPUCR[IRD] is set, the reset is disabled and a processor exception is generated as detailed below.

A line-F opcode is defined when bits 15-12 of the opword are 0b1111. This exception is generated when attempting to execute an undefined line-F opcode.

11.3.3.9 Debug Interrupt

See the debug chapter for a detailed explanation of this exception, which is generated in response to a hardware breakpoint register trigger. The processor does not generate an IACK cycle, but rather calculates the vector number internally (vector number 12). Additionally, SR[M,I] are unaffected by the interrupt.

11.3.3.10 RTE and Format Error Exception

The default operation of the V1 ColdFire processor is the generation of an illegal address reset event if an RTE format error is detected. If CPUCR[ARD] is set, the reset is disabled and a processor exception is generated as detailed below.

When an RTE instruction is executed, the processor first examines the 4-bit format field to validate the frame type. For a ColdFire core, any attempted RTE execution (where the format is not equal to {4,5,6,7}) generates a format error. The exception stack frame for the format error is created without disturbing the original RTE frame and the stacked PC pointing to the RTE instruction.

The selection of the format value provides some limited debug support for porting code from M68000 applications. On M68000 family processors, the SR was located at the top of the stack. On those processors, bit 30 of the longword addressed by the system stack pointer is typically zero. Thus, if an RTE is attempted using this old format, it generates a format error on a ColdFire processor.

If the format field defines a valid type, the processor: (1) reloads the SR operand, (2) fetches the second longword operand, (3) adjusts the stack pointer by adding the format value to the auto-incremented address after the fetch of the first longword, and then (4) transfers control to the instruction address defined by the second longword operand within the stack frame.

11.3.3.11 TRAP Instruction Exception

The TRAP #n instruction always forces an exception as part of its execution and is useful for implementing system calls. The TRAP instruction may be used to change from user to supervisor mode.

This set of 16 instructions provides a similar but expanded functionality compared to the S08's SWI (software interrupt) instruction. Do not confuse these instructions and their functionality with the software-scheduled interrupt requests, which are handled like normal I/O interrupt requests by the interrupt controller. The processing of the software-scheduled IRQs can be masked, based on the interrupt priority level defined by the SR[I] field.

11.3.3.12 Unsupported Instruction Exception

If execution of a valid instruction is attempted but the required hardware is not present in the processor (e.g., if the MAC is not present), an unsupported instruction exception is generated. The instruction functionality can then be emulated in the exception handler, if desired.

All ColdFire cores record the processor hardware configuration in the D0 register immediately after the negation of $\overline{\text{RESET}}$. See Reset Exception," for details.

11.3.3.13 Interrupt Exception

Interrupt exception processing includes interrupt recognition and the fetch of the appropriate vector from the interrupt controller using an IACK cycle or using the previously-supplied vector number, under control of CPUCR[IAE]. See the interrupt chapter for details on the interrupt controller.

11.3.3.14 Fault-on-Fault Halt

The default operation of the V1 ColdFire processor is the generation of an illegal address reset event if a fault-on-fault halt condition is detected. If CPUCR[ARD] is set, the reset is disabled and the processor is halted as detailed below.

If a ColdFire processor encounters any type of fault during the exception processing of another fault, the processor immediately halts execution with the catastrophic fault-on-fault condition. A reset is required to to exit this state.

11.3.3.15 Reset Exception

Resetting the processor causes a reset exception. The reset exception has the highest priority of any exception; it provides for system initialization and recovery from catastrophic failure. Reset also aborts any processing in progress when the reset input is recognized. Processing cannot be recovered.

The reset exception places the processor in the supervisor mode by setting the SR[S] bit and disables tracing by clearing the SR[T] bit. This exception also clears the SR[M] bit and sets the processor's SR[I] field to the highest level (level 7, 0b111). Next, the VBR is initialized to zero (0x0000_0000). The control registers specifying the operation of any memories (such as cache and/or RAM modules) connected directly to the processor are disabled.

Note

Other implementation-specific registers are also affected. Refer to each module in this reference manual for details on these registers.

After the processor is granted the bus, it performs two longword read-bus cycles. The first longword at address 0x(00)00_0000 is loaded into the supervisor stack pointer and the second longword at address 0x(00)00_0004 is loaded into the program counter. After the initial instruction is fetched from memory, program execution begins at the address in the PC. If an access error or address error occurs before the first instruction is executed, the processor enters the fault-on-fault state.

ColdFire processors load hardware configuration information into the D0 and D1 general-purpose registers after system reset. The hardware configuration information is loaded immediately after the reset-in signal is negated. This allows an emulator to read out the contents of these registers via the BDM to determine the hardware configuration.

Information loaded into D0 defines the processor hardware configuration as shown in Table 11-19.

Table 11-19. D0 Hardware Configuration Information

BDM:	Load: 0x60 (D0) Store: 0x40 (D0)												Access: User read-only BDM read-only			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PF								VER				REV			
W																
Reset	1	1	0	0	1	1	1	1	0	0	0	1	Device-specific			

Table continues on the next page...

Table 11-19. D0 Hardware Configuration Information (continued)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	DIV	EMAC	0	0	CAU	0	0	ISA				DEBUG			
W																
Reset	0	1	1	0	0	1	0	0	0	0	1	0	1	0	0	1

Table 11-20. D0 Hardware Configuration Information Field Descriptions

Field	Description	
31–24 PF	Processor family. This field is fixed to a hex value of 0xCF indicating a ColdFire core is present.	
23–20 VER	ColdFire core version number. Defines the hardware microarchitecture version of ColdFire core.	
	0001	V1 ColdFire core
19–16 REV	Processor revision number	
15	Reserved	
14 DIV	Divide present. This bit signals if the hardware divider (DIV) is present in the processor core.	
	0	Divide execute engine not present in core
	1	Divide execute engine is present in core
13 EMAC	EMAC present. This bit signals if the optional enhanced multiply-accumulate (EMAC) execution engine is present in processor core.	
	0	EMAC execute engine not present in core
	1	EMAC execute engine is present in core
12	Reserved	
11	Reserved	
10 CAU	Cryptographic acceleration unit present. This bit signals if the optional cryptographic acceleration unit (CAU) is present in the processor core.	
	0	CAU coprocessor engine not present in core
	1	CAU coprocessor engine is present in core
9–8	Reserved	
7–4 ISA	ISA revision. Defines the instruction-set architecture (ISA) revision level implemented in ColdFire processor core.	
	0010	ISA_C
3–0 DEBUG	Debug module revision number. Defines revision level of the debug module used in the ColdFire processor core.	
	1001	DEBUG_B+

Information loaded into D1 defines the local memory hardware configuration as shown in the following tables.

Table 11-21. D1 Hardware Configuration Information

BDM:	Load: 0x61 (D1) Store: 0x41 (D1)												Access: User read-only BDM read-only			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FLASHSZ				FLASHH		FLEXNVMSZ				FLEXNVMH		EEESIZE			
W																
Reset	Device-specific															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DEPART				Reserved						RAMSZ				RAMH	
W																
Reset	Device-specific															

Table 11-22. D1 Hardware Configuration Information Field Descriptions

Field	Description
31–28 FLASHSZ	Program flash memory size 0111: 32 KB 1000: 64 KB 1001: 128 KB 1010: 256 KB Other: Reserved
27–26 FLASHH	Program flash memory hole
25–22 FLEXNVMSZ	FlexNVM size 0110: 16 KB 0111: 32 KB Other: Reserved
21–20 FLEXNVMH	FlexNVM hole
19–16 EEESIZE	Total available FlexRAM size as defined by the EEESIZE field of data flash IFR. Refer to the detailed description of the flash memory module.
15–12 DEPART	FlexNVM partitioning between data flash and EEPROM as defined by the DEPART field of data flash IFR. Refer to the detailed description of the flash memory module.
11–6	Reserved
5–2 RAMSZ	RAM size 0101: 8 KB 0110: 16 KB 0111: 32 KB 1000: 64 KB Other: Reserved

Table continues on the next page...

Table 11-22. D1 Hardware Configuration Information Field Descriptions (continued)

Field	Description
1–0 RAMH	RAM hole

11.3.4 Instruction Execution Timing

This section presents processor instruction execution times in terms of processor-core clock cycles. The number of operand references for each instruction is enclosed in parentheses following the number of processor clock cycles. Each timing entry is presented as C(R/W) where:

- C is the number of processor clock cycles, including all applicable operand fetches and writes, and all internal core cycles required to complete the instruction execution.
- R/W is the number of operand reads (R) and writes (W) required by the instruction. An operation performing a read-modify-write function is denoted as (1/1).

This section includes the assumptions concerning the timing values and the execution time details.

11.3.4.1 Timing Assumptions

For the timing data presented in this section, these assumptions apply:

1. The OEP is loaded with the opword and all required extension words at the beginning of each instruction execution. This implies that the OEP does not wait for the IFP to supply opwords and/or extension words.
2. The OEP does not experience any sequence-related pipeline stalls. The most common example of stall involves consecutive store operations, excluding the MOVEM instruction. For all STORE operations (except MOVEM), certain hardware resources within the processor are marked as busy for two clock cycles after the final decode and select/operand fetch cycle (DSOC) of the store instruction. If a subsequent STORE instruction is encountered within this 2-cycle window, it is stalled until the resource again becomes available. Thus, the maximum pipeline stall involving consecutive STORE operations is two cycles. The MOVEM instruction uses a different set of resources and this stall does not apply.

3. The OEP completes all memory accesses without any stall conditions caused by the memory itself. Thus, the timing details provided in this section assume that an infinite zero-wait state memory is attached to the processor core.
4. All operand data accesses are aligned on the same byte boundary as the operand size; for example, 16-bit operands aligned on 0-modulo-2 addresses, 32-bit operands aligned on 0-modulo-4 addresses.

   The processor core decomposes misaligned operand references into a series of aligned accesses as shown in Table 11-23.

Table 11-23. Misaligned Operand References

address[1:0]	Size	Bus Operations	Additional C(R/W)
01 or 11	Word	Byte, Byte	2(1/0) if read 1(0/1) if write
01 or 11	Long	Byte, Word, Byte	3(2/0) if read 2(0/2) if write
10	Long	Word, Word	2(1/0) if read 1(0/1) if write

11.3.4.2 MOVE Instruction Execution Times

Table 11-25 lists execution times for MOVE.{B,W} instructions. Table 11-26 lists execution times for MOVE.L.

For all tables in this section, the execution time of any instruction using the PC-relative effective addressing modes is the same for the comparable An-relative mode. Refer to the following table for elaboration.

Table 11-24. Effective addressing modes with equal execution time

PC-relative effective addressing mode	An-relative effective addressing mode
ET with {<ea> = (d16,PC)}	ET with {<ea> = (d16,An)}
ET with {<ea> = (d8,PC,Xi*SF)}	ET with {<ea> = (d8,An,Xi*SF)}

The nomenclature xxx.wl refers to both forms of absolute addressing, xxx.w and xxx.l.

Table 11-25. MOVE Byte and Word Execution Times

Source	Destination						
	RX	(Ax)	(Ax)+	-(Ax)	(d16,Ax)	(d8,Ax,Xi*SF)	xxx.wl
Dy	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)
Ay	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)
(Ay)	2 (1/0)	3 (1/1)	3 (1/1)	3 (1/1)	3 (1/1)	4 (1/1))	3 (1/1)
(Ay)+	2 (1/0)	3 (1/1)	3 (1/1)	3 (1/1)	3 (1/1)	4 (1/1))	3 (1/1)
-(Ay)	2 (1/0)	3 (1/1)	3 (1/1)	3 (1/1)	3 (1/1)	4 (1/1))	3 (1/1)
(d16,Ay)	2 (1/0)	3 (1/1)	3 (1/1)	3 (1/1)	3 (1/1)	—	—
(d8,Ay,Xi*SF)	3 (1/0)	4 (1/1)	4 (1/1)	4 (1/1)	—	—	—
xxx.w	2 (1/0)	3 (1/1)	3 (1/1)	3 (1/1)	—	—	—
xxx.l	2 (1/0)	3 (1/1)	3 (1/1)	3 (1/1)	—	—	—
(d16,PC)	2 (1/0)	3 (1/1)	3 (1/1)	3 (1/1)	3 (1/1)	—	—
(d8,PC,Xi*SF)	3 (1/0)	4 (1/1)	4 (1/1)	4 (1/1))	—	—	—
#xxx	1(0/0)	3 (0/1)	3 (0/1)	3 (0/1)	1(0/1)	—	—

Table 11-26. MOVE Long Execution Times

Source	Destination						
	Rx	(Ax)	(Ax)+	-(Ax)	(d16,Ax)	(d8,Ax,Xi*SF)	xxx.wl
Dy	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)
Ay	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)
(Ay)	2 (1/0)	2 (1/1)	2 (1/1)	2 (1/1)	2 (1/1)	3 (1/1)	2 (1/1)
(Ay)+	2 (1/0)	2 (1/1)	2 (1/1)	2 (1/1)	2 (1/1)	3 (1/1)	2 (1/1)
-(Ay)	2 (1/0)	2 (1/1)	2 (1/1)	2 (1/1)	2 (1/1)	3 (1/1)	2 (1/1)
(d16,Ay)	2 (1/0)	2 (1/1)	2 (1/1)	2 (1/1)	2 (1/1)	—	—
(d8,Ay,Xi*SF)	3 (1/0)	3 (1/1)	3 (1/1)	3 (1/1)	—	—	—
xxx.w	2 (1/0)	2 (1/1)	2 (1/1)	2 (1/1)	—	—	—
xxx.l	2 (1/0)	2 (1/1)	2 (1/1)	2 (1/1)	—	—	—
(d16,PC)	2 (1/0)	2 (1/1)	2 (1/1)	2 (1/1)	2 (1/1)	—	—
(d8,PC,Xi*SF)	3 (1/0)	3 (1/1)	3 (1/1)	3 (1/1)	—	—	—
#xxx	1(0/0)	2 (0/1)	2 (0/1)	2 (0/1)	—	—	—

11.3.4.3 Standard One Operand Instruction Execution Times

Table 11-27. One Operand Instruction Execution Times

Opcode	<EA>	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xn*SF)	xxx.wl	#xxx
BITREV	Dx	1(0/0)	—	—	—	—	—	—	—

Table continues on the next page...

Table 11-27. One Operand Instruction Execution Times (continued)

Opcode	<EA>	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xn*SF)	xxx.wl	#xxx
BYTEREV	Dx	1(0/0)	—	—	—	—	—	—	—
CLR.B	<ea>	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
CLR.W	<ea>	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
CLR.L	<ea>	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
EXT.W	Dx	1(0/0)	—	—	—	—	—	—	—
EXT.L	Dx	1(0/0)	—	—	—	—	—	—	—
EXTB.L	Dx	1(0/0)	—	—	—	—	—	—	—
FF1	Dx	1(0/0)	—	—	—	—	—	—	—
NEG.L	Dx	1(0/0)	—	—	—	—	—	—	—
NEGX.L	Dx	1(0/0)	—	—	—	—	—	—	—
NOT.L	Dx	1(0/0)	—	—	—	—	—	—	—
SATS.L	Dx	1(0/0)	—	—	—	—	—	—	—
SCC	Dx	1(0/0)	—	—	—	—	—	—	—
SWAP	Dx	1(0/0)	—	—	—	—	—	—	—
TAS.B	<ea>	—	3 (1/1)	3 (1/1)	3 (1/1)	3 (1/1)	4 (1/1)	3 (1/1)	—
TST.B	<ea>	1(0/0)	2 (1/0)	2 (1/0)	2 (1/0)	2 (1/0)	3 (1/0)	2 (1/0)	1(0/0)
TST.W	<ea>	1(0/0)	2 (1/0)	2 (1/0)	2 (1/0)	2 (1/0)	3 (1/0)	2 (1/0)	1(0/0)
TST.L	<ea>	1(0/0)	2 (1/0)	2 (1/0)	2 (1/0)	2 (1/0)	3 (1/0)	2 (1/0)	1(0/0)

11.3.4.4 Standard Two Operand Instruction Execution Times

Table 11-28. Two Operand Instruction Execution Times

Opcode	<EA>	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An) (d16,PC)	(d8,An,Xn*SF) (d8,PC,Xn*SF)	xxx.wl	#xxx
ADD.L	<ea>,Rx	1(0/0)	3 (1/0)	3 (1/0)	3 (1/0)	3 (1/0)	4 (1/0)	3 (1/0)	1(0/0)
ADD.L	Dy,<ea>	—	3 (1/1)	3 (1/1)	3 (1/1)	3 (1/1)	4 (1/1)	3 (1/1)	—
ADDI.L	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
ADDQ.L	#imm,<ea>	1(0/0)	3 (1/1)	3 (1/1)	3 (1/1)	3 (1/1)	4 (1/1)	3 (1/1)	—
ADDX.L	Dy,Dx	1(0/0)	—	—	—	—	—	—	—
AND.L	<ea>,Rx	1(0/0)	3 (1/0)	3 (1/0)	3 (1/0)	3 (1/0)	4 (1/0)	3 (1/0)	1(0/0)
AND.L	Dy,<ea>	—	3 (1/1)	3 (1/1)	3 (1/1)	3 (1/1)	4 (1/1)	3 (1/1)	—
ANDI.L	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
ASL.L	<ea>,Dx	1(0/0)	—	—	—	—	—	—	1(0/0)
ASR.L	<ea>,Dx	1(0/0)	—	—	—	—	—	—	1(0/0)
BCHG	Dy,<ea>	2(0/0)	4 (1/1)	4 (1/1)	4 (1/1)	4 (1/1)	5 (1/1)	4 (1/1)	—

Table continues on the next page...

Table 11-28. Two Operand Instruction Execution Times (continued)

Opcode	<EA>	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An) (d16,PC)	(d8,An,Xn*SF) (d8,PC,Xn*SF)	xxx.wl	#xxx
BCHG	#imm,<ea>	2(0/0)	4 (1/1)	4 (1/1)	4 (1/1)	4 (1/1)	—	—	—
BCLR	Dy,<ea>	2(0/0)	4 (1/1)	4 (1/1)	4 (1/1)	4 (1/1)	5 (1/1)	4 (1/1)	—
BCLR	#imm,<ea>	2(0/0)	4 (1/1)	4 (1/1)	4 (1/1)	4 (1/1)	—	—	—
BSET	Dy,<ea>	2(0/0)	4 (1/1)	4 (1/1)	4 (1/1)	4 (1/1)	5 (1/1)	4 (1/1)	—
BSET	#imm,<ea>	2(0/0)	4 (1/1)	4 (1/1)	4 (1/1)	4 (1/1)	—	—	—
BTST	Dy,<ea>	2(0/0)	3 (1/0)	3 (1/0)	3 (1/0)	3 (1/0)	4 (1/0)	3 (1/0)	—
BTST	#imm,<ea>	1(0/0)	3 (1/0)	3 (1/0)	3 (1/0)	3 (1/0)	—	—	—
CMP.B	<ea>,Rx	1(0/0)	3 (1/0)	3 (1/0)	3 (1/0)	3 (1/0)	4 (1/0)	3 (1/0)	1(0/0)
CMP.W	<ea>,Rx	1(0/0)	3 (1/0)	3 (1/0)	3 (1/0)	3 (1/0)	4 (1/0)	3 (1/0)	1(0/0)
CMP.L	<ea>,Rx	1(0/0)	3 (1/0)	3 (1/0)	3 (1/0)	3 (1/0)	4 (1/0)	3 (1/0)	1(0/0)
CMPI.B	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
CMPI.W	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
CMPI.L	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
DIVS.W	<ea>,Dx	20(0/0)	23 (1/0)	23 (1/0)	23 (1/0)	23 (1/0)	24 (1/0)	23 (1/0)	20(0/0)
DIVU.W	<ea>,Dx	20(0/0)	23 (1/0)	23 (1/0)	23 (1/0)	23 (1/0)	24 (1/0)	23 (1/0)	20(0/0)
DIVS.L	<ea>,Dx	≤35(0/0)	≤38 (1/0)	≤38 (1/0)	≤38 (1/0)	≤38 (1/0)	—	—	—
DIVU.L	<ea>,Dx	≤35(0/0)	≤38(1/0)	≤38 (1/0)	≤38 (1/0)	≤38 (1/0)	—	—	—
EOR.L	Dy,<ea>	1(0/0)	3 (1/1)	3 (1/1)	3 (1/1)	3 (1/1)	4 (1/1)	3 (1/1)	—
EORI.L	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
LEA	<ea>,Ax	—	1(0/0)	—	—	1(0/0)	2(0/0)	1(0/0)	—
LSL.L	<ea>,Dx	1(0/0)	—	—	—	—	—	—	1(0/0)
LSR.L	<ea>,Dx	1(0/0)	—	—	—	—	—	—	1(0/0)
MOVEQ.L	#imm,Dx	—	—	—	—	—	—	—	1(0/0)
OR.L	<ea>,Rx	1(0/0)	3 (1/0)	3 (1/0)	3 (1/0)	3 (1/0)	4 (1/0)	3 (1/0)	1(0/0)
OR.L	Dy,<ea>	—	3 (1/1)	3 (1/1)	3 (1/1)	3 (1/1)	4 (1/1)	3 (1/1)	—
ORI.L	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
REMS.L	<ea>,Dx	≤35(0/0)	≤38 (1/0)	≤38 (1/0)	≤38 (1/0)	≤38 (1/0)	—	—	—
REMU.L	<ea>,Dx	≤35(0/0)	≤38 (1/0)	≤38 (1/0)	≤38 (1/0)	≤38 (1/0)	—	—	—
SUB.L	<ea>,Rx	1(0/0)	3 (1/0)	3 (1/0)	3 (1/0)	3 (1/0)	4 (1/0)	3 (1/0)	1(0/0)
SUB.L	Dy,<ea>	—	3 (1/1)	3 (1/1)	3 (1/1)	3 (1/1)	4 (1/1)	3 (1/1)	—
SUBI.L	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
SUBQ.L	#imm,<ea>	1(0/0)	3 (1/1)	3 (1/1)	3 (1/1)	3 (1/1)	4 (1/1)	3 (1/1)	—
SUBX.L	Dy,Dx	1(0/0)	—	—	—	—	—	—	—

11.3.4.5 Miscellaneous Instruction Execution Times

Table 11-29. Miscellaneous Instruction Execution Times

Opcode	<EA>	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xn*SF)	xxx.wl	#xxx
LINK.W	Ay,#imm	2(0/1)	—	—	—	—	—	—	—
MOV3Q.L	#imm,<ea>	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
MOVE.L	Ay,USP	3(0/0)	—	—	—	—	—	—	—
MOVE.L	USP,Ax	3(0/0)	—	—	—	—	—	—	—
MOVE.W	CCR,Dx	1(0/0)	—	—	—	—	—	—	—
MOVE.W	<ea>,CCR	1(0/0)	—	—	—	—	—	—	1(0/0)
MOVE.W	SR,Dx	1(0/0)	—	—	—	—	—	—	—
MOVE.W	<ea>,SR	7 (0/0)	—	—	—	—	—	—	7 (0/0) [1]
MOVEC	Ry,Rc	9 (0/1)	—	—	—	—	—	—	—
MOVEM.L	<ea>,and list	—	1+ n(n/0) [2]	—	—	1+ n(n/0)	—	—	—
MOVEM.L	and list,<ea>	—	1+ n(0/n)	—	—	1+ n(0/n)	—	—	—
MVS	<ea>,Dx	1(0/0)	2 (1/0)	2 (1/0)	2 (1/0)	2 (1/0)	3 (1/0)	2 (1/0)	1(0/0)
MVZ	<ea>,Dx	1(0/0)	2 (1/0)	2 (1/0)	2 (1/0)	2 (1/0)	3 (1/0)	2 (1/0)	1(0/0)
NOP		3 (0/0)	—	—	—	—	—	—	—
PEA	<ea>	—	2 (0/1)	—	—	2 (0/1) [3]	3 (0/1) [4]	2 (0/1)	—
PULSE		1(0/0)	—	—	—	—	—	—	—
STLDSR	#imm	—	—	—	—	—	—	—	5(0/1)
STOP	#imm	—	—	—	—	—	—	—	3 (0/0) [5]
TRAP	#imm	—	—	—	—	—	—	—	13 (1/2)
TPF		1(0/0)	—	—	—	—	—	—	—
TPF.W		1(0/0)	—	—	—	—	—	—	—
TPF.L		1(0/0)	—	—	—	—	—	—	—
UNLK	Ax	2 (1/0)	—	—	—	—	—	—	—
WDDATA	<ea>	—	3 (1/0)	3 (1/0)	3 (1/0)	3 (1/0)	4 (1/0)	3 (1/0)	—
WDEBUG	<ea>	—	5 (2/0)	—	—	5 (2/0)	—	—	—

1. If a MOVE.W #imm,SR instruction is executed and imm[13] equals 1, the execution time is 1(0/0).
2. The n is the number of registers moved by the MOVEM opcode.
3. PEA execution times are the same for (d16,PC).
4. PEA execution times are the same for (d8,PC,Xn*SF).
5. The execution time for STOP is the time required until the processor begins sampling continuously for interrupts.

11.3.4.6 EMAC Instruction Execution Times

Table 11-30. EMAC Instruction Execution Times

Opcode	<EA>	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An, Xn*SF)	xxx.wl	#xxx
MAC.L	Ry, Rx, Raccx	1(0/0)	—	—	—	—	—	—	—
MAC.L	Ry, Rx, <ea>, Rw, Raccx	—	2 (1/0)	2 (1/0)	2 (1/0)	2 (1/0) [1]	—	—	—
MAC.W	Ry, Rx, Raccx	1(0/0)	—	—	—	—	—	—	—
MAC.W	Ry, Rx, <ea>, Rw, Raccx	—	2 (1/0)	2 (1/0)	2 (1/0)	2 (1/0)[1]	—	—	—
MOVE.L	<ea>y, Raccx	1(0/0)	—	—	—	—	—	—	1(0/0)
MOVE.L	Raccy,Raccx	1(0/0)	—	—	—	—	—	—	—
MOVE.L	<ea>y, MACSR	5 (0/0)	—	—	—	—	—	—	5 (0/0)
MOVE.L	<ea>y, Rmask	4 (0/0)	—	—	—	—	—	—	4 (0/0)
MOVE.L	<ea>y,Raccext01	1(0/0)	—	—	—	—	—	—	1(0/0)
MOVE.L	<ea>y,Raccext23	1(0/0)	—	—	—	—	—	—	1(0/0)
MOVE.L	Raccx,<ea>x	1(0/0) [2]	—	—	—	—	—	—	—
MOVE.L	MACSR,<ea>x	1(0/0)	—	—	—	—	—	—	—
MOVE.L	Rmask, <ea>x	1(0/0)	—	—	—	—	—	—	—
MOVE.L	Raccext01,<ea.x	1(0/0)	—	—	—	—	—	—	—
MOVE.L	Raccext23,<ea>x	1(0/0)	—	—	—	—	—	—	—
MOVECLR.L	ACCy,Rx	1(0/0)	—	—	—	—	—	—	—
MSAC.L	Ry, Rx, Raccx	1(0/0)	—	—	—	—	—	—	—
MSAC.W	Ry, Rx, Raccx	1(0/0)	—	—	—	—	—	—	—
MSAC.L	Ry, Rx, <ea>, Rw, Raccx	—	2 (1/0)	2 (1/0)	2 (1/0)	2 (1/0) [1]	—	—	—
MSAC.W	Ry, Rx, <ea>, Rw, Raccx	—	2 (1/0)	2 (1/0)	2 (1/0)	2 (1/0) [1]	—	—	—
MULS.L	<ea>y, Dx	3 (0/0)	5 (1/0)	5 (1/0)	5 (1/0)	5 (1/0)	—	—	—
MULS.W	<ea>y, Dx	3 (0/0)	5 (1/0)	5 (1/0)	5 (1/0)	5 (1/0)	6 (1/0)	5 (1/0)	3 (0/0)
MULU.L	<ea>y, Dx	3 (0/0)	5 (1/0)	5 (1/0)	5 (1/0)	5 (1/0)	—	—	—
MULU.W	<ea>y, Dx	3 (0/0)	5 (1/0)	5 (1/0)	5 (1/0)	5 (1/0)	6 (1/0)	5 (1/0)	3 (0/0)

1. Effective address of (d16,PC) not supported
2. Storing an accumulator requires one additional processor clock cycle when saturation is enabled, or fractional rounding is performed (MACSR[7:4] equals 1---, -11-, --11)

Note

The execution times for moving the contents of the Racc, Raccext[01,23], MACSR, or Rmask into a destination location <ea>x shown in this table represent the best-case scenario when the store instruction is executed and there are no load or

M{S}AC instructions in the EMAC execution pipeline. In general, these store operations require only a single cycle for execution, but if they are preceded immediately by a load, MAC, or MSAC instruction, the depth of the EMAC pipeline is exposed and the execution time is three cycles.

11.3.4.7 Branch Instruction Execution Times

Table 11-31. General Branch Instruction Execution Times

Opcode	<EA>	Effective Address							
		Rn	(An)	(An)+	-(An)	(d_{16},An) (d_{16},PC)	(d8,An,Xi*SF) (d8,PC,Xi*SF)	xxx.wl	#xxx
BRA		—	—	—	—	2 (0/1)	—	—	—
BSR		—	—	—	—	3 (0/1)	—	—	—
JMP	<ea>	—	3 (0/0)	—	—	3 (0/0)	4 (0/0)	3 (0/0)	—
JSR	<ea>	—	3 (0/1)	—	—	3 (0/1)	4 (0/1)	3 (0/1)	—
RTE		—	—	7 (2/0)	—	—	—	—	—
RTS		—	—	5 (1/0)	—	—	—	—	—

Table 11-32. Bcc Instruction Execution Times

Opcode	Forward Taken	Forward Not Taken	Backward Taken	Backward Not Taken
Bcc	3 (0/0)	1(0/0)	2 (0/0)	3 (0/0)

Chapter 12
Enhanced Multiply-Accumulate Unit (EMAC)

12.1 Introduction

This chapter describes the functionality, microarchitecture, and performance of the enhanced multiply-accumulate (EMAC) unit in the ColdFire family of processors.

12.1.1 Overview

The EMAC design provides a set of DSP operations that can improve the performance of embedded code while supporting the integer multiply instructions of the baseline ColdFire architecture.

The EMAC provides functionality in three related areas:

1. Signed and unsigned integer multiplication
2. Multiply-accumulate operations supporting signed and unsigned integer operands as well as signed, fixed-point, and fractional operands
3. Miscellaneous register operations

The ColdFire family supports two MAC implementations with different performance levels and capabilities. The original MAC features a three-stage execution pipeline optimized for 16-bit operands, with a 16×16 multiply array and a single 32-bit accumulator. The EMAC features a three-stage pipeline optimized for 32-bit operands, with a fully pipelined 32×32 multiply array and four 48-bit accumulators.

The first ColdFire MAC supported signed and unsigned integer operands and was optimized for 16x16 operations, such as those found in applications including servo control and image compression. As ColdFire-based systems proliferated, the desire for more precision on input operands increased. The result was an improved ColdFire MAC with user-programmable control to optionally enable use of fractional input operands.

EMAC improvements target three primary areas:

- Improved performance of 32 × 32 multiply operation.
- Addition of three more accumulators to minimize MAC pipeline stalls caused by exchanges between the accumulator and the pipeline's general-purpose registers
- A 48-bit accumulation data path to allow a 40-bit product, plus 8 extension bits increase the dynamic number range when implementing signal processing algorithms

The three areas of functionality are addressed in detail in following sections. The logic required to support this functionality is contained in a MAC module as shown below.

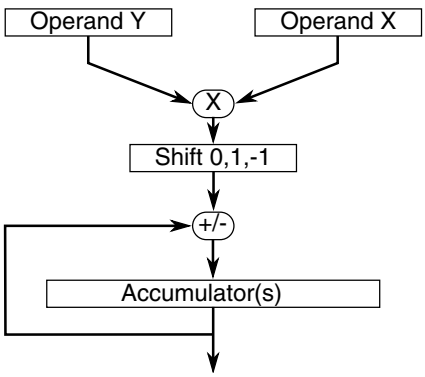


Figure 12-1. Multiply-Accumulate Functionality Diagram

12.1.1.1 Introduction to the MAC

The MAC is an extension of the basic multiplier in most microprocessors. It is typically implemented in hardware within an architecture and supports rapid execution of signal processing algorithms in fewer cycles than comparable non-MAC architectures. For example, small digital filters can tolerate some variance in an algorithm's execution time, but larger, more complicated algorithms such as orthogonal transforms may have more demanding speed requirements beyond scope of any processor architecture and may require full DSP implementation.

To balance speed, size, and functionality, the ColdFire MAC is optimized for a small set of operations that involve multiplication and cumulative additions. Specifically, the multiplier array is optimized for single-cycle pipelined operations with a possible accumulation after product generation. This functionality is common in many signal processing applications. The ColdFire core architecture is also modified to allow an operand to be fetched in parallel with a multiply, increasing overall performance for certain DSP operations.

Consider a typical filtering operation where the filter is defined as in the following equation.

$$y(i) = \sum_{k=1}^{N-1} a(k)y(i-k) + \sum_{k=0}^{N-1} b(k)x(i-k)$$

Here, the output y(i) is determined by past output values and past input values. This is the general form of an infinite impulse response (IIR) filter. A finite impulse response (FIR) filter can be obtained by setting coefficients a(k) to zero. In either case, the operations involved in computing such a filter are multiplies and product summing. To show this point, reduce the preceding equation to a simple, four-tap FIR filter, shown in the following equation, in which the accumulated sum is a past data values and coefficients sum.

$$y(i) = \sum_{k=0}^{3} b(k)\,x\,(i-k) = b(0)\,x\,(i) + b(1)\,x\,(i-1) + b(2)\,x\,(i-2) + b(3)\,x\,(i-3)$$

12.2 Memory Map/Register Definition

The following table and sections explain the MAC registers.

Table 12-1. EMAC Memory Map

BDM	Register	Width (bits)	Access	Reset Value
Read: 0xE4 Write: 0xC4	MAC status register (MACSR)	32	R/W	0x0000_0000
Read: 0xE5 Write: 0xC5	MAC address mask register (MASK)	32	R/W	0xFFFF_FFFF
Read: 0xE6 Write: 0xC6	MAC accumulator 0 (ACC0)	32	R/W	Undefined
Read: 0xE7 Write: 0xC7	MAC accumulator 0,1 extension bytes (ACCext01)	32	R/W	Undefined
Read: 0xE8 Write: 0xC8	MAC accumulator 2,3 extension bytes (ACCext23)	32	R/W	Undefined
Read: 0xE9 Write: 0xC9	MAC accumulator 1 (ACC1)	32	R/W	Undefined
Read: 0xEA Write: 0xCA	MAC accumulator 2 (ACC2)	32	R/W	Undefined
Read: 0xEB Write: 0xCB	MAC accumulator 2 (ACC3)	32	R/W	Undefined

12.2.1 MAC Status Register (MACSR)

The MAC status register (MACSR) contains a 4-bit operational mode field and condition flags. Operational mode bits control whether operands are signed or unsigned and whether they are treated as integers or fractions. These bits also control the overflow/saturation mode and the way in which rounding is performed. Negative, zero, and multiple overflow condition flags are also provided.

Table 12-2. MAC Status Register (MACSR)

BDM:	Read: 0xE4 (MACSR) Write: 0xC4												Access: Supervisor read/write BDM read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	PAV*n*				OMC	S/U	F/I	R/T	N	Z	V	EV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-3. MACSR Field Descriptions

Field	Description
31–12	Reserved, must be cleared.
11–8 PAV*n*	Product/accumulation overflow flags. Contains four flags, one per accumulator, that indicate if past MAC or MSAC instructions generated an overflow during product calculation or the 48-bit accumulation. When a MAC or MSAC instruction is executed, the PAVn flag associated with the destination accumulator forms the general overflow flag, MACSR[V]. Once set, each flag remains set until V is cleared by a move.l, MACSR instruction or the accumulator is loaded directly. Bit 11: Accumulator 3 . . . Bit 8: Accumulator 0
7 OMC	Overflow saturation mode. Enables or disables saturation mode on overflow. If set, the accumulator is set to the appropriate constant (see S/U field description) on any operation that overflows the accumulator. After saturation, the accumulator remains unaffected by any other MAC or MSAC instructions until the overflow bit is cleared or the accumulator is directly loaded.

Table continues on the next page...

Table 12-3. MACSR Field Descriptions (continued)

Field	Description	
6 S/U	Signed/unsigned operations. **In integer mode:** S/U determines whether operations performed are signed or unsigned. It also determines the accumulator value during saturation, if enabled.	
	0	Signed numbers. On overflow, if OMC is enabled, an accumulator saturates to the most positive (0x7FFF_FFFF) or the most negative (0x8000_0000) number, depending on the instruction and the product value that overflowed.
	1	Unsigned numbers. On overflow, if OMC is enabled, an accumulator saturates to the smallest value (0x0000_0000) or the largest value (0xFFFF_FFFF), depending on the instruction.
	In fractional mode: S/U controls rounding while storing an accumulator to a general-purpose register.	
	0	Move accumulator without rounding to a 16-bit value. Accumulator is moved to a general-purpose register as a 32-bit value.
	1	The accumulator is rounded to a 16-bit value using the round-to-nearest (even) method when moved to a general-purpose register. The resulting 16-bit value is stored in the lower word of the destination register. The upper word is zero-filled. This rounding procedure does not affect the accumulator value.
5 F/I	Fractional/integer mode. Determines whether input operands are treated as fractions or integers.	
	0	Integers can be represented in signed or unsigned notation, depending on the value of S/U.
	1	Fractions are represented in signed, fixed-point, two's complement notation. Values range from –1 to $1 - 2^{-15}$ for 16-bit fractions and –1 to $1 - 2^{-31}$ for 32-bit fractions.
4 R/T	Round/truncate mode. Controls rounding procedure for move.l ACCx,Rx, or MSAC.L instructions when in fractional mode.	
	0	Truncate. The product's lsbs are dropped before it is combined with the accumulator. Additionally, when a store accumulator instruction is executed (move.l ACCx,Rx), the 8 lsbs of the 48-bit accumulator logic are truncated.
	1	Round-to-nearest (even). The 64-bit product of two 32-bit, fractional operands is rounded to the nearest 40-bit value. If the low-order 24 bits equal 0x80_0000, the upper 40 bits are rounded to the nearest even (lsb = 0) value. Additionally, when a store accumulator instruction is executed (move.l ACCx,Rx), the lsbs of the 48-bit accumulator logic round the resulting 16- or 32-bit value. If MACSR[S/U] is cleared and MACSR[R/T] is set, the low-order 8 bits are used to round the resulting 32-bit fraction. If MACSR[S/U] is set, the low-order 24 bits are used to round the resulting 16-bit fraction.
3 N	Negative. Set if the msb of the result is set, otherwise cleared. N is affected only by MAC, MSAC, and load operations; it is not affected by MULS and MULU instructions.	
2 Z	Zero. Set if the result equals zero, otherwise cleared. This bit is affected only by MAC, MSAC, and load operations; it is not affected by MULS and MULU instructions.	
1 V	Overflow. Set if an arithmetic overflow occurs on a MAC or MSAC instruction, indicating that the result cannot be represented in the limited width of the EMAC. V is set only if a product overflow occurs or the accumulation overflows the 48-bit structure. V is evaluated on each MAC or MSAC operation and uses the appropriate PAVn flag in the next-state V evaluation.	
0 EV	Extension overflow. Signals that the last MAC or MSAC instruction overflowed the 32 lsbs in integer mode or the 40 lsbs in fractional mode of the destination accumulator. However, the result remains accurately represented in the combined 48-bit accumulator structure. Although an overflow has occurred, the correct result, sign, and magnitude are contained in the 48-bit accumulator. Subsequent MAC or MSAC operations may return the accumulator to a valid 32/40-bit result.	

This table summarizes the interaction of the MACSR[S/U,F/I,R/T] control bits.

Table 12-4. Summary of S/U, F/I, and R/T Control Bits

S/U	F/I	R/T	Operational Modes
0	0	x	Signed, integer
0	1	0	Signed, fractional Truncate on MAC.L and MSAC.L No round on accumulator stores
0	1	1	Signed, fractional Round on MAC.L and MSAC.L Round-to-32-bits on accumulator stores
1	0	x	Unsigned, integer
1	1	0	Signed, fractional Truncate on MAC.L and MSAC.L Round-to-16-bits on accumulator stores
1	1	1	Signed, fractional Round on MAC.L and MSAC.L Round-to-16-bits on accumulator stores

12.2.2 Mask Register (MASK)

The 32-bit MASK implements the low-order 16 bits to minimize the alignment complications involved with loading and storing only 16 bits. When the MASK is loaded, the low-order 16 bits of the source operand are actually loaded into the register. When it is stored, the upper 16 bits are all forced to ones. This register performs a simple AND with the operand address for MAC instructions. The processor calculates the normal operand address and, if enabled, that address is then ANDed with {0xFFFF, MASK[15:0]} to form the final address. Therefore, with certain MASK bits cleared, the operand address can be constrained to a certain memory region. This is used primarily to implement circular queues with the (An)+ addressing mode.

This minimizes the addressing support required for filtering, convolution, or any routine that implements a data array as a circular queue. For MAC + MOVE operations, the MASK contents can optionally be included in all memory effective address calculations. The syntax is as follows:

```
mac.sz  Ry,RxSF,<ea>y&,Rw
```

The & operator enables the MASK use and causes bit 5 of the extension word to be set. The exact algorithm for the use of MASK is:

```
if extension word, bit [5] = 1, the MASK bit, then
if <ea> = (An)
oa  =  An & {0xFFFF, MASK}

if <ea> = (An)+
oa  =  An
An  = (An + 4) & {0xFFFF, MASK}

if <ea> =-(An)
oa  = (An - 4) & {0xFFFF, MASK}
An  = (An - 4) & {0xFFFF, MASK}

if <ea> = (d16,An)
oa  = (An + se_d16) & {0xFFFF0x, MASK}
```

Here, oa is the calculated operand address and se_d16 is a sign-extended 16-bit displacement. For auto-addressing modes of post-increment and pre-decrement, the updated An value calculation is also shown.

Use of the post-increment addressing mode, {(An)+} with the MASK is suggested for circular queue implementations.

Table 12-5. Mask Register (MASK)

BDM:	0x5 (MASK)												Access: User read/write BDM read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MASK															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 12-6. MASK Field Descriptions

Field	Description
31–12	Reserved; must be set.

Table continues on the next page...

Table 12-6. MASK Field Descriptions (continued)

Field	Description
15–0 MASK	Performs a simple AND with the operand address for MAC instructions.

12.2.3 Accumulator Registers (ACC0-3)

Each accumulator register stores 32 bits of the MAC operation result. The entire 48-bit accumulator result consists of the accumulator register concatenated with the corresponding fields of the accumulator extension registers.

Table 12-7. Accumulator Registers (ACC0-3)

BDM:	Read: 0xE6 (ACC0) Write: 0xC6 Read: 0xE9 (ACC1) Write: 0xC9 Read: 0xEA (ACC2) Write: 0xCA Read: 0xEB (ACC3) Write: 0xCB												Access: User read/write BDM read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Accumulator															
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Accumulator															
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 12-8. ACC0-3 Field Descriptions

Field	Description
31–0 Accumulator	Store 32-bits of the result of the MAC operation.

12.2.4 Accumulator Extension Registers (ACCext01, ACCext23)

Each pair of 8-bit accumulator extension fields are concatenated with the corresponding 32-bit accumulator register to form the 48-bit accumulator. For more information, see the functional description.

Table 12-9. Accumulator Extension Register (ACCext01)

BDM:	Read: 0xE7 (ACCext01) Write: 0xC7												Access: User read/write BDM read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ACC0U								ACC0L							
W																
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ACC1U								ACC1L							
W																
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 12-10. ACCext01 Field Descriptions

Field	Description
31–24 ACC0U	Accumulator 0 upper extension byte
23–16 ACC0L	Accumulator 0 lower extension byte
15–8 ACC1U	Accumulator 1 upper extension byte
7–0 ACC1L	Accumulator 1 lower extension byte

Table 12-11. Accumulator Extension Register (ACCext23)

BDM:	Read: 0xE8 (ACCext01) Write: 0xC8												Access: User read/write BDM read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ACC2U								ACC2L							
W																
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table continues on the next page...

Table 12-11. Accumulator Extension Register (ACCext23) (continued)

R W	ACC3U								ACC3L							
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 12-12. ACCext23 Field Descriptions

Field	Description
31–24 ACC2U	Accumulator 2 upper extension byte
23–16 ACC2L	Accumulator 2 lower extension byte
15–8 ACC3U	Accumulator 3 upper extension byte
7–0 ACC3L	Accumulator 3 lower extension byte

12.3 Functional Description

The MAC speeds execution of ColdFire integer-multiply instructions (MULS and MULU) and provides additional functionality for multiply-accumulate operations. By executing MULS and MULU in the MAC, execution times are minimized and deterministic compared to the 2-bit/cycle algorithm with early termination that the OEP normally uses if no MAC hardware is present.

The added MAC instructions to the ColdFire ISA provide for the multiplication of two numbers, followed by the addition or subtraction of the product to or from the value in an accumulator. Optionally, the product may be shifted left or right by 1 bit before addition or subtraction. Hardware support for saturation arithmetic can be enabled to minimize software overhead when dealing with potential overflow conditions. Multiply-accumulate operations support 16- or 32-bit input operands in these formats:

- Signed integers
- Unsigned integers
- Signed, fixed-point, fractional numbers

The EMAC is optimized for single-cycle, pipelined 32 × 32 multiplications. For word- and longword-sized integer input operands, the low-order 40 bits of the product are formed and used with the destination accumulator. For fractional operands, the entire 64-

bit product is calculated and truncated or rounded to the most-significant 40-bit result using the round-to-nearest (even) method before it is combined with the destination accumulator.

For all operations, the resulting 40-bit product is extended to a 48-bit value (using sign-extension for signed integer and fractional operands, zero-fill for unsigned integer operands) before being combined with the 48-bit destination accumulator.

The following two figures show relative alignment of input operands, the full 64-bit product, the resulting 40-bit product used for accumulation, and 48-bit accumulator formats.

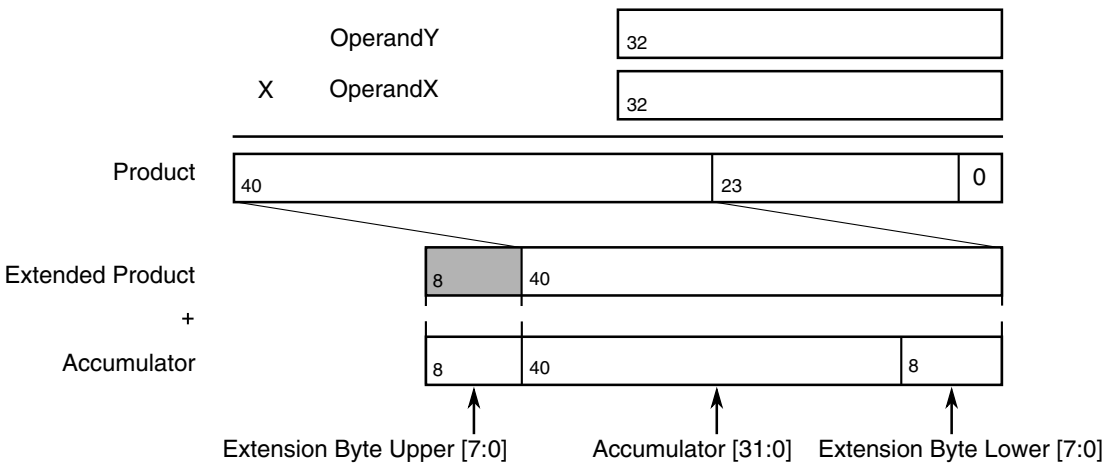


Figure 12-2. Fractional Alignment

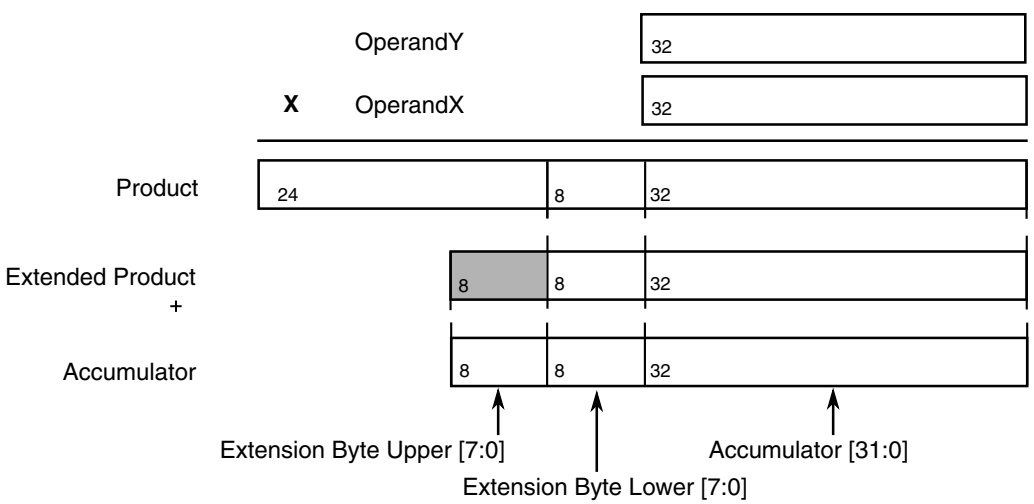


Figure 12-3. Signed and Unsigned Integer Alignment

Therefore, the 48-bit accumulator definition is a function of the EMAC operating mode. Given that each 48-bit accumulator is the concatenation of 16-bit accumulator extension register (ACCext*n*) contents and 32-bit ACC*n* contents, the specific definitions are:

```
if MACSR[6:5] == 00           /* signed integer mode */
        Complete Accumulator[47:0] = {ACCextn[15:0], ACCn[31:0]}
if MACSR[6:5] == 01 or 11     /* signed fractional mode */
        Complete Accumulator [47:0] = {ACCextn[15:8], ACCn[31:0], ACCextn[7:0]}
if MACSR[6:5] == 10           /* unsigned integer mode */
        Complete Accumulator[47:0] = {ACCextn[15:0], ACCn[31:0]}
```

The four accumulators are represented as an array, ACC*n*, where *n* selects the register.

Although the multiplier array is implemented in a three-stage pipeline, all arithmetic MAC instructions have an effective issue rate of 1 cycle, regardless of input operand size or type.

All arithmetic operations use register-based input operands, and summed values are stored in an accumulator. Therefore, an additional MOVE instruction is needed to store data in a general-purpose register. One new feature in EMAC instructions is the ability to choose the upper or lower word of a register as a 16-bit input operand. This is useful in filtering operations if one data register is loaded with the input data and another is loaded with the coefficient. Two 16-bit multiply accumulates can be performed without fetching additional operands between instructions by alternating word choice during calculations.

The EMAC has four accumulator registers versus the MAC's single accumulator. The additional registers improve the performance of some algorithms by minimizing pipeline stalls needed to store an accumulator value back to general-purpose registers. Many algorithms require multiple calculations on a given data set. By applying different accumulators to these calculations, it is often possible to store one accumulator without any stalls while performing operations involving a different destination accumulator.

The need to move large amounts of data presents an obstacle to obtaining high throughput rates in DSP engines. Existing ColdFire instructions can accommodate these requirements. A MOVEM instruction can efficiently move large data blocks. The ability to load an operand simultaneously from memory into a register and execute a MAC instruction makes some DSP operations such as filtering and convolution more manageable.

The programming model includes a mask register (MASK), which can optionally be used to generate an operand address during MAC + MOVE instructions. The register application with auto-increment addressing mode supports efficient implementation of circular data queues for memory operands.

12.3.1 Fractional Operation Mode

This section describes behavior when the fractional mode is used (MACSR[F/I] is set).

12.3.1.1 Rounding

When the processor is in fractional mode, there are two operations during which rounding can occur:

1. Execution of a store accumulator instruction (move.l ACCx,Rx). The lsbs of the 48-bit accumulator logic are used to round the resulting 16- or 32-bit value. If MACSR[S/U] is cleared, the low-order 8 bits round the resulting 32-bit fraction. If MACSR[S/U] is set, the low-order 24 bits are used to round the resulting 16-bit fraction.
2. Execution of a MAC (or MSAC) instruction with 32-bit operands. If MACSR[R/T] is zero, multiplying two 32-bit numbers creates a 64-bit product truncated to the upper 40 bits; otherwise, it is rounded using round-to-nearest (even) method.

To understand the round-to-nearest-even method, consider the following example involving the rounding of a 32-bit number, R0, to a 16-bit number. Using this method, the 32-bit number is rounded to the closest 16-bit number possible. Let the high-order 16 bits of R0 be named R0.U and the low-order 16 bits be R0.L.

- If R0.L is less than 0x8000, the result is truncated to the value of R0.U.
- If R0.L is greater than 0x8000, the upper word is incremented (rounded up).
- If R0.L is 0x8000, R0 is half-way between two 16-bit numbers. In this case, rounding is based on the lsb of R0.U, so the result is always even (lsb = 0).
  - If the lsb of R0.U equals 1 and R0.L equals 0x8000, the number is rounded up.
  - If the lsb of R0.U equals 0 and R0.L equals 0x8000, the number is rounded down.

This method minimizes rounding bias and creates as statistically correct an answer as possible.

The rounding algorithm is summarized in the following pseudocode:

```
if R0.L < 0x8000
         then Result = R0.U
else if R0.L > 0x8000
         then Result = R0.U + 1
else if lsb of R0.U = 0                 /* R0.L = 0x8000 */
         then Result = R0.U
else Result = R0.U + 1
```

The round-to-nearest-even technique is also known as convergent rounding.

12.3.1.2 Saving and Restoring the EMAC Programming Model

The presence of rounding logic in the EMAC output datapath requires special care during the EMAC's save/restore process. In particular, any result rounding modes must be disabled during the save/restore process so the exact bit-wise contents of the EMAC registers are accessed. Consider the memory structure containing the EMAC programming model:

```
struct   macState {

         int acc0;
         int acc1;
         int acc2;
         int acc3;
         int accext01;
         int accext02;
         int mask;
         int macsr;
} macState;
```

The following assembly language routine shows the proper sequence for a correct EMAC state save. This code assumes all Dn and An registers are available for use, and the memory location of the state save is defined by A7.

```
EMAC_state_save:
        move.l  macsr,d7                ; save the macsr
        clr.l   d0                      ; zero the register to ...
        move.l  d0,macsr                ; disable rounding in the macsr

        move.l  acc0,d0                 ; save the accumulators
        move.l  acc1,d1
        move.l  acc2,d2
        move.l  acc3,d3
        move.l  accext01,d4             ; save the accumulator extensions
        move.l  accext23,d5
        move.l  mask,d6                 ; save the address mask

        movem.l #0x00ff,(a7)            ; move the state to memory
```

This code performs the EMAC state restore:

```
EMAC_state_restore:


        movem.l (a7),#0x00ff                ; restore the state from memory

        move.l  d5,acc                      ; restore the accumulator
        move.l  d0,acc0                     ; restore the accumulators
        move.l  d1,acc1
        move.l  d2,acc2
        move.l  d3,acc3
        move.l  d4,accext01                  ; restore the accumulator extensions
        move.l  d5,accext23
        move.l  d6,mask                      ; restore the address mask
        move.l  d7,macsr                     ; restore the macsr
```

Executing this sequence type can correctly save and restore the exact state of the EMAC programming model.

12.3.1.3 MULS/MULU

MULS and MULU are unaffected by fractional-mode operation; operands remain assumed to be integers.

12.3.1.4 Scale Factor in MAC or MSAC Instructions

The scale factor is ignored while the MAC is in fractional mode.

12.3.2 EMAC Instruction Set Summary

The following table summarizes EMAC unit instructions.

Table 12-13. EMAC Instruction Summary

Command	Mnemonic	Description
Multiply Accumulate	mac Ry,RxSF,ACCx msac Ry,RxSF,ACCx	Multiplies two operands and adds/subtracts the product to/from an accumulator
Multiply Accumulate with Load	mac Ry,Rx,<ea>y,Rw,ACCx msac Ry,Rx,<ea>y,Rw,ACCx	Multiplies two operands and combines the product to an accumulator while loading a register with the memory operand
Load Accumulator	move.l {Ry,#imm},ACCx	Loads an accumulator with a 32-bit operand
Store Accumulator	move.l ACCx,Rx	Writes the contents of an accumulator to a CPU register
Copy Accumulator	move.l ACCy,ACCx	Copies a 48-bit accumulator
Load MACSR	move.l {Ry,#imm},MACSR	Writes a value to MACSR
Store MACSR	move.l MACSR,Rx	Write the contents of MACSR to a CPU register
Store MACSR to CCR	move.l MACSR,CCR	Write the contents of MACSR to the CCR
Load MAC Mask Reg	move.l {Ry,#imm},MASK	Writes a value to the MASK register
Store MAC Mask Reg	move.l MASK,Rx	Writes the contents of the MASK to a CPU register
Load Accumulator Extensions 01	move.l {Ry,#imm},ACCext01	Loads the accumulator 0,1 extension bytes with a 32-bit operand
Load Accumulator Extensions 23	move.l {Ry,#imm},ACCext23	Loads the accumulator 2,3 extension bytes with a 32-bit operand
Store Accumulator Extensions 01	move.l ACCext01,Rx	Writes the contents of accumulator 0,1 extension bytes into a CPU register
Store Accumulator Extensions 23	move.l ACCext23,Rx	Writes the contents of accumulator 2,3 extension bytes into a CPU register
Move and Clear Accumulator	moveclr.l ACCy,Rx	Writes the contents of an accumulator to a destination and then clears the accumulator

12.3.3 EMAC Instruction Execution Times

The instruction execution times for the EMAC can be found in the "EMAC instruction execution times" section of the core chapter.

The EMAC execution pipeline overlaps the AGEX stage of the OEP (the first stage of the EMAC pipeline is the last stage of the basic OEP). EMAC units are designed for sustained, fully-pipelined operation on accumulator load, copy, and multiply-accumulate instructions. However, instructions that store contents of the multiply-accumulate programming model can generate OEP stalls that expose the EMAC execution pipeline depth:

```
mac.w     Ry, Rx, Acc0
move.l    Acc0, Rz
```

The MOVE.L instruction that stores the accumulator to an integer register (Rz) stalls until the program-visible copy of the accumulator is available. The following figure shows EMAC timing.

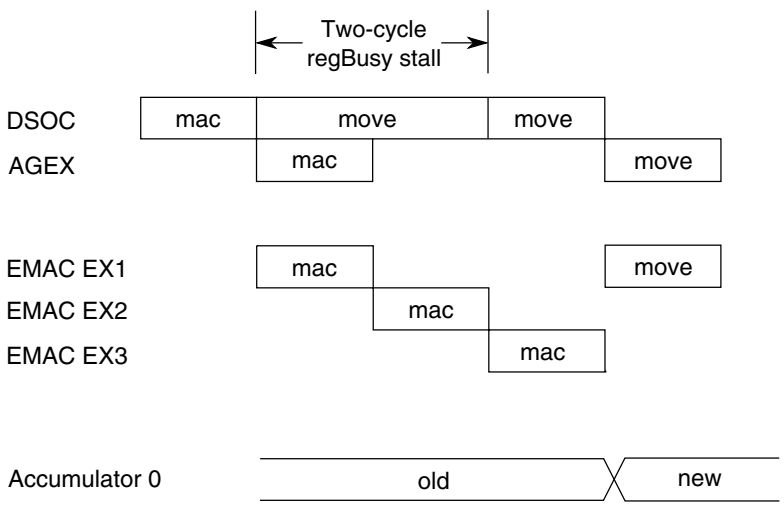


Figure 12-4. EMAC-Specific OEP Sequence Stall

The OEP stalls the store-accumulator instruction for two cycles: the EMAC pipeline depth minus 1. The minus 1 factor is needed because the OEP and EMAC pipelines overlap by a cycle, the AGEX stage. As the store-accumulator instruction reaches the AGEX stage where the operation is performed, the recently updated accumulator 0 value is available.

As with change or use stalls between accumulators and general-purpose registers, introducing intervening instructions that do not reference the busy register can reduce or eliminate sequence-related store-MAC instruction stalls. A major benefit of the EMAC is the addition of three accumulators to minimize stalls caused by exchanges between the accumulator(s) and general-purpose registers.

12.3.4 Data Representation

MACSR[S/U,F/I] selects one of the following three modes, where each mode defines a unique operand type:

1. Two's complement signed integer: In this format, an N-bit operand value lies in the range $-2^{(N-1)} \le$ operand $\le 2^{(N-1)} - 1$. The binary point is right of the lsb.
2. Unsigned integer: In this format, an N-bit operand value lies in the range $0 \le$ operand $\le 2^N - 1$. The binary point is right of the lsb.
3. Two's complement signed fractional: In an N-bit number, the first bit is the sign bit. The remaining bits signify the first N-1 bits after the binary point. Given an N-bit number, $a_{N-1}a_{N-2}a_{N-3}... a_2a_1a_0$, its value is given by the equation in the following equation.

$$\text{value} = -\left(1.a_{N-1}\right) + \sum_{i=0}^{N-2} 2^{-(i+1-N)}.ai$$

This format can represent numbers in the range $-1 \le$ operand $\le 1 - 2^{-(N-1)}$.

For words and longwords, the largest negative number that can be represented is -1, whose internal representation is 0x8000 and 0x8000_0000, respectively. The largest positive word is 0x7FFF or $(1 - 2^{-15})$; the most positive longword is 0x7FFF_FFFF or $(1 - 2^{-31})$. . Thus, the number range for these signed fractional numbers is [-1.0, ..., 1.0].

12.3.5 MAC Opcodes

MAC opcodes are described in the *ColdFire Programmer's Reference Manual*.

Remember the following:

- Unless otherwise noted, the value of MACSR[N,Z] is based on the result of the final operation that involves the product and the accumulator.
- The overflow (V) flag is managed differently. It is set if the complete product cannot be represented as a 40-bit value (this applies to 32 × 32 integer operations only) or if the combination of the product with an accumulator cannot be represented in the

given number of bits. The EMAC design includes an additional product/ accumulation overflow bit for each accumulator that are treated as sticky indicators and are used to calculate the V bit on each MAC or MSAC instruction. See MAC Status Register (MACSR).

- For the MAC design, the assembler syntax of the MAC (multiply and add to accumulator) and MSAC (multiply and subtract from accumulator) instructions does not include a reference to the single accumulator. For the EMAC, assemblers support this syntax and no explicit reference to an accumulator is interpreted as a reference to ACC0. Assemblers also support syntaxes where the destination accumulator is explicitly defined.
- The optional 1-bit shift of the product is specified using the notation {<< | >>} SF, where <<1 indicates a left shift and >>1 indicates a right shift. The shift is performed before the product is added to or subtracted from the accumulator. Without this operator, the product is not shifted. If the EMAC is in fractional mode (MACSR[F/I] is set), SF is ignored and no shift is performed. Because a product can overflow, the following guidelines are implemented:
  - For unsigned word and longword operations, a zero is shifted into the product on right shifts.
  - For signed, word operations, the sign bit is shifted into the product on right shifts unless the product is zero. For signed, longword operations, the sign bit is shifted into the product unless an overflow occurs or the product is zero, in which case a zero is shifted in.
  - For all left shifts, a zero is inserted into the lsb position.

The following pseudocode explains basic MAC or MSAC instruction functionality. This example is presented as a case statement covering the three basic operating modes with signed integers, unsigned integers, and signed fractionals. Throughout this example, a comma-separated list in curly brackets, { }, indicates a concatenation operation.

```
switch (MACSR[6:5])      /* MACSR[S/U, F/I] */
{
   case 0:               /* signed integers */
     if (MACSR.OMC == 0 || MACSR.PAVn == 0)
         then {
               MACSR.PAVn = 0
               /* select the input operands */
               if (sz == word)
                  then {if (U/Ly == 1)
                        then operandY[31:0] = {sign-extended Ry[31], Ry[31:16]}
                         else operandY[31:0] = {sign-extended Ry[15], Ry[15:0]}
                         if (U/Lx == 1)
                        then operandX[31:0] = {sign-extended Rx[31], Rx[31:16]}
                         else operandX[31:0] = {sign-extended Rx[15], Rx[15:0]}
                  }
```

```
                else {operandY[31:0] = Ry[31:0]
                      operandX[31:0] = Rx[31:0]
                }
             /* perform the multiply */
             product[63:0] = operandY[31:0] * operandX[31:0]
             /* check for product overflow */
     if ((product[63:39] != 0x0000_00_0) && (product[63:39] != 0xffff_ff_1))
                then {        /* product overflow */
                      MACSR.PAVn = 1
                      MACSR.V = 1
                      if (inst == MSAC && MACSR.OMC == 1)
                         then if (product[63] == 1)
                                 then result[47:0] = 0x0000_7fff_ffff
                                 else result[47:0] = 0xffff_8000_0000
                         else if (MACSR.OMC == 1)
                                 then /* overflowed MAC,
                                         saturationMode enabled */
                                      if (product[63] == 1)
                                        then result[47:0] = 0xffff_8000_0000
                                        else result[47:0] = 0x0000_7fff_ffff
                }
              /* sign-extend to 48 bits before performing any scaling */
              product[47:40] = {8{product[39]}}    /* sign-extend */
             /* scale product before combining with accumulator */
             switch (SF)      /* 2-bit scale factor */
             {
                 case 0:      /* no scaling specified */
                    break;
                 case 1:      /* SF = "<< 1" */
                    product[40:0] = {product[39:0], 0}
                    break;
                 case 2:      /* reserved encoding */
                    break;
                 case 3:      /* SF = ">> 1" */
                    product[39:0] = {product[39], product[39:1]}
                    break;
             }
             if (MACSR.PAVn == 0)
                then {if (inst == MSAC)
                         then result[47:0] = ACCx[47:0] - product[47:0]
                         else result[47:0] = ACCx[47:0] + product[47:0]
                }
             /* check for accumulation overflow */
             if (accumulationOverflow == 1)
                then {MACSR.PAVn = 1
                      MACSR.V = 1
                      if (MACSR.OMC == 1)
                         then /* accumulation overflow,
                                 saturationMode enabled */
                              if (result[47] == 1)
                                 then result[47:0] = 0x0000_7fff_ffff
                                 else result[47:0] = 0xffff_8000_0000
                }
             /* transfer the result to the accumulator */
             ACCx[47:0] = result[47:0]
       }
       MACSR.V = MACSR.PAVn
       MACSR.N = ACCx[47]
       if (ACCx[47:0] == 0x0000_0000_0000)
          then MACSR.Z = 1
          else MACSR.Z = 0
       if ((ACCx[47:31] == 0x0000_0) || (ACCx[47:31] == 0xffff_1))
          then MACSR.EV = 0
          else MACSR.EV = 1
break;
   case 1,3:              /* signed fractionals */
   if (MACSR.OMC == 0 || MACSR.PAVn == 0)
      then {
             MACSR.PAVn = 0
             if (sz == word)
```

```
                then {if (U/Ly == 1)
                         then operandY[31:0] = {Ry[31:16], 0x0000}
                         else operandY[31:0] = {Ry[15:0],  0x0000}
                      if (U/Lx == 1)
                         then operandX[31:0] = {Rx[31:16], 0x0000}
                         else operandX[31:0] = {Rx[15:0],  0x0000}
                }
                else {operandY[31:0] = Ry[31:0]
                      operandX[31:0] = Rx[31:0]
                }
             /* perform the multiply */
             product[63:0] = (operandY[31:0] * operandX[31:0]) << 1
             /* check for product rounding */
             if (MACSR.R/T == 1)
                then { /* perform convergent rounding */
                      if (product[23:0] > 0x80_0000)
                         then product[63:24] = product[63:24] + 1
                else if ((product[23:0] == 0x80_0000) && (product[24] == 1))
                                then product[63:24] = product[63:24] + 1
                }
             /* sign-extend to 48 bits and combine with accumulator */
             /* check for the -1 * -1 overflow case */
     if ((operandY[31:0] == 0x8000_0000) && (operandX[31:0] == 0x8000_0000))
                then product[71:64] = 0x00                  /* zero-fill */
                else product[71:64] = {8{product[63]}}    /* sign-extend */
             if (inst == MSAC)
                then result[47:0] = ACCx[47:0] - product[71:24]
                else result[47:0] = ACCx[47:0] + product[71:24]
             /* check for accumulation overflow */
             if (accumulationOverflow == 1)
                then {MACSR.PAVn = 1
                      MACSR.V = 1
                      if (MACSR.OMC == 1)
                         then /* accumulation overflow,
                                 saturationMode enabled */
                              if (result[47] == 1)
                                 then result[47:0] = 0x007f_ffff_ff00
                                 else result[47:0] = 0xff80_0000_0000
                }
             /* transfer the result to the accumulator */
             ACCx[47:0] = result[47:0]
          }
       MACSR.V = MACSR.PAVn
       MACSR.N = ACCx[47]
       if (ACCx[47:0] == 0x0000_0000_0000)
           then MACSR.Z = 1
           else MACSR.Z = 0
       if ((ACCx[47:39] == 0x00_0) || (ACCx[47:39] == 0xff_1))
           then MACSR.EV = 0
           else MACSR.EV = 1
break;
case 2:              /* unsigned integers */
   if (MACSR.OMC == 0 || MACSR.PAVn == 0)
      then {
             MACSR.PAVn = 0
             /* select the input operands */
             if (sz == word)
                then {if (U/Ly == 1)
                         then operandY[31:0] = {0x0000, Ry[31:16]}
                         else operandY[31:0] = {0x0000, Ry[15:0]}
                      if (U/Lx == 1)
                         then operandX[31:0] = {0x0000, Rx[31:16]}
                         else operandX[31:0] = {0x0000, Rx[15:0]}
                }
                else {operandY[31:0] = Ry[31:0]
                      operandX[31:0] = Rx[31:0]
                }
             /* perform the multiply */
             product[63:0] = operandY[31:0] * operandX[31:0]
             /* check for product overflow */
```

```
                if (product[63:40] != 0x0000_00)
                   then {        /* product overflow */
                         MACSR.PAVn = 1
                         MACSR.V = 1
                         if (inst == MSAC && MACSR.OMC == 1)
                            then result[47:0] = 0x0000_0000_0000
                            else if (MACSR.OMC == 1)
                                    then /* overflowed MAC,
                                            saturationMode enabled */
                                         result[47:0] = 0xffff_ffff_ffff
                   }

                 /* zero-fill to 48 bits before performing any scaling */
                 product[47:40] = 0    /* zero-fill upper byte */
                /* scale product before combining with accumulator */
                switch (SF)     /* 2-bit scale factor */
                {
                    case 0:     /* no scaling specified */
                       break;
                    case 1:     /* SF = "<< 1" */
                       product[40:0] = {product[39:0], 0}
                       break;
                    case 2:     /* reserved encoding */
                       break;
                    case 3:     /* SF = ">> 1" */
                       product[39:0] = {0, product[39:1]}
                       break;
                }
                /* combine with accumulator */
                if (MACSR.PAVn == 0)
                   then {if (inst == MSAC)
                            then result[47:0] = ACCx[47:0] - product[47:0]
                            else result[47:0] = ACCx[47:0] + product[47:0]
                   }
                /* check for accumulation overflow */
                if (accumulationOverflow == 1)
                   then {MACSR.PAVn = 1
                         MACSR.V = 1
                         if (inst == MSAC && MACSR.OMC == 1)
                            then result[47:0] = 0x0000_0000_0000
                            else if (MACSR.OMC == 1)
                                 then /* overflowed MAC,
                                         saturationMode enabled */
                                      result[47:0] = 0xffff_ffff_ffff
                   }
                /* transfer the result to the accumulator */
                ACCx[47:0] = result[47:0]
          }
          MACSR.V = MACSR.PAVn
          MACSR.N = ACCx[47]
          if (ACCx[47:0] == 0x0000_0000_0000)
             then MACSR.Z = 1
             else MACSR.Z = 0
          if (ACCx[47:32] == 0x0000)
             then MACSR.EV = 0
             else MACSR.EV = 1
   break;
}
```

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

- Configuration for system clocking
  - Clock source selection for LPTMR, RTC, COP, and CLKOUT.
  - System clock divide values
- Architectural clock gating control
- Flash configuration
- RAM size configuration
- Flextimer external clock select and fault source
- Modulation and pad drive strength control of SCI2_TX
- 32 kHz OSC control selection

13.1.2 Modes of operation

- Run mode
- Wait mode
- Stop mode
- VLPR, VLPW, VLPS, and VLLSx modes

13.2 Memory Map and Registers

NOTE

Different SIM registers reset on different MCU reset types. The reset type represented by each register's displayed reset value is as follows:

Table 13-1. SIM Registers Reset Types

SIM Register	Reset Type
SOPT1	Chip Reset
SOPT2	Chip POR not VLLS
SOPT3	Chip Reset
SOPT4	Chip POR not VLLS
SOPT5	VLLDBGREG - Chip Reset not VLLS
SOPT6	Chip Reset
SOPT7	Chip Reset
SOPT8	Chip POR
SRVCOP	Chip Reset
OSC1	Chip Reset
SDIDH	Chip POR
SDIDL	Chip POR
SCGCx	Chip Reset
CLKOUT	Chip Reset
CLKDIV0	Chip POR
SPIWKUP	Chip Reset
FTMFAULT	Chip Reset
SPCR	Chip POR
UIDH[3:0], UIDMH[3:0], UIDML[3:0], UIDL[3:0]	Chip POR

NOTE

For each list, the registers are reset only by the specified reset type or any reset type that triggers the specified reset type. Other reset types do not affect the registers. For information about the various reset types on this chip, refer to the Reset details.

SIM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_80C0	System Options Register 1 (SIM_SOPT1)	8	R/W	0Ch	13.2.1/224
FFFF_80C1	System Options Register 2 (SIM_SOPT2)	8	R/W	See section	13.2.2/225
FFFF_80C2	System Options Register 3 (SIM_SOPT3)	8	R/W	02h	13.2.3/225
FFFF_80C3	System Options Register 4 (SIM_SOPT4)	8	R/W	74h	13.2.4/226
FFFF_80C4	System Options Register 5 (SIM_SOPT5)	8	R/W	See section	13.2.5/227
FFFF_80C5	System Options Register 6 (SIM_SOPT6)	8	R/W	00h	13.2.6/228
FFFF_80C6	System Options Register 7 (SIM_SOPT7)	8	R/W	00h	13.2.7/228
FFFF_80C7	System Options Register 8 (SIM_SOPT8)	8	R	Undefined	13.2.8/229
FFFF_80C8	COP Service Register (SIM_SRVCOP)	8	W (always reads 0)	00h	13.2.9/230
FFFF_80CB	Oscillator 1 Control Register (SIM_OSC1)	8	R/W	00h	13.2.10/231
FFFF_80CC	System Device Identification High Register (SIM_SDIDH)	8	R	0Eh	13.2.11/232
FFFF_80CD	System Device Identification Low Register (SIM_SDIDL)	8	R	02h	13.2.12/232
FFFF_80CE	System Clock Gate Control Register 1 (SIM_SCGC1)	8	R/W	00h	13.2.13/232
FFFF_80D0	System Clock Gate Control Register 3 (SIM_SCGC3)	8	R/W	00h	13.2.14/234
FFFF_80D1	System Clock Gate Control Register 4 (SIM_SCGC4)	8	R/W	81h	13.2.15/234
FFFF_80D2	System Clock Gate Control Register 5 (SIM_SCGC5)	8	R/W	01h	13.2.16/236
FFFF_80D3	System Clock Gate Control Register 6 (SIM_SCGC6)	8	R/W	00h	13.2.17/237
FFFF_80D4	Flash Configuration Register 1 (SIM_FCFG1)	8	R/W	Undefined	13.2.18/238
FFFF_80D5	Flash Configuration Register 2 (SIM_FCFG2)	8	R/W	See section	13.2.19/238
FFFF_80D7	Clockout Register (SIM_CLKOUT)	8	R/W	07h	13.2.20/239
FFFF_80D8	Clock Divider 0 Register (SIM_CLKDIV0)	8	R/W	See section	13.2.21/239
FFFF_80DA	SPI Wakeup Control Register (SIM_SPIWKUP)	8	R/W	00h	13.2.22/240
FFFF_80DC	FTM Fault Configuration Register (SIM_FTM_FAULT)	8	R/W	00h	13.2.23/241
FFFF_80DE	Flash Configuration Register (SIM_SPCR)	8	R	39h	13.2.24/242
FFFF_80E0	Unique Identification Register (SIM_UIDH3)	8	R	Undefined	13.2.25/243
FFFF_80E1	Unique Identification Register (SIM_UIDH2)	8	R	Undefined	13.2.26/243
FFFF_80E2	Unique Identification Register (SIM_UIDH1)	8	R	Undefined	13.2.27/244
FFFF_80E3	Unique Identification Register (SIM_UIDH0)	8	R	Undefined	13.2.28/244
FFFF_80E4	Unique Identification Register (SIM_UIDMH3)	8	R	Undefined	13.2.29/245
FFFF_80E5	Unique Identification Register (SIM_UIDMH2)	8	R	Undefined	13.2.30/245
FFFF_80E6	Unique Identification Register (SIM_UIDMH1)	8	R	Undefined	13.2.31/246
FFFF_80E7	Unique Identification Register (SIM_UIDMH0)	8	R	Undefined	13.2.32/246
FFFF_80E8	Unique Identification Register (SIM_UIDML3)	8	R	Undefined	13.2.33/247
FFFF_80E9	Unique Identification Register (SIM_UIDML2)	8	R	Undefined	13.2.34/247
FFFF_80EA	Unique Identification Register (SIM_UIDML1)	8	R	Undefined	13.2.35/248
FFFF_80EB	Unique Identification Register (SIM_UIDML0)	8	R	Undefined	13.2.36/248

Table continues on the next page...

SIM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_80EC	Unique Identification Register (SIM_UIDL3)	8	R	Undefined	13.2.37/249
FFFF_80ED	Unique Identification Register (SIM_UIDL2)	8	R	Undefined	13.2.38/249
FFFF_80EE	Unique Identification Register (SIM_UIDL1)	8	R	Undefined	13.2.39/250
FFFF_80EF	Unique Identification Register (SIM_UIDL0)	8	R	Undefined	13.2.40/250

13.2.1 System Options Register 1 (SIM_SOPT1)

Address: FFFF_80C0h base + 0h offset = FFFF_80C0h

Bit	7	6	5	4	3	2	1	0
Read	0				COPT		COPCLKS	COPW
Write								
Reset	0	0	0	0	1	1	0	0

SIM_SOPT1 field descriptions

Field	Description
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–2 COPT	COP watchdog timeout These write-once bits select the timeout period of the COP. COPT along with SOPT1[COPCLKS] defines the COP timeout period. Refer to COP Watchdog for more details.
1 COPCLKS	COP watchdog clock select This write-once bit selects the clock source of the COP watchdog. 0 Internal 1 kHz clock is a source to COP. (reset value). 1 Bus clock is a source to COP.
0 COPW	COP window mode This write-once bit specifies whether the COP operates in Normal or Window mode. In Window mode, the 0x55–0xAA write sequence to the SRS register must occur within the last 25% of the selected period. Any write to the SRS register during the first 75% of the selected period resets the MCU. 0 Normal mode (reset value). 1 Window mode.

13.2.2 System Options Register 2 (SIM_SOPT2)

Address: FFFF_80C0h base + 1h offset = FFFF_80C1h

Bit	7	6	5	4	3	2	1	0
Read	1	0	Reserved				OSC1ISO	0
Write								
Reset	1	0	*	*	*	*	0	0

* Notes:
- Reserved field: The value of each bit of this reserved field can be 0 or 1.

SIM_SOPT2 field descriptions

Field	Description
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.
5–2 Reserved	This field is reserved. The value is set by IFR bits.
1 OSC1ISO	Oscillator Isolation This bit indicates if OSC1 is in latched state and ignores any change of input digital control signals. This bit only gets reset in POR/LVD and holds its vaue in VLLS mode. OSC1 will be in latched state during VLLS mode irrespective of the state of this bit. **NOTE:** This bit resets on Chip POR not VLLS. 0 OSC1 is in normal state and any change of input control signal will come into effect. 1 All input controls of OSC1 analog blocks are latched. Any change to the control will have no effect on OSC1.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

13.2.3 System Options Register 3 (SIM_SOPT3)

Address: FFFF_80C0h base + 2h offset = FFFF_80C2h

Bit	7	6	5	4	3	2	1	0
Read	0		0		MECS		F1ECS	F0ECS
Write								
Reset	0	0	0	0	0	0	1	0

SIM_SOPT3 field descriptions

Field	Description
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIM_SOPT3 field descriptions (continued)

Field	Description
5–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–2 MECS	MTIM16 external clock source select 00 TMR_CLKIN0 (reset value) 01 Reserved 10 FTM0 Ch1 Out 11 FTM1 Ch1 Out
1 F1ECS	FTM1 external clock select 0 TMR_CLKIN0 1 Reserved
0 F0ECS	FTM0 external clock select 0 TMR_CLKIN0 (reset value) 1 Reserved

13.2.4 System Options Register 4 (SIM_SOPT4)

Address: FFFF_80C0h base + 3h offset = FFFF_80C3h

Bit	7	6	5	4	3	2	1	0
Read Write	RAMSIZE				STOPE	WAITE	LPT0CS	RTCCS
Reset	0	1	1	1	0	1	0	0

SIM_SOPT4 field descriptions

Field	Description
7–4 RAMSIZE	Size of RAM array. Specifies the amount of system RAM available on the device. 0000 Reserved 0001 Reserved 0010 Reserved 0011 Reserved 0100 Reserved 0101 Reserved 0110 Reserved 0111 64KB Other Reserved
3 STOPE	Stop mode enable

Table continues on the next page...

SIM_SOPT4 field descriptions (continued)

Field	Description
	This write-once bit is used to enable Stop mode. If both Stop and Wait modes are disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset may be generated depending on CPUCR[IRD].
2 WAITE	Wait mode enable This write-anytime bit is used to enable Wait mode. If both Stop and Wait modes are disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset may be generated depending on CPUCR[IRD].
1 LPT0CS	LPT0 external clock source select 0 OSC1_32KCLK 1 OSC2_32KCLK
0 RTCCS	RTC clock source select 0 OSC1_32KCLK 1 MCGIRCLK

13.2.5 System Options Register 5 (SIM_SOPT5)

Address: FFFF_80C0h base + 4h offset = FFFF_80C4h

Bit	7	6	5	4	3	2	1	0
Read	0				Reserved			VLLDBGREQ
Write								
Reset	0	0	0	0	*	*	*	0

* Notes:
- Reserved field: The value of each bit of this reserved field can be 0 or 1.

SIM_SOPT5 field descriptions

Field	Description
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–1 Reserved	This field is reserved. The value is set by IFR bits.
0 VLLDBGREQ	VLLS debug request upon wakeup VLLDBGREQ is writable only via BDM commands. 0 Wakeup from VLLS is via reset (reset value). 1 Wakeup from VLLS is via reset to active background mode.

13.2.6 System Options Register 6 (SIM_SOPT6)

Address: FFFF_80C0h base + 5h offset = FFFF_80C5h

Bit	7	6	5	4	3	2	1	0
Read	0		MTBASE1		0	MODTX2	PTB4PAD	0
Write								
Reset	0	0	0	0	0	0	0	0

SIM_SOPT6 field descriptions

Field	Description
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–4 MTBASE1	SCI2 TX modulation time base select 00 FTM0 Ch0 (reset value) 01 FTM0 Ch1 10 FTM1 Ch0 11 MTIM
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 MODTX2	Modulate TX2 0 Do not modulate the output of SCI2 (reset value). 1 Modulate the output of SCI2 with the timebase selected via the SOPT6[MTBASE1].
1 PTB4PAD	PTB4 SCI2 Tx double pad strength 0 Standard drive strength (reset value) 1 High drive strength
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

13.2.7 System Options Register 7 (SIM_SOPT7)

Address: FFFF_80C0h base + 6h offset = FFFF_80C6h

Bit	7	6	5	4	3	2	1	0
Read	0	0			IICDR0	0	FTM1SYNC	FTM0SYNC
Write								
Reset	0	0	0	0	0	0	0	0

SIM_SOPT7 field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIM_SOPT7 field descriptions (continued)

Field	Description
6–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 IICDR0	IIC link for IIC0 and IIC1 0 IIC0 and IIC1 operate in their respective pins (reset value). 1 IIC0 and IIC1 have their pins connected internally.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 FTM1SYNC	FTM1 synchronization trigger 0 No trigger generated. (reset value) 1 Generate a PWM synchronization trigger to the FTM1 module.
0 FTM0SYNC	FTM0 synchronization trigger 0 No trigger generated. (reset value) 1 Generate a PWM synchronization trigger to the FTM0 module.

13.2.8 System Options Register 8 (SIM_SOPT8)

Address: FFFF_80C0h base + 7h offset = FFFF_80C7h

Bit	7	6	5	4	3	2	1	0
Read	EEESIZE				DEPART			
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

SIM_SOPT8 field descriptions

Field	Description
7–4 EEESIZE	EEE size EEE data size as set by IFR bits. 0000 Reserved 0001 Reserved 0010 Reserved 0011 2 KB 0100 1 KB 0101 512 bytes 0110 256 bytes 0111 128 Bytes 1000 64 bytes 1001 32 bytes

Table continues on the next page...

SIM_SOPT8 field descriptions (continued)

Field	Description
	1010 0 bytes 1011 0 bytes 1100 0 bytes 1101 0 bytes 1110 0 bytes 1111 0 bytes
3–0 DEPART	D-Flash/E-Flash split Value is set by IFR bits. 0000 When Flexmemory=32 KB, D-Flash size is 32 KB and E-Flash size is 0 KB. When Flexmemory=16 KB, D-Flash size is 16 KB and E-Flash size is 0 KB. 0001 When Flexmemory=32 KB, D-Flash size is 24 KB and E-Flash size is 8 KB. When Flexmemory=16 KB, D-Flash size is 8 KB and E-Flash size is 8 KB. 0010 When Flexmemory=32 KB, D-Flash size is 16 KB and E-Flash size is 16 KB. When Flexmemory=16 KB, D-Flash size is 0 KB and E-Flash size is 16 KB. 0011 When Flexmemory=32 KB, D-Flash size is 0 KB and E-Flash size is 32 KB. When Flexmemory=16 KB, reserved. 0100 When Flexmemory=32 KB, reserved. When Flexmemory=16 KB, reserved. 0101 When Flexmemory=32 KB, reserved. When Flexmemory=16 KB, reserved. 0110 When Flexmemory=32 KB, reserved. When Flexmemory=16 KB, reserved. 0111 When Flexmemory=32 KB, reserved. When Flexmemory=16 KB, reserved. 1000 When Flexmemory=32 KB, D-Flash size 0 KB and E-Flash size is 32 KB. When Flexmemory=16 KB, D-Flash size is 0 KB and E-Flash size is 16 KB. 1001 When Flexmemory=32 KB, D-Flash size is 8 KB and E-Flash size is 24 KB. When Flexmemory=16 KB, D-Flash size is 8 KB and E-Flash size is 8 KB. 1010 When Flexmemory=32 KB, D-Flash size is 16 KB and E-Flash size is 16 KB. When Flexmemory=16 KB, D-Flash size is 16 KB and E-Flash size is 0 KB. 1011 When Flexmemory=32 KB, D-Flash size is 32 KB and E-Flash size is 0 KB. When Flexmemory=16 KB, reserved. 1100 When Flexmemory=32 KB, reserved. When Flexmemory=16 KB, reserved. 1101 When Flexmemory=32 KB, reserved. When Flexmemory=16 KB, reserved. 1110 When Flexmemory=32 KB, reserved. When Flexmemory=16 KB, reserved. 1111 When Flexmemory=32 KB, reserved (32/0). When Flexmemory=16 KB, reserved (16/0).

13.2.9 COP Service Register (SIM_SRVCOP)

Address: FFFF_80C0h base + 8h offset = FFFF_80C8h

Bit	7	6	5	4	3	2	1	0
Read	0							
Write	SRVCOP							
Reset	0	0	0	0	0	0	0	0

SIM_SRVCOP field descriptions

Field	Description
7–0 SRVCOP	Write 0x55 followed by 0xAA to reset the COP timeout counter. As soon as the write sequence is complete, the COP timeout period is restarted. If the program fails to perform this restart during the timeout period, the microcontroller resets. If any value other than 0x55 or 0xAA is written to the SRVCOP register, the microcontroller immediately resets.

13.2.10 Oscillator 1 Control Register (SIM_OSC1)

Controls OSC1. OSC2 is controlled by the MCG registers.

Address: FFFF_80C0h base + Bh offset = FFFF_80CBh

Bit	7	6	5	4	3	2	1	0
Read	OSC1EN	0		RANGE1		HG01	EREFS1	0
Write								
Reset	0	0	0	0	0	0	0	0

SIM_OSC1 field descriptions

Field	Description
7 OSC1EN	Oscillator 1 enable Enables oscillator 1. 0 Oscillator 1 inactive. 1 Oscillator 1 active.
6–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–3 RANGE1	Frequency range select Selects the frequency range for the crystal oscillator or external clock source. For details, refer to the OSC chapter. 00 Encoding 0: Low frequency range for the crystal oscillator of 32 kHz to 40 kHz (reset default). 01 Reserved 1x Reserved
2 HG01	High gain oscillator select Controls the crystal oscillator mode of operation. For details, refer to the OSC chapter. 0 Configure crystal oscillator for low-power operation. 1 Reserved
1 EREFS1	External reference select Selects the source for the external reference clock. Refer to the OSC chapter for details. 0 External reference clock requested. 1 Oscillator requested.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

13.2.11 System Device Identification High Register (SIM_SDIDH)

Address: FFFF_80C0h base + Ch offset = FFFF_80CCh

Bit	7	6	5	4	3	2	1	0
Read	REV				Reserved			
Write								
Reset	0	0	0	0	1	1	1	0

SIM_SDIDH field descriptions

Field	Description
7–4 REV	Device Revision Number Specifies the silicon implementation number for the device. This value comes from plugs, not IFR bits.
3–0 Reserved	This field is reserved.

13.2.12 System Device Identification Low Register (SIM_SDIDL)

Address: FFFF_80C0h base + Dh offset = FFFF_80CDh

Bit	7	6	5	4	3	2	1	0
Read	Reserved		ID[5:0]					
Write								
Reset	0	0	0	0	0	0	1	0

SIM_SDIDL field descriptions

Field	Description
7–6 Reserved	This field is reserved.
5–0 ID[5:0]	Device identification number Specifies the device identification number as set by the IFR bits. Its reset value is loaded from the IFR register.

13.2.13 System Clock Gate Control Register 1 (SIM_SCGC1)

Address: FFFF_80C0h base + Eh offset = FFFF_80CEh

Bit	7	6	5	4	3	2	1	0
Read Write	IIC1	IIC0	SPI2	SPI1	SPI0	SCI2	SCI1	SCI0
Reset	0	0	0	0	0	0	0	0

SIM_SCGC1 field descriptions

Field	Description
7 IIC1	IIC1 clock gate control Controls the clock gate to the IIC1 module. 0 The clock to the module is disabled 1 The clock to the module is enabled.
6 IIC0	IIC0 clock gate control Controls the clock gate to the IIC0 module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
5 SPI2	SPI2 clock gate control Controls the clock gate to the SPI2 module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
4 SPI1	SPI1 clock gate control Controls the clock gate to the SPI1 module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
3 SPI0	SPI0 clock gate control Controls the clock gate to the SPI0 module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
2 SCI2	SCI2 clock gate control Controls the clock gate to the SCI2 module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
1 SCI1	SCI1 clock gate control Controls the clock gate to the SCI1 module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
0 SCI0	SCI0 clock gate control Controls the clock gate to the SCI0 module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.

13.2.14 System Clock Gate Control Register 3 (SIM_SCGC3)

Address: FFFF_80C0h base + 10h offset = FFFF_80D0h

Bit	7	6	5	4	3	2	1	0
Read	CRC	0		MTIM	FTM1	FTM0	0	
Write								
Reset	0	0	0	0	0	0	0	0

SIM_SCGC3 field descriptions

Field	Description
7 CRC	CRC clock gate control Controls the clock gate to the CRC module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
6–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 MTIM	MTIM clock gate control Controls the clock gate to the MTIM module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
3 FTM1	FTM1 clock gate control Controls the clock gate to the FTM1 module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
2 FTM0	FTM0 clock gate control Controls the clock gate to the FTM0 module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
1–0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

13.2.15 System Clock Gate Control Register 4 (SIM_SCGC4)

Address: FFFF_80C0h base + 11h offset = FFFF_80D1h

Bit	7	6	5	4	3	2	1	0
Read	FTFL	DMA	0	IRQ3	IRQ2	IRQ1	IRQ0	COP
Write								
Reset	1	0	0	0	0	0	0	1

SIM_SCGC4 field descriptions

Field	Description
7 FTFL	FTFL clock gate control Controls the clock gate to the FTFL module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
6 DMA	DMA clock gate control Controls the clock gate to the DMA module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 IRQ3	IRQ3 clock gate control Controls the clock gate to the IRQ3 module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
3 IRQ2	IRQ2 clock gate control Controls the clock gate to the IRQ2 module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
2 IRQ1	IRQ1 clock gate control Controls the clock gate to the IRQ1 module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
1 IRQ0	IRQ0 clock gate control Controls the clock gate to the IRQ0 module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
0 COP	COP clock gate control Controls the clock gate to the COP module. **CAUTION:** If the COP is using the bus clock and is enabled and counting, and then the bus clock to the COP is disabled: • The COP cannot be serviced. • The COP will still time out and cause a reset. 0 The clock to the module is disabled. 1 The clock to the module is enabled.

13.2.16 System Clock Gate Control Register 5 (SIM_SCGC5)

Address: FFFF_80C0h base + 12h offset = FFFF_80D2h

Bit	7	6	5	4	3	2	1	0
Read	RNGA	0	0	0		OSC2	OSC1	MCG
Write								
Reset	0	0	0	0	0	0	0	1

SIM_SCGC5 field descriptions

Field	Description
7 RNGA	RNGA clock gate control Controls the clock gate to the RNGA module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
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.
4–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 OSC2	OSC2 clock gate control Controls the clock gate to the OSC2 module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
1 OSC1	OSC1 clock gate control Controls the clock gate to the OSC1 module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
0 MCG	MCG clock gate control Controls the clock gate to the MCG module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.

13.2.17 System Clock Gate Control Register 6 (SIM_SCGC6)

Address: FFFF_80C0h base + 13h offset = FFFF_80D3h

Bit	7	6	5	4	3	2	1	0
Read	0		0		PORTD	PORTC	PORTB	PORTA
Write								
Reset	0	0	0	0	0	0	0	0

SIM_SCGC6 field descriptions

Field	Description
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 PORTD	PORTD clock gate control Controls the clock gate to the PORTD module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.
2 PORTC	PORTC clock gate control Controls the clock gate to the PORTC module. 0 The clock to the module is disabled. 1 The clock to the module is enabled
1 PORTB	PORTB clock gate control Controls the clock gate to the PORTB module. 0 The clock to the module is disabled. 1 The clock to the module is enabled
0 PORTA	PORTA clock gate control Controls the clock gate to the PORTA module. 0 The clock to the module is disabled. 1 The clock to the module is enabled.

13.2.18 Flash Configuration Register 1 (SIM_FCFG1)

Address: FFFF_80C0h base + 14h offset = FFFF_80D4h

Bit	7	6	5	4	3	2	1	0
Read Write	MAXADDR1				MAXADDR0			
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

SIM_FCFG1 field descriptions

Field	Description
7–5 MAXADDR1	MSB of first invalid address of Array 1 (D-Flash). These bits are set by IFR.
4–0 MAXADDR0	MSB of first invalid address of Array 0 (D-Flash). These bits are set by IFR.

13.2.19 Flash Configuration Register 2 (SIM_FCFG2)

Address: FFFF_80C0h base + 15h offset = FFFF_80D5h

Bit	7	6	5	4
Read	PFLASH_SWAP	0		
Write				
Reset	*	0	0	0

Bit	3	2	1	0
Read	0			
Write				
Reset	0	0	0	0

* Notes:
- PFLASH_SWAP field: The value of each bit of this reserved field can be 0 or 1.

SIM_FCFG2 field descriptions

Field	Description
7 PFLASH_SWAP	Flash Swap Indicator. This indicates if the flash swap system has been initialized and thus potentially ready to be swapped.
6–0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

13.2.20 Clockout Register (SIM_CLKOUT)

Address: FFFF_80C0h base + 17h offset = FFFF_80D7h

Bit	7	6	5	4	3	2	1	0
Read	0	CS			0	CLKOUTDIV		
Write								
Reset	0	0	0	0	0	1	1	1

SIM_CLKOUT field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–4 CS	CLKOUT pin clock select 000 Disabled (reset value) 001 OSC1ERCLK 010 OSC2ERCLK 011 MCGOUT 100 SYSCLK 101 BUSCLK 110 LPOCLK 111 LPTMR0 prescaler clock output
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–0 CLKOUTDIV	Division of the CLKOUT pin 000 Division by 1 001 Division by 2 010 Division by 4 011 Division by 8 100 Division by 16 101 Division by 32 110 Division by 64 111 Division by 128 (reset value)

13.2.21 Clock Divider 0 Register (SIM_CLKDIV0)

Address: FFFF_80C0h base + 18h offset = FFFF_80D8h

Bit	7	6	5	4	3	2	1	0
Read	0				OUTDIV			
Write								
Reset	0	0	0	0	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

SIM_CLKDIV0 field descriptions

Field	Description
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–0 OUTDIV	MCG clock output divider value to generate CPU clock Sets the divide value for the core/system clock. At the end of reset, it is loaded with either 0000 or 1111 depending on flash option register bit 0 (FOPT[0]). **Restriction:** • This bitfield cannot be changed when the MCU is in a VLPx mode. • The CPU clock should never exceed 50 MHz. 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

13.2.22 SPI Wakeup Control Register (SIM_SPIWKUP)

Address: FFFF_80C0h base + 1Ah offset = FFFF_80DAh

Bit	7	6	5	4	3	2	1	0
Read	WIF_EN_SPI2	WIF_EN_SPI1	WIF_EN_SPI0	0		WIF_SPI2	WIF_SPI1	WIF_SPI0
Write								
Reset	0	0	0	0	0	0	0	0

SIM_SPIWKUP field descriptions

Field	Description
7 WIF_EN_SPI2	SPI2 wakeup interrupt enable When asserted, this field enables SPI2 to wakeup from low power modes (except VLLS mode) when there is any activity on SPI2_SS signal.
6 WIF_EN_SPI1	SPI1 wakeup interrupt enable When asserted, this field enables SPI1 to wakeup from low power modes (except VLLS mode) when there is any activity on SPI1_SS signal.

Table continues on the next page...

SIM_SPIWKUP field descriptions (continued)

Field	Description
5 WIF_EN_SPI0	SPI0 wakeup interrupt enable When asserted, this field enables SPI0 to wakeup from low power modes (except VLLS mode) when there is any activity on SPI0_SS signal.
4–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 WIF_SPI2	SPI2 wakeup interrupt flag This flag is set during a SPI2 wakeup event.
1 WIF_SPI1	SPI1 wakeup interrupt flag This flag is set during a SPI1 wakeup event.
0 WIF_SPI0	SPI0 wakeup interrupt flag This flag is set during a SPI0 wakeup event.

13.2.23 FTM Fault Configuration Register (SIM_FTM_FAULT)

Address: FFFF_80C0h base + 1Ch offset = FFFF_80DCh

Bit	7	6	5	4	3	2	1	0
Read	0				FTM_FLT3	FTM_FLT2	FTM_FLT1	FTM_FLT0
Write								
Reset	0	0	0	0	0	0	0	0

SIM_FTM_FAULT field descriptions

Field	Description
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 FTM_FLT3	FTM fault input 3 This field sets the fault input (fault3) to both FTM0 and FTM1.
2 FTM_FLT2	FTM fault input 2 This field sets the fault input (fault2) to both FTM0 and FTM1.
1 FTM_FLT1	FTM fault input 1 This field sets the fault input (fault1) to both FTM0 and FTM1.
0 FTM_FLT0	FTM fault input 0 This field sets the fault input (fault0) to both FTM0 and FTM1.

13.2.24 Flash Configuration Register (SIM_SPCR)

Address: FFFF_80C0h base + 1Eh offset = FFFF_80DEh

Bit	7	6	5	4	3	2	1	0
Read	FlexNVM				PFlash			
Write								
Reset	0	0	1	1	1	0	0	1

SIM_SPCR field descriptions

Field	Description
7–4 FlexNVM	Data flash configuration Specifies the amount of program flash memory available on the device. It is set by IFR bits. 00h Reserved 01h Reserved 02h Reserved 03h 32 KB 04h Reserved 05h Reserved 06h Reserved 07h Reserved 08h Reserved Other Reserved
3–0 PFlash	Program flash configuration Specifies the amount of program flash memory available on the device. It is set by IFR bits. 00h Reserved 01h Reserved 02h Reserved 03h Reserved 04h Reserved 05h Reserved 06h Reserved 07h Reserved 08h 256 KB Other Reserved

13.2.25 Unique Identification Register (SIM_UIDH3)

Address: FFFF_80C0h base + 20h offset = FFFF_80E0h

Bit	7	6	5	4	3	2	1	0
Read	UID[127:120]							
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

SIM_UIDH3 field descriptions

Field	Description
7–0 UID[127:120]	Unique identification Provides unique identification for the device. These bits are set by the IFR bits.

13.2.26 Unique Identification Register (SIM_UIDH2)

Address: FFFF_80C0h base + 21h offset = FFFF_80E1h

Bit	7	6	5	4	3	2	1	0
Read	UID[119:112]							
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

SIM_UIDH2 field descriptions

Field	Description
7–0 UID[119:112]	Unique identification Provides unique identification for the device. These bits are set by the IFR bits.

13.2.27 Unique Identification Register (SIM_UIDH1)

Address: FFFF_80C0h base + 22h offset = FFFF_80E2h

Bit	7	6	5	4	3	2	1	0
Read	UID[111:104]							
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

SIM_UIDH1 field descriptions

Field	Description
7–0 UID[111:104]	Unique identification Provides unique identification for the device. These bits are set by the IFR bits.

13.2.28 Unique Identification Register (SIM_UIDH0)

Address: FFFF_80C0h base + 23h offset = FFFF_80E3h

Bit	7	6	5	4	3	2	1	0
Read	UID[103:96]							
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

SIM_UIDH0 field descriptions

Field	Description
7–0 UID[103:96]	Unique identification Provides unique identification for the device. These bits are set by the IFR bits.

13.2.29 Unique Identification Register (SIM_UIDMH3)

Address: FFFF_80C0h base + 24h offset = FFFF_80E4h

Bit	7	6	5	4	3	2	1	0
Read	UID[95:88]							
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

SIM_UIDMH3 field descriptions

Field	Description
7–0 UID[95:88]	Unique identification Provides unique identification for the device. These bits are set by the IFR bits.

13.2.30 Unique Identification Register (SIM_UIDMH2)

Address: FFFF_80C0h base + 25h offset = FFFF_80E5h

Bit	7	6	5	4	3	2	1	0
Read	UID[87:80]							
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

SIM_UIDMH2 field descriptions

Field	Description
7–0 UID[87:80]	Unique identification Provides unique identification for the device. These bits are set by the IFR bits.

13.2.31 Unique Identification Register (SIM_UIDMH1)

Address: FFFF_80C0h base + 26h offset = FFFF_80E6h

Bit	7	6	5	4	3	2	1	0
Read	UID[79:72]							
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

SIM_UIDMH1 field descriptions

Field	Description
7–0 UID[79:72]	Unique identification Provides unique identification for the device. These bits are set by the IFR bits.

13.2.32 Unique Identification Register (SIM_UIDMH0)

Address: FFFF_80C0h base + 27h offset = FFFF_80E7h

Bit	7	6	5	4	3	2	1	0
Read	UID[71:64]							
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

SIM_UIDMH0 field descriptions

Field	Description
7–0 UID[71:64]	Unique identification Provides unique identification for the device. These bits are set by the IFR bits.

13.2.33 Unique Identification Register (SIM_UIDML3)

Address: FFFF_80C0h base + 28h offset = FFFF_80E8h

Bit	7	6	5	4	3	2	1	0
Read	UID[63:56]							
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

SIM_UIDML3 field descriptions

Field	Description
7–0 UID[63:56]	Unique identification Provides unique identification for the device. These bits are set by the IFR bits.

13.2.34 Unique Identification Register (SIM_UIDML2)

Address: FFFF_80C0h base + 29h offset = FFFF_80E9h

Bit	7	6	5	4	3	2	1	0
Read	UID[55:48]							
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

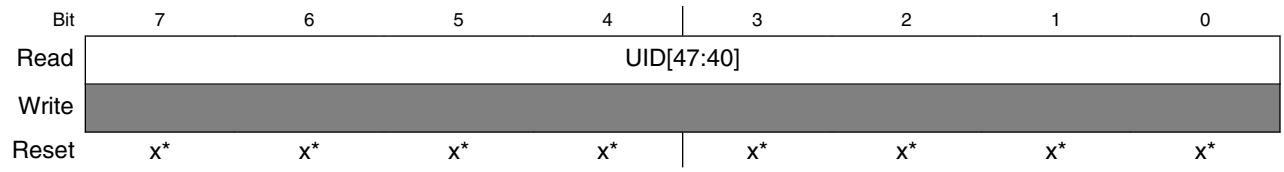
* Notes:
- x = Undefined at reset.

SIM_UIDML2 field descriptions

Field	Description
7–0 UID[55:48]	Unique identification Provides unique identification for the device. These bits are set by the IFR bits.

13.2.35 Unique Identification Register (SIM_UIDML1)

Address: FFFF_80C0h base + 2Ah offset = FFFF_80EAh



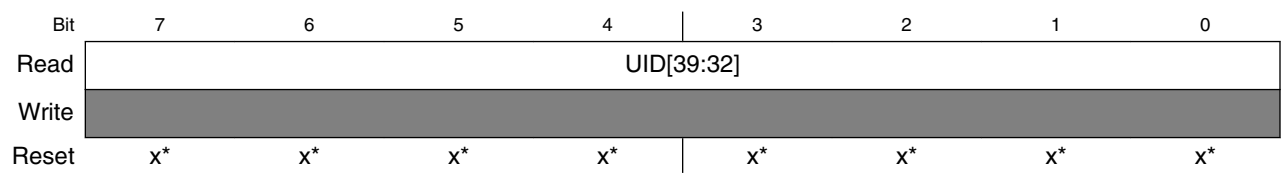
* Notes:
- x = Undefined at reset.

SIM_UIDML1 field descriptions

Field	Description
7–0 UID[47:40]	Unique identification Provides unique identification for the device. These bits are set by the IFR bits.

13.2.36 Unique Identification Register (SIM_UIDML0)

Address: FFFF_80C0h base + 2Bh offset = FFFF_80EBh



* Notes:
- x = Undefined at reset.

SIM_UIDML0 field descriptions

Field	Description
7–0 UID[39:32]	Unique identification Provides unique identification for the device. These bits are set by the IFR bits.

13.2.37 Unique Identification Register (SIM_UIDL3)

Address: FFFF_80C0h base + 2Ch offset = FFFF_80ECh

Bit	7	6	5	4	3	2	1	0
Read	UID[31:24]							
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

SIM_UIDL3 field descriptions

Field	Description
7–0 UID[31:24]	Unique identification Provides unique identification for the device. These bits are set by the IFR bits.

13.2.38 Unique Identification Register (SIM_UIDL2)

Address: FFFF_80C0h base + 2Dh offset = FFFF_80EDh

Bit	7	6	5	4	3	2	1	0
Read	UID[23:16]							
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

SIM_UIDL2 field descriptions

Field	Description
7–0 UID[23:16]	Unique identification Provides unique identification for the device. These bits are set by the IFR bits.

13.2.39 Unique Identification Register (SIM_UIDL1)

Address: FFFF_80C0h base + 2Eh offset = FFFF_80EEh

Bit	7	6	5	4	3	2	1	0
Read	UID[15:8]							
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

SIM_UIDL1 field descriptions

Field	Description
7–0 UID[15:8]	Unique identification Provides unique identification for the device. These bits are set by the IFR bits.

13.2.40 Unique Identification Register (SIM_UIDL0)

Address: FFFF_80C0h base + 2Fh offset = FFFF_80EFh

Bit	7	6	5	4	3	2	1	0
Read	UID[7:0]							
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

SIM_UIDL0 field descriptions

Field	Description
7–0 UID[7:0]	Unique identification Provides unique identification for the device. These bits are set by the IFR bits.

Chapter 14
Crossbar Switch

14.1 Introduction

This chapter provides information on the layout, configuration, and programming of the crossbar switch. The crossbar switch connects bus masters and bus slaves using a crossbar switch structure. This structure allows up to four bus masters to access different bus slaves simultaneously, while providing arbitration among the bus masters when they access the same slave.

14.1.1 Features

The crossbar switch includes these features:

- Symmetric crossbar bus switch implementation
  - Allows concurrent accesses from different masters to different slaves
- 32-bit data bus
- Operation at a 1-to-1 clock frequency with the bus masters
- Dynamic master priority elevation
- Programmable configuration for fixed-priority or round-robin slave port arbitration

14.2 Memory Map / Register Definition

This crossbar switch is designed for minimal gate count. It, therefore, has no memory-mapped configuration registers.

14.3 Functional Description

14.3.1 General operation

When a master accesses the crossbar switch the access is immediately taken. If the targeted slave port of the access is available, then the access is immediately presented on the slave port. Single-clock or zero-wait-state accesses are possible through the crossbar. If the targeted slave port of the access is busy or parked on a different master port, the requesting master simply sees wait states inserted until the targeted slave port can service the master's request. The latency in servicing the request depends on each master's priority level and the responding slave's access time.

Because the crossbar switch appears to be just another slave to the master device, the master device has no knowledge of whether it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting, it simply waits.

A master is given control of the targeted slave port only after a previous access to a different slave port completes, regardless of its priority on the newly targeted slave port. This prevents deadlock from occurring when:

- A higher priority master has:
  - An outstanding request to one slave port that has a long response time and
  - A pending access to a different slave port, and
- A lower priority master is also making a request to the same slave port as the pending access of the higher priority master.

After the master has control of the slave port it is targeting, the master remains in control of the slave port until it relinquishes the slave port by running an IDLE cycle or by targeting a different slave port for its next access.

The master can also lose control of the slave port if another higher-priority master makes a request to the slave port.

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 master had to arbitrate to gain control of the slave port.

14.3.2 Arbitration

The crossbar switch supports two arbitration algorithms:

- Fixed priority
- Round robin

The selection of the global slave port arbitration is controlled by MCM_PLACR[ARB]. For fixed priority, set ARB to 0. For round robin, set ARB to 1. This arbitration setting applies to all slave ports.

The crossbar arbitration scheme is controlled by CPUCR[CBRR] bit in the processor's CPU Configuration Register (CPUCR). See CPU Configuration Register (CPUCR) for details. At reset, fixed-priority arbitration is enabled.

14.3.2.1 Arbitration During Undefined Length Bursts

All lengths of burst accesses lock out arbitration until the last beat of the burst.

14.3.2.2 Fixed-priority operation

When operating in fixed-priority mode, each master is assigned a unique priority level with the highest numbered master having the highest priority (master 1 has lower priority than master 3). If two masters request access to the same slave port, the master with the highest priority gains control over the slave port.

NOTE

In this arbitration mode, a higher-priority master can monopolize a slave port, preventing accesses from any lower-priority master to the port.

When a master makes a request to a slave port, the slave port checks whether the new requesting master's priority level is higher than that of the master that currently has control over the slave port, unless the slave port is in a parked state. The slave port performs an arbitration check at every clock edge to ensure that the proper master, if any, has control of the slave port.

The following table describes possible scenarios based on the requesting master port:

Table 14-1. How AXBS grants control of a slave port to a master

When	Then AXBS grants control to the requesting master
Both of the following are true: • The current master is not running a transfer. • The new requesting master's priority level is higher than that of the current master.	At the next clock edge
The requesting master's priority level is lower than the current master.	At the conclusion of one of the following cycles: • An IDLE cycle • A non-IDLE cycle to a location other than the current slave port

14.3.2.3 Round-robin priority operation

When operating in round-robin mode, each master is assigned a relative priority based on the master port number. This relative priority is compared to the master port number (ID) of the last master to perform a transfer on the slave bus. The highest priority requesting master becomes owner of the slave bus at the next transfer boundary. Priority is based on how far ahead the ID of the requesting master is to the ID of the last master.

After granted access to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line is granted access to the slave port at the next transfer boundary, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the crossbar is implemented with master ports 0, 1, 4, and 5. If the last master of the slave port was master 1, and master 0, 4, and 5 make simultaneous requests, they are serviced in the order: 4 then 5 then 0.

The round-robin arbitration mode generally provides a more fair allocation of the available slave-port bandwidth (compared to fixed priority) as the fixed master priority does not affect the master selection.

14.3.2.4 Priority Elevation

The processor master port may elevate it's priority level to the slave ports. This is controlled by CPUCR[HAE] and CPUCR[FHP] bits. See CPU Configuration Register (CPUCR) for details. The CPUCR[FHP] bit always elevates the processor's priority while the CPUCR[HAE] bit elevates the processor's priority while it is processing any interrupt service routine.

When priority elevation is enabled for the processor's master port, that master port has a higher priority than the other master ports that do not, regardless of the fixed priority level or round robin conditions.

This functionality allows the user to automatically elevate a master port's priority level throughout the crossbar to quickly perform temporary tasks such as servicing interrupts. However, when using this functionality take care to not to occupy 100% of a slave's bandwidth with one master, and essentially locking out the rest of the masters. This should only be used for important temporary tasks, such as interrupt handling.

14.3.2.4.1 Priority Elevation in Round-Robin Mode

If a slave port is programmed for round-robin mode and a master is configured for priority elevation, that master can force the slave port into fixed priority mode. The slave port remains in fixed priority mode while that master's (or any other master's) configured for priority elevation and it attempts access to that particular slave port.

After the masters are no longer configured for priority elevation, or the masters no longer attempt accesses to that particular slave port, the slave port reverts to round-robin priority mode. Then, the pointer is set on the last master to access the slave port.

14.4 Initialization/application information

No initialization is required for the crossbar switch. See the AXBS section of the configuration chapter for the reset state of the arbitration scheme.

Chapter 15
Interrupt Controller (INTC)

15.1 Introduction

The CF1_INTC interrupt controller (CF1_INTC) is intended for use in low-cost microcontroller designs using the Version 1 (V1) ColdFire processor core. In keeping with the general philosophy for devices based on this low-end 32-bit processor, the interrupt controller generally supports less programmability compared to similar modules in other ColdFire microcontrollers and embedded microprocessors. However, CF1_INTC provides the required functionality with a minimal silicon cost.

These requirements guide the CF1_INTC module definition to support Freescale's Controller Continuum:

- The priorities of the interrupt requests between comparable HCS08 and V1 ColdFire devices are identical.
- Supports a mode of operation (through software convention with hardware assists) equivalent to the S08's interrupt processing with only one level of nesting.
- Leverages the current ColdFire interrupt controller programming model and functionality, but with a minimal hardware implementation and cost.

The following table provides a high-level architectural comparison between HCS08 and ColdFire exception processing as these differences are important in the definition of the CF1_INTC module. Throughout this document, the term IRQ refers to an interrupt request and ISR refers to an interrupt service routine to process an interrupt exception.

Table 15-1. Exception Processing Comparison

Attribute	HCS08	V1 ColdFire
Exception Vector Table	32 two-byte entries, fixed location at upper end of memory	115 four-byte entries, located at lower end of memory at reset, relocatable with the VBR

Table continues on the next page...

Table 15-1. Exception Processing Comparison (continued)

Attribute	HCS08	V1 ColdFire
More on Vectors	2 for CPU + 30 for IRQs, reset at upper address	64 for CPU + 39 for IRQs, reset at lowest address
Exception Stack Frame	5-byte frame: CCR, A, X, PC	8-byte frame: F/V, SR, PC; General-purpose registers (An, Dn) must be saved/ restored by the ISR
Interrupt Levels	1 = *f*(CCR[I])	7= *f* (SR[I]) with automatic hardware support for nesting
Non-Maskable IRQ Support	No	Yes, with level 7 interrupts
Core-enforced IRQ Sensitivity	No	Level 7 is edge sensitive, else level sensitive
INTC Vectoring	Fixed priorities and vector assignments	Fixed priorities and vector assignments, plus any 2 IRQs can be remapped as the highest priority level 6 requests
Software IACK	No	Yes
Exit Instruction from ISR	RTI	RTE

15.1.1 Overview

Interrupt exception processing includes interrupt recognition, aborting the current instruction execution stream, storing an 8-byte exception stack frame in the memory, calculation of the appropriate vector, and passing control to the specified interrupt service routine.

Unless specifically noted otherwise, all ColdFire processors sample for interrupts once during each instruction's execution during the first cycle of execution in the OEP. Additionally, all ColdFire processors use an instruction restart exception model.

The ColdFire processor architecture defines a 3-bit interrupt priority mask field in the processor's status register (SR[I]). This field, and the associated hardware, support seven levels of interrupt requests with the processor providing automatic nesting capabilities. The levels are defined in descending numeric order with 7 > 6 ... > 1. Level 7 interrupts are treated as non-maskable, edge-sensitive requests while levels 6–1 are maskable, level-sensitive requests. The SR[I] field defines the processor's current interrupt level. The processor continuously compares the encoded IRQ level from CF1_INTC against SR[I]. Recall that interrupt requests are inhibited for all levels less than or equal to the current level, except the edge-sensitive level 7 request that cannot be masked.

Exception processing for ColdFire processors is streamlined for performance and includes all actions from detecting the fault condition to the initiation of fetch for the first handler instruction. Exception processing is comprised of four major steps.

1. The processor makes an internal copy of the status register (SR) and enters supervisor mode by setting SR[S] and disabling trace mode by clearing SR[T]. The occurrence of an interrupt exception also forces the master mode (M) bit to clear and the interrupt priority mask (I) to set to the level of the current interrupt request.
2. The processor determines the exception vector number. For all faults except interrupts, the processor performs this calculation based on the exception type. For interrupts, the processor performs an IACK bus cycle to obtain the vector number from the interrupt controller if CPUCR[IAE] equals 1. The IACK cycle is mapped to special locations within the interrupt controller's IPS address space with the interrupt level encoded in the address. If CPUCR[IAE] equals 0, the processor uses the vector number supplied by the interrupt controller at the time the request was signaled (for improved performance).
3. The processor saves the current context by creating an exception stack frame on the system stack. As a result, exception stack frame is created at a 0-modulo-4 address on top of the system stack defined by the supervisor stack pointer (SSP). The processor uses an 8-byte stack frame for all exceptions. It contains the vector number of the exception, the contents of the status register at the time of the exception, and the program counter (PC) at the time of the exception. The exception type determines whether the program counter placed in the exception stack frame defines the location of the faulting instruction (fault) or the address of the next instruction to be executed (next). For interrupts, the stacked PC is always the address of the next instruction to be executed.
4. The processor calculates the address of the first instruction of the exception handler. By definition, the exception vector table is aligned on a 1MB boundary. This instruction address is generated by fetching a 32-bit exception vector from the table located at the address defined in the vector base register (VBR). The index into the exception table is calculated as (4 × vector number). After the exception vector has been fetched, the contents of the vector serves as a 32-bit pointer to the address of the first instruction of the desired handler. After the instruction fetch for the first opcode of the handler has been initiated, exception processing terminates and normal instruction processing continues in the handler.

All ColdFire processors support a 1024-byte vector table aligned on any 1 MB address boundary. For the V1 ColdFire core, the only practical locations for the vector table are based at 0x(00)00_0000 in the flash or 0x(00)80_0000 in the RAM. The table contains 256 exception vectors; the first 64 are reserved for internal processor exceptions, and the remaining 192 are device-specific interrupt vectors. The IRQ assignment table is partially populated depending on the exact set of peripherals for the given device.

The basic ColdFire interrupt controller supports up to 63 request sources mapped as nine priorities for each of the seven supported levels (7 levels × 9 priorities per level). Within the nine priorities within a level, the mid-point is typically reserved for package-level IRQ inputs. The levels and priorities within the level follow a descending order: 7 > 6 > ... > 1 > 0.

The HCS08 architecture supports a 32-entry exception vector table: the first two vectors are reserved for internal CPU/system exceptions and the remaining are available for I/O interrupt requests. The requirement for an exact match between the interrupt requests and priorities across two architectures means the sources are mapped to a sparsely-populated two-dimensional ColdFire array of seven interrupt levels and nine priorities within the level. The following association between the HCS08 and ColdFire vector numbers applies:

```
ColdFire Vector Number = 62 + HCS08 Vector Number
```

The CF1_INTC performs a cycle-by-cycle evaluation of the active requests and signals the highest-level, highest-priority request to the V1 ColdFire core in the form of an encoded interrupt level and the exception vector associated with the request. The module also includes a byte-wide interface to access its programming model. These interfaces are shown in the following simplified block diagram.

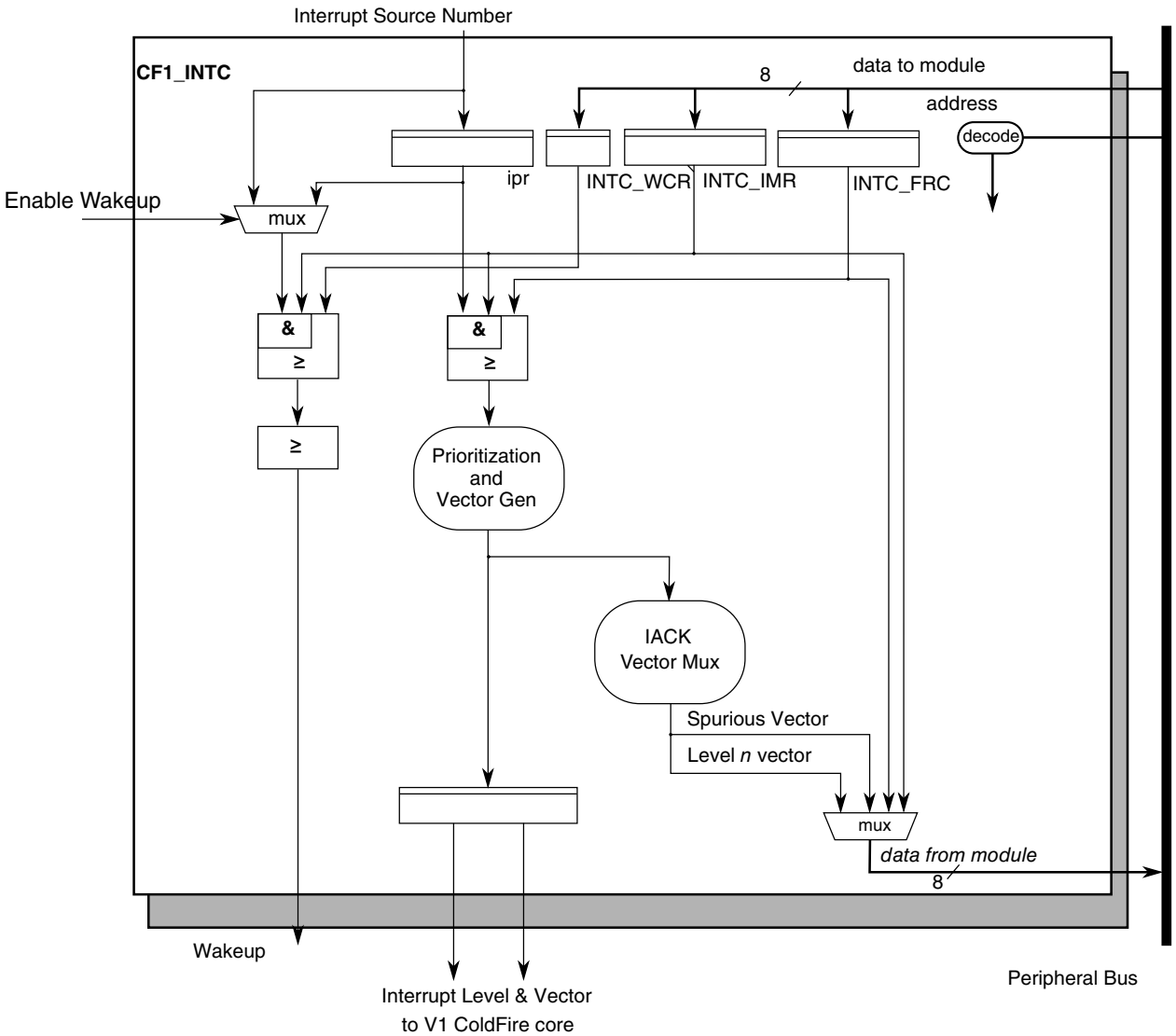


Figure 15-1. CF1_INTC Block Diagram

15.1.2 Features

The Version 1 ColdFire interrupt controller includes:

- Memory-mapped off-platform slave module
  - 64-byte space located at top end of memory: 0x(FF)FF_FFC0–0x(FF)FF_FFFF
  - Programming model accessed via the peripheral bus
  - Encoded interrupt level and vector sent directly to processor core
- Support of 30 peripheral I/O interrupt requests plus seven software (one per level) interrupt requests

- Fixed association between interrupt request source and level plus priority
  - 30 I/O requests assigned across seven available levels and nine priorities per level
  - Exactly matches HCS08 interrupt request priorities
  - Up to two requests can be remapped to the highest maskable level + priority
- Unique vector number for each interrupt source
  - ColdFire vector number = 62 + HCS08 vector number
  - Details on IRQ and vector assignments are device-specific
- Support for service routine interrupt acknowledge (software IACK) read cycles for improved system performance
- Combinatorial path provides wakeup signal from wait and stop modes
- Ability to mask any individual interrupt source, plus global mask-all capability

15.1.3 Modes of Operation

The CF1_INTC module does not support any special modes of operation. As a memory-mapped slave peripheral located on the platform's slave bus, it responds based strictly on the memory addresses of the connected bus.

One special behavior of the CF1_INTC deserves mention. When the device enters a wait or stop mode and certain clocks are disabled, there is an input signal that can be asserted to enable a purely-combinational logic path for monitoring the assertion of an interrupt request. After a request of unmasked level is asserted, this combinational logic path asserts an output signal that is sent to the clock generation logic to re-enable the internal device clocks to exit the low-power mode.

15.2 External Signal Description

The CF1_INTC module does not include any external interfaces.

15.3 Interrupt Request Level and Priority Assignments

The CF1_INTC module implements a sparsely populated 7 × 9 matrix of levels (7) and priorities within each level (9).

For details, see the chip-specific information about interrupt vector assignments.

Note

For remapped and forced interrupts, the interrupt source number entry indicates the register or register field that enables the corresponding interrupt.

15.4 Memory Map and Registers

The CF1_INTC module provides a 64-byte programming model mapped to the upper region of the 16 MB address space. All the register names are prefixed with INTC_ as an abbreviation for the full module name.

In the programming model, attempted references to undefined (reserved) addresses or with a non-supported access type (for example, a write to a read-only register) generate a bus error termination.

The programming model follows the definition from previous ColdFire interrupt controllers. This compatibility accounts for the various memory holes in this module memory map.

INTC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_FFC8	Interrupt Mask Register High (INTC_IMRH)	32	R/W	0000_0000h	15.4.1/264
FFFF_FFCC	Interrupt Mask Register Low (INTC_IMRL)	32	R/W	0000_0000h	15.4.2/266
FFFF_FFD0	Force Interrupt Register (INTC_FRC)	8	R/W	00h	15.4.3/270
FFFF_FFD8	INTC Programmable Level 6 Priority Registers (INTC_PL6P7)	8	R/W	00h	15.4.4/271
FFFF_FFD9	INTC Programmable Level 6 Priority Registers (INTC_PL6P6)	8	R/W	00h	15.4.4/271
FFFF_FFDB	INTC Wakeup Control Register (INTC_WCR)	8	R/W	80h	15.4.5/272
FFFF_FFDC	Set Interrupt Mask Register (INTC_SIMR)	8	W	00h	15.4.6/273
FFFF_FFDD	Clear Interrupt Mask Register (INTC_CIMR)	8	W	00h	15.4.7/274
FFFF_FFDE	INTC Set Interrupt Force Register (INTC_SFRC)	8	W	00h	15.4.8/275
FFFF_FFDF	INTC Clear Interrupt Force Register (INTC_CFRC)	8	W	00h	15.4.9/276
FFFF_FFE0	INTC Software IACK Register (INTC_SWIACK)	8	R	00h	15.4.10/277

Table continues on the next page...

INTC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_FFE4	INTC Level-n IACK Registers (INTC_LVL1IACK)	8	R	18h	15.4.11/277
FFFF_FFE8	INTC Level-n IACK Registers (INTC_LVL2IACK)	8	R	18h	15.4.11/277
FFFF_FFEC	INTC Level-n IACK Registers (INTC_LVL3IACK)	8	R	18h	15.4.11/277
FFFF_FFF0	INTC Level-n IACK Registers (INTC_LVL4IACK)	8	R	18h	15.4.11/277
FFFF_FFF4	INTC Level-n IACK Registers (INTC_LVL5IACK)	8	R	18h	15.4.11/277
FFFF_FFF8	INTC Level-n IACK Registers (INTC_LVL6IACK)	8	R	18h	15.4.11/277
FFFF_FFFC	INTC Level-n IACK Registers (INTC_LVL7IACK)	8	R	18h	15.4.11/277

15.4.1 Interrupt Mask Register High (INTC_IMRH)

INTC_IMRH, along with INTC_IMRL, provides a bit map for each interrupt to allow the request to be disabled (masked) (1 = disable the request, 0 = enable the request). The IMR is cleared by reset, enabling all interrupt requests to preserve compatibility with earlier V1 ColdFire devices. The IMR can be read and written directly, or individual mask flags can be set or cleared by accessing set/clear interrupt mask registers (INTC_SIMR, INTC_CIMR).

Each bit of the IMR[n] is associated with a corresponding bit of the interrupt request input vector. The equations defining this association are:

```
For Vectors 64-102, n=Vector_Number - 64, else for Vectors 110-114, n = Vector_Number - 71
```

Therefore, vector 64 corresponds to n = 0, vector 65 to n = 1, etc., vector 113 to n = 42, and vector 114 to n = 43.

Each peripheral request input is first qualified by the contents of the IMR registers before it is used elsewhere in the interrupt controller, that is:

```
qualified_interrupt_request[n] = interrupt_request_input[n] + ~INTC_IMR[n]
```

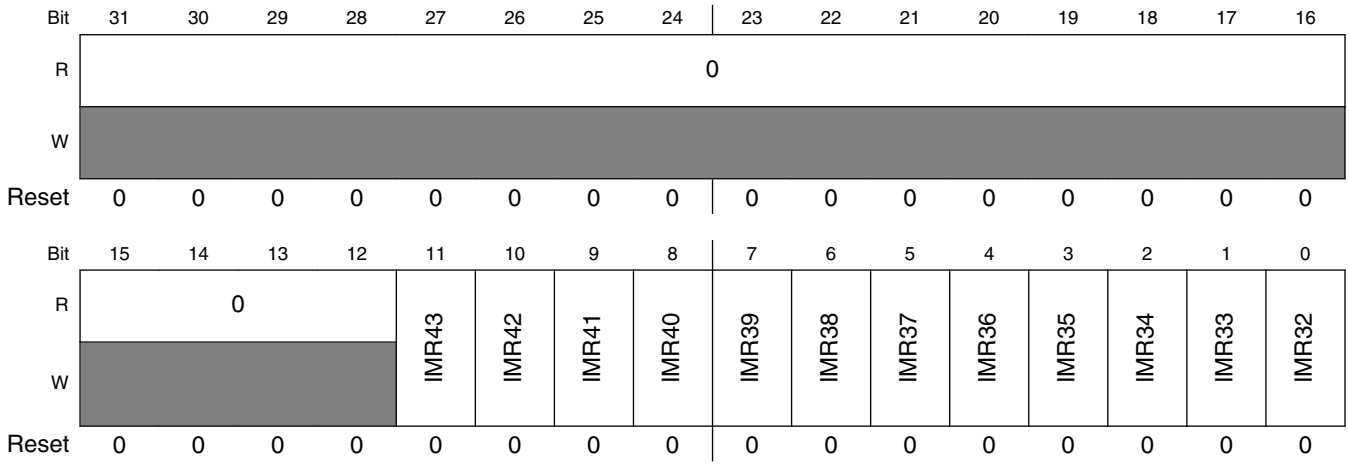
Since this interrupt controller supports 44 request inputs, the upper 20 bits of the INTC_IMRH are reserved for future use. Writes to these bits are ignored and reads return zeroes.

The contents of the IMR do not affect the operation of the software settable force interrupt registers.

NOTE

Because this register and the corresponding low register together represent 64 bits, this register's bits are actually numbered 63-32 (not 31-0).

Address: FFFF_FFC0h base + 8h offset = FFFF_FFC8h



INTC_IMRH field descriptions

Field	Description
31–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 IMR43	Interrupt mask register 43 0 The interrupt request is enabled. 1 The interrupt request is disabled.
10 IMR42	Interrupt mask register 42 0 The interrupt request is enabled. 1 The interrupt request is disabled.
9 IMR41	Interrupt mask register 41 0 The interrupt request is enabled. 1 The interrupt request is disabled.
8 IMR40	Interrupt mask register 40 0 The interrupt request is enabled. 1 The interrupt request is disabled.
7 IMR39	Interrupt mask register 39 0 The interrupt request is enabled. 1 The interrupt request is disabled.
6 IMR38	Interrupt mask register 38 0 The interrupt request is enabled. 1 The interrupt request is disabled.
5 IMR37	Interrupt mask register 37 0 The interrupt request is enabled. 1 The interrupt request is disabled.
4 IMR36	Interrupt mask register 36

Table continues on the next page...

INTC_IMRH field descriptions (continued)

Field	Description
	0 The interrupt request is enabled. 1 The interrupt request is disabled.
3 IMR35	Interrupt mask register 35 0 The interrupt request is enabled. 1 The interrupt request is disabled.
2 IMR34	Interrupt mask register 34 0 The interrupt request is enabled. 1 The interrupt request is disabled.
1 IMR33	Interrupt mask register 33 0 The interrupt request is enabled. 1 The interrupt request is disabled.
0 IMR32	Interrupt mask register 32 0 The interrupt request is enabled. 1 The interrupt request is disabled.

15.4.2 Interrupt Mask Register Low (INTC_IMRL)

INTC_IMRL, along with INTC_IMRH, provides a bit map for each interrupt to allow the request to be disabled (masked) (1 = disable the request, 0 = enable the request). The IMR is cleared by reset, enabling all interrupt requests to preserve compatibility with earlier V1 ColdFire devices. The IMR can be read and written directly, or individual mask flags can be set or cleared by accessing set/clear interrupt mask registers (INTC_SIMR, INTC_CIMR).

Each bit of the IMR[n] is associated with a corresponding bit of the interrupt request input vector. The equations defining this association are:

```
For Vectors 64-102, n=Vector_Number - 64, else for Vectors 110-114, n = Vector_Number - 71
```

Therefore, vector 64 corresponds to n = 0, vector 65 to n = 1, etc., vector 113 to n = 42, and vector 114 to n = 43.

Each peripheral request input is first qualified by the contents of the IMR registers before it is used elsewhere in the interrupt controller, that is:

```
qualified_interrupt_request[n] = interrupt_request_input[n] + ~INTC_IMR[n]
```

Since this interrupt controller supports 44 request inputs, the upper 20 bits of the INTC_IMRH are reserved for future use. Writes to these bits are ignored and reads return zeroes.

The contents of the IMR do not affect the operation of the software settable force interrupt registers.

Address: FFFF_FFC0h base + Ch offset = FFFF_FFCCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	IMR31	IMR30	IMR29	IMR28	IMR27	IMR26	IMR25	IMR24	IMR23	IMR22	IMR21	IMR20	IMR19	IMR18	IMR17	IMR16
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	IMR15	IMR14	IMR13	IMR12	IMR11	IMR10	IMR9	IMR8	IMR7	IMR6	IMR5	IMR4	IMR3	IMR2	IMR1	IMR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IMRL field descriptions

Field	Description
31 IMR31	Interrupt mask register 31 0 The interrupt request is enabled. 1 The interrupt request is disabled.
30 IMR30	Interrupt mask register 30 0 The interrupt request is enabled. 1 The interrupt request is disabled.
29 IMR29	Interrupt mask register 29 0 The interrupt request is enabled. 1 The interrupt request is disabled.
28 IMR28	Interrupt mask register 28 0 The interrupt request is enabled. 1 The interrupt request is disabled.
27 IMR27	Interrupt mask register 27 0 The interrupt request is enabled. 1 The interrupt request is disabled.
26 IMR26	Interrupt mask register 26 0 The interrupt request is enabled. 1 The interrupt request is disabled.
25 IMR25	Interrupt mask register 25 0 The interrupt request is enabled. 1 The interrupt request is disabled.

Table continues on the next page...

INTC_IMRL field descriptions (continued)

Field	Description
24 IMR24	Interrupt mask register 24 0 The interrupt request is enabled. 1 The interrupt request is disabled.
23 IMR23	Interrupt mask register 23 0 The interrupt request is enabled. 1 The interrupt request is disabled.
22 IMR22	Interrupt mask register 22 0 The interrupt request is enabled. 1 The interrupt request is disabled.
21 IMR21	Interrupt mask register 21 0 The interrupt request is enabled. 1 The interrupt request is disabled.
20 IMR20	Interrupt mask register 20 0 The interrupt request is enabled. 1 The interrupt request is disabled.
19 IMR19	Interrupt mask register 19 0 The interrupt request is enabled. 1 The interrupt request is disabled.
18 IMR18	Interrupt mask register 18 0 The interrupt request is enabled. 1 The interrupt request is disabled.
17 IMR17	Interrupt mask register 17 0 The interrupt request is enabled. 1 The interrupt request is disabled.
16 IMR16	Interrupt mask register 16 0 The interrupt request is enabled. 1 The interrupt request is disabled.
15 IMR15	Interrupt mask register 15 0 The interrupt request is enabled. 1 The interrupt request is disabled.
14 IMR14	Interrupt mask register 14 0 The interrupt request is enabled. 1 The interrupt request is disabled.
13 IMR13	Interrupt mask register 13 0 The interrupt request is enabled. 1 The interrupt request is disabled.

Table continues on the next page...

INTC_IMRL field descriptions (continued)

Field	Description
12 IMR12	Interrupt mask register 12 0 The interrupt request is enabled. 1 The interrupt request is disabled.
11 IMR11	Interrupt mask register 11 0 The interrupt request is enabled. 1 The interrupt request is disabled.
10 IMR10	Interrupt mask register 10 0 The interrupt request is enabled. 1 The interrupt request is disabled.
9 IMR9	Interrupt mask register 9 0 The interrupt request is enabled. 1 The interrupt request is disabled.
8 IMR8	Interrupt mask register 8 0 The interrupt request is enabled. 1 The interrupt request is disabled.
7 IMR7	Interrupt mask register 7 0 The interrupt request is enabled. 1 The interrupt request is disabled.
6 IMR6	Interrupt mask register 6 0 The interrupt request is enabled. 1 The interrupt request is disabled.
5 IMR5	Interrupt mask register 5 0 The interrupt request is enabled. 1 The interrupt request is disabled.
4 IMR4	Interrupt mask register 4 0 The interrupt request is enabled. 1 The interrupt request is disabled.
3 IMR3	Interrupt mask register 3 0 The interrupt request is enabled. 1 The interrupt request is disabled.
2 IMR2	Interrupt mask register 2 0 The interrupt request is enabled. 1 The interrupt request is disabled.
1 IMR1	Interrupt mask register 1 0 The interrupt request is enabled. 1 The interrupt request is disabled.

Table continues on the next page...

INTC_IMRL field descriptions (continued)

Field	Description
0 IMR0	Interrupt mask register 0 0 The interrupt request is enabled. 1 The interrupt request is disabled.

15.4.3 Force Interrupt Register (INTC_FRC)

The INTC_FRC register allows software to generate a unique interrupt for each possible level at the lowest priority within the level for functional or debug purposes. These interrupts may be self-scheduled by setting one or more of the bits in the INTC_FRC register. In some cases, the handling of a normal interrupt request may cause critical processing by the service routine along with the scheduling (using the INTC_FRC register) of a lower priority level interrupt request to be processed at a later time for less-critical task handling.

The INTC_FRC register may be modified directly using a read-modify-write sequence or through a simple write operation using the set/clear force interrupt registers (INTC_SFRC, INTC_CFRC).

NOTE

For compatibility with other ColdFire interrupt controllers, this register's bit numbers are actually 63-56 (not 7-0 as shown in the register diagram).

Address: FFFF_FFC0h base + 10h offset = FFFF_FFD0h

Bit	7	6	5	4	3	2	1	0
Read	0	LVL1	LVL2	LVL3	LVL4	LVL5	LVL6	LVL7
Write								
Reset	0	0	0	0	0	0	0	0

INTC_FRC field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 LVL1	Force level 1 interrupt 0 Negates the forced level 1 interrupt request. 1 Forces a level 1 interrupt request.
5 LVL2	Force level 2 interrupt 0 Negates the forced level 2 interrupt request. 1 Forces a level 2 interrupt request.

Table continues on the next page...

INTC_FRC field descriptions (continued)

Field	Description
4 LVL3	Force level 3 interrupt 0 Negates the forced level 3 interrupt request. 1 Forces a level 3 interrupt request.
3 LVL4	Force level 4 interrupt 0 Negates the forced level 4 interrupt request. 1 Forces a level 4 interrupt request.
2 LVL5	Force level 5 interrupt 0 Negates the forced level 5 interrupt request. 1 Forces a level 5 interrupt request.
1 LVL6	Force level 6 interrupt 0 Negates the forced level 6 interrupt request. 1 Forces a level 6 interrupt request.
0 LVL7	Force level 7 interrupt 0 Negates the forced level 7 interrupt request. 1 Forces a level 7 interrupt request.

15.4.4 INTC Programmable Level 6 Priority Registers (INTC_PL6P*n*)

The level seven interrupt requests cannot have their levels reassigned. However, any of the remaining peripheral interrupt requests can be reassigned as the highest priority maskable requests using these two registers (INTC_PL6P7 and INTC_PL6P6). The vector number associated with the interrupt requests does not change. Rather, only the interrupt request's level and priority are altered, based on the contents of the INTC_PL6P{7,6} registers.

NOTE

The requests associated with the INTC_FRC register have a fixed level and priority that cannot be altered.

The INTC_PL6P7 register specifies the highest-priority, maskable interrupt request that is defined as the level six, priority seven request. The INTC_PL6P6 register specifies the second-highest-priority, maskable interrupt request defined as the level six, priority six request. Reset clears both registers, disabling any request re-mapping.

For an example of the use of these registers, see Using INTC_PL6P{7,6} Registers.

Address: FFFF_FFC0h base + 18h offset + (1d × i), where i=0d to 1d

Bit	7	6	5	4	3	2	1	0
Read	0		REQN					
Write								
Reset	0	0	0	0	0	0	0	0

INTC_PL6P*n* field descriptions

Field	Description
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–0 REQN	Request number Defines the peripheral IRQ number to be remapped as the level 6, priority 7 (for INTC_PL6P7) request and level 6, priority 6 (for INTC_PL6P6) request. **NOTE:** The value must be a valid interrupt number. Unused or reserved interrupt numbers are ignored. The selected vector number is derived from the decimal value of the REQN field, which is *n* in the following mapping formula: for vectors 64-102, *n* = *Vector_Number - 64*, while for vectors 110-114, *n* = *Vector_Number - 71*. In other words, the REQN field's minimum value is 4d (selecting vector 68) and maximum value is 43d (selecting vector 114).

15.4.5 INTC Wakeup Control Register (INTC_WCR)

The interrupt controller provides a combinatorial logic path to generate a special wakeup signal to exit from the wait or stop modes. The INTC_WCR register defines wakeup condition for interrupt recognition during wait and stop modes. This mode of operation works as follows:

1. Write to the INTC_WCR to enable this operation (set INTC_WCR[ENB]) and define the interrupt mask level needed to force the core to exit wait or stop mode (INTC_WCR[MASK]). The maximum value of INTC_WCR[MASK] is 0x6 (0b110). The INTC_WCR is enabled with a mask level of 0 as the default after reset.
2. Execute a stop instruction to place the processor into wait or stop mode.
3. After the processor is stopped, the interrupt controller enables special logic that evaluates the incoming interrupt sources in a purely combinatorial path; no clocked storage elements are involved.
4. If an active interrupt request is asserted and the resulting interrupt level is greater than the mask value contained in INTC_WCR[MASK], the interrupt controller asserts the wakeup output signal. This signal is routed to the clock generation logic to exit the low-power mode and resume processing.

Typically, the interrupt mask level loaded into the processor's status register field (SR[I]) during the execution of the stop instruction matches the INTC_WCR[MASK] value.

The interrupt controller wakeup signal is defined as:

```
wakeup = INTC_WCR[ENB] + (level of any asserted_int_request > INTC_WCR[MASK])
```

Address: FFFF_FFC0h base + 1Bh offset = FFFF_FFDBh

Bit	7	6	5	4	3	2	1	0
Read	ENB	0				MASK		
Write								
Reset	1	0	0	0	0	0	0	0

INTC_WCR field descriptions

Field	Description
7 ENB	Enable wakeup signal 0 Wakeup signal disabled. 1 Enables the assertion of the combinational wakeup signal to the clock generation logic.
6–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–0 MASK	Interrupt mask level

15.4.6 Set Interrupt Mask Register (INTC_SIMR)

The INTC_SIMR register provides a simple memory-mapped mechanism to set a given bit in the INTC_IMR{H,L} registers to disable (mask) a given interrupt request. The data value on a register write causes the corresponding bit in the INTC_IMR{H,L} registers to be set. Setting INTC_SIMR[SALL] forces the entire contents of INTC_IMR{H,L} registers to set, masking all interrupts. Attempting to read this register generates an error termination.

IMR[63:44] are reserved for future use, so writes using these data values are ignored.

This register is provided so interrupt service routines can easily mask the given interrupt request without the need to perform a read-modify-write sequence on the INTC_IMR{H,L} registers.

Address: FFFF_FFC0h base + 1Ch offset = FFFF_FFDCh

Bit	7	6	5	4	3	2	1	0
Read	Reserved							
Write		SALL	SIMR					
Reset	0	0	0	0	0	0	0	0

INTC_SIMR field descriptions

Field	Description
7 Reserved	This field is reserved.
6 SALL	Set all Set all bits in the INTC_IMR{H,L} register, masking all interrupt requests. 0 Set only those bits specified in the SIMR field. 1 Set all bits in INTC_IMR{H,L} register. The SIMR field is ignored.
5–0 SIMR	Set IMR Set the corresponding bit in the INTC_IMR{H,L} register, masking the interrupt request.

15.4.7 Clear Interrupt Mask Register (INTC_CIMR)

The INTC_SIMR register provides a simple memory-mapped mechanism to set a given bit in the INTC_IMR{H,L} registers to disable (mask) a given interrupt request. The data value on a register write causes the corresponding bit in the INTC_IMR{H,L} registers to be set. Setting INTC_SIMR[SALL] forces the entire contents of INTC_IMR{H,L} registers to set, masking all interrupts. Attempting to read this register generates an error termination.

IMR[63:44] are reserved for future use, so writes using these data values are ignored.

This register is provided so interrupt service routines can easily mask the given interrupt request without the need to perform a read-modify-write sequence on the INTC_IMR{H,L} registers.

Address: FFFF_FFC0h base + 1Dh offset = FFFF_FFDDh

Bit	7	6	5	4	3	2	1	0
Read	Reserved							
Write		CALL	CIMR					
Reset	0	0	0	0	0	0	0	0

INTC_CIMR field descriptions

Field	Description
7 Reserved	This field is reserved.
6 CALL	Clear all Clear all bits in the INTC_IMR{H,L} register, enabling all interrupt requests. 0 Set only those bits specified in the CIMR field. 1 Clear all bits in INTC_IMR{H,L} register. The CIMR field is ignored.

Table continues on the next page...

INTC_CIMR field descriptions (continued)

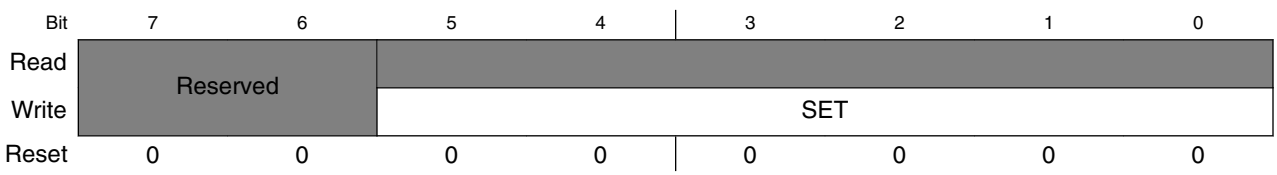
Field	Description
5–0 CIMR	Clear the corresponding bit in the INTC_IMR{H,L} registers, enabling the interrupt request.

15.4.8 INTC Set Interrupt Force Register (INTC_SFRC)

The INTC_SFRC register provides a simple memory-mapped mechanism to set a given bit in the INTC_FRC register to assert a specific level interrupt request. The data value written causes the appropriate bit in the INTC_FRC register to be set. Attempted reads of this register generate an error termination.

This register is provided so interrupt service routines can generate a forced interrupt request without the need to perform a read-modify-write sequence on the INTC_FRC register.

Address: FFFF_FFC0h base + 1Eh offset = FFFF_FFDEh



INTC_SFRC field descriptions

Field	Description
7–6 Reserved	This field is reserved.
5–0 SET	For data values within the 56–62 range, the corresponding bit in the INTC_FRC register is set, as defined below. **NOTE:** Data values outside this range do not affect the INTC_FRC register. It is recommended the data values be restricted to the 0x38–0x3E (56–62) range to ensure compatibility with future devices. 111000 Bit 56, INTC_FRC[LVL7] is set 111001 Bit 57, INTC_FRC[LVL6] is set 111010 Bit 58, INTC_FRC[LVL5] is set 111011 Bit 59, INTC_FRC[LVL4] is set 111100 Bit 60, INTC_FRC[LVL3] is set 111101 Bit 61, INTC_FRC[LVL2] is set 111110 Bit 62, INTC_FRC[LVL1] is set

15.4.9 INTC Clear Interrupt Force Register (INTC_CFRC)

The INTC_CFRC register provides a simple memory-mapped mechanism to clear a given bit in the INTC_FRC register to negate a specific level interrupt request. The data value on the register write causes the appropriate bit in the INTC_FRC register to be cleared. Attempted reads of this register generate an error termination.

This register is provided so interrupt service routines can negate a forced interrupt request without the need to perform a read-modify-write sequence on the INTC_FRC register.

Address: FFFF_FFC0h base + 1Fh offset = FFFF_FFDFh

Bit	7	6	5	4	3	2	1	0
Read								
Write	Reserved		CLR					
Reset	0	0	0	0	0	0	0	0

INTC_CFRC field descriptions

Field	Description
7–6 Reserved	This field is reserved.
5–0 CLR	For data values within the 56–62 range, the corresponding bit in the INTC_FRC register is cleared, as defined below. **NOTE:** Data values outside this range do not affect the INTC_FRC register. It is recommended the data values be restricted to the 0x38–0x3E (56–62) range to ensure compatibility with future devices. 111000 Bit 56, INTC_FRC[LVL7] is cleared 111001 Bit 57, INTC_FRC[LVL6] is cleared 111010 Bit 58, INTC_FRC[LVL5] is cleared 111011 Bit 59, INTC_FRC[LVL4] is cleared 111100 Bit 60, INTC_FRC[LVL3] is cleared 111101 Bit 61, INTC_FRC[LVL2] is cleared 111110 Bit 62, INTC_FRC[LVL1] is cleared

15.4.10 INTC Software IACK Register (INTC_SWIACK)

Refer to the description of the Level-*n* IACK registers.

Address: FFFF_FFC0h base + 20h offset = FFFF_FFE0h

Bit	7	6	5	4	3	2	1	0
Read	0	VECN						
Write								
Reset	0	0	0	0	0	0	0	0

INTC_SWIACK field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–0 VECN	Vector number Indicates the appropriate vector number. It is the highest-level, highest-priority request currently being asserted in the CF1_INTC module. If there are no pending requests, VECN is zero.

15.4.11 INTC Level-*n* IACK Registers (INTC_LVL*n*IACK)

The eight read-only interrupt acknowledge (IACK) registers can be explicitly addressed by the memory-mapped accesses or implicitly addressed by a processor-generated interrupt acknowledge cycle during exception processing when CPUCR[IAE] is set. In either case, the interrupt controller's actions are similar.

First, consider an IACK cycle to a specific level, a level-n IACK. When this type of IACK arrives in the interrupt controller, the controller examines all currently-active level-n interrupt requests, determines the highest priority within the level, and then responds with the unique vector number corresponding to that specific interrupt source. The vector number is supplied as the data for the byte-sized IACK read cycle.

If there is no active interrupt source at the time of the level-n IACK, a special spurious interrupt vector (vector number 24 [0x18]) is returned. It is the responsibility of the service routine to manage this error situation.

This protocol implies the interrupting peripheral is not accessed during the acknowledge cycle because the interrupt controller completely services the acknowledge. This means the interrupt source must be explicitly disabled in the peripheral device by the interrupt service routine. This approach provides unique vector capability for all interrupt requests, regardless of the complexity of the peripheral device.

Second, the interrupt controller also supports the concept of a software IACK. This is the ability to query the interrupt controller near the end of an interrupt service routine (after the current interrupt request has been negated) to determine if there are any pending (but currently masked) interrupt requests. If the response to the software IACK's byte operand read is non-zero, the service routine uses the returned value as the vector number of the highest pending interrupt request and passes control to the appropriate new handler. If the returned value is zero, there is no pending interrupt request.

This process avoids the overhead of a context restore and RTE instruction execution, followed immediately by another interrupt exception and context save. In system environments with high rates of interrupt activity, this mechanism can noticeably improve overall performance.

Address: FFFF_FFC0h base + 24h offset + (4d × i), where i=0d to 6d

Bit	7	6	5	4	3	2	1	0
Read	0	VECN						
Write								
Reset	0	0	0	1	1	0	0	0

INTC_LVL*n*IACK field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–0 VECN	Vector number Indicates the appropriate vector number. It is the highest priority request within the specified level-n. If there are no pending requests within the level, VECN is 0x18 (24) to signal a spurious interrupt.

15.5 Functional Description

The basic operation of the CF1_INTC is detailed in the preceding sections. This section describes special rules applicable to non-maskable level 7 interrupt requests and the module's interfaces.

15.5.1 Handling of Non-Maskable Level 7 Interrupt Requests

In this context of this discussion, the non-maskable level 7 interrupt requests refer only to the masking capability provided by the processor's SR[I] field. The ability to mask individual interrupt requests using the interrupt controller's IMR is always available, regardless of the level of a particular interrupt request.

The CPU treats level 7 interrupts as non-maskable, edge-sensitive requests, while levels 1 through 6 are maskable, level-sensitive requests. As a result of this definition, level 7 interrupt requests are a special case. The edge-sensitive nature of these requests means the encoded 3-bit level input from the CF1_INTC to the V1 ColdFire core must change state before the CPU detects an interrupt. A non-maskable interrupt (NMI) is generated each time the encoded interrupt level changes to level 7 (regardless of the SR[I] field) and each time the SR[I] mask changes from 7 to a lower value while the encoded request level remains at 7.

15.6 Initialization Information

The reset state of the CF1_INTC module enables the default IRQ mappings and clears any software-forced interrupt requests (INTC_FRC is cleared). Immediately after reset, the CF1_INTC begins its cycle-by-cycle evaluation of any asserted interrupt requests and forms the appropriate encoded interrupt level and vector information for the V1 ColdFire processor core. The ability to mask individual interrupt requests using the interrupt controller's IMR is always available, regardless of the level of a particular interrupt request.

15.7 Application Information

This section discusses three application topics: emulation of the HCS08's one level interrupt nesting structure, elevating the priority of two IRQs, and more details on the operation of the software interrupt acknowledge (SWIACK) mechanism.

15.7.1 Emulation of the HCS08's 1-Level IRQ Handling

As noted in Table 10-7, the HCS08 architecture specifies a 1-level IRQ nesting capability. Interrupt masking is controlled by CCR[I], the interrupt mask flag: clearing CCR[I] enables interrupts, while setting CCR[I] disables interrupts. The ColdFire

architecture defines seven interrupt levels, controlled by the 3-bit interrupt priority mask field in the status register, SR[I], and the hardware automatically supports nesting of interrupts.

To emulate the HCS08's 1-level IRQ capabilities on V1 ColdFire, only two SR[I] settings are used:

- Writing 0 to SR[I] enables interrupts.
- Writing 7 to SR[I] disables interrupts.

The ColdFire core treats the level seven requests as non-maskable, edge-sensitive interrupts.

ColdFire processors inhibit interrupt sampling during the first instruction of all exception handlers. This allows any handler to effectively disable interrupts, if necessary, by raising the interrupt mask level contained in the status register as the first instruction in the ISR. In addition, the V1 instruction set architecture (ISA_C) includes an instruction (STLDSR) that stores the current interrupt mask level and loads a value into the SR. This instruction is specifically intended for use as the first instruction of an interrupt service routine that services multiple interrupt requests with different interrupt levels. For more details see the *ColdFire Family Programmer's Reference Manual*. A MOVE-to-SR instruction also performs a similar function.

To emulate the HCS08's 1-level IRQ nesting mechanisms, the ColdFire implementation enables interrupts by clearing SR[I] (typically when using RTE to return to a process) and disables interrupts upon entering every interrupt service routine by one of three methods:

1. Execution of STLDSR #0x2700 as the first instruction of an ISR.
2. Execution of MOVE.w #0x2700,SR as the first instruction of an ISR.
3. Static assertion of CPUCR[IME] that forces the processor to load SR[I] with seven automatically upon the occurrence of an interrupt exception. Because this method removes the need to execute multi-cycle instructions of #1 or #2, this approach improves system performance.

15.7.2 Using INTC_PL6P{7,6} Registers

The INTC Programmable Level 6, Priority {7,6} registers (INTC_PL6P{7,6}) provide the ability to dynamically alter the request level and priority of two IRQs. Specifically, these registers provide the ability to reassign two IRQs to be the highest level 6 (maskable) requests. Consider the following example.

Suppose the system operation desires to remap the receive and transmit interrupt requests of a serial communication device (SCI1) as the highest two maskable interrupts. The default assignments for the SCI1 transmit and receive interrupts are:

- sci1_rx = interrupt source 13 (0Dh) = vector 77 = level 24, priority 6
- sci1_tx = interrupt source 14 (0Eh) = vector 78 = level 24, priority 5

To remap these two requests, the INTC_PL6P{7,6} registers are programmed with the desired interrupt source number:

- Setting INTC_PL6P7 to 13 (0Dh), remaps sci1_rx as level 6, priority 7.
- Setting INTC_PL6P6 to 14 (0Eh), remaps sci1_tx as level 6, priority 6.

The reset state of the INTC_PL6P{7,6} registers disables any request remapping.

15.7.3 More on Software IACKs

As previously mentioned, the notion of a software IACK refers to the ability to query the interrupt controller near the end of an interrupt service routine (after the current interrupt request has been cleared) to determine if there are any pending (but currently masked) interrupt requests. If the response to the software IACK's byte operand read is non-zero, the service routine uses the value as the vector number of the highest pending interrupt request and passes control to the appropriate new handler. This process avoids the overhead of a context restore and RTE instruction execution, followed immediately by another interrupt exception and context save. In system environments with high rates of interrupt activity, this mechanism can improve overall system performance noticeably.

To illustrate this concept, consider the following ISR code snippet shown in Figure 15-22.

```
                    align   4
                    irqxx_entry:
00588: 4fef fff0    lea     -16(sp),sp          # allocate stack space
0058c: 48d7 0303    movem.l #0x0303,(sp)        # save d0/d1/a0/a1 on stack

                    irqxx_alternate_entry:
00590:
       ....
                    irqxx_swiack:
005c0: 71b8 ffe0    mvz.b   INTC_SWIACK.w,d0    # perform software IACK
005c4: 0c00 0041    cmpi.b  #0x41,d0            # pending IRQ or level 7?
005c8: 6f0a         ble.b   irqxx_exit          # no pending IRQ, then exit
005ca: 91c8         sub.l   a0,a0               # clear a0
005cc: 2270 0c00    move.l  0(a0,d0.l*4),a1     # fetch pointer from xcpt table
005d0: 4ee9 0008    jmp     8(a1)               # goto alternate isr entry point

                    align   4
                    irqxx_exit:
005d4: 4cd7 0303    movem.l (sp),#0x0303        # restore d0/d1/a0/a1
005d8: 4fef 0010    lea     16(sp),sp           # deallocate stack space
005dc: 4e73         rte                         # return from handler
```

Figure 15-22. ISR Code Snippet with SWIACK

This snippet includes the prologue and epilogue for an interrupt service routine as well as code needed to perform software IACK.

At the entry point (irqxx_entry), there is a two-instruction prologue to allocate space on the supervisor stack to save the four volatile registers (d0, d1, a0, a1) defined in the ColdFire application binary interface. After saving these registers, the ISR continues at the alternate entry point.

The software IACK is performed near the end of the ISR, after the source of the current interrupt request is negated. First, the appropriate memory-mapped byte location in the interrupt controller is read (PC = 0x5C0). The CF1_INTC module returns the vector number of the highest priority pending request. If no request is pending, zero is returned. The compare instruction is needed to manage a special case involving pending level seven requests. Because the level seven requests are non-maskable, the ISR is interrupted to service one of these requests. To avoid any race conditions, this check ignores the level seven vector numbers. The result is the conditional branch (PC = 0x5C8) is taken if there are no pending requests or if the pending request is a level seven.

If there is a pending non-level seven request, execution continues with a three instruction sequence to calculate and then branch to the appropriate alternate ISR entry point. This sequence assumes the exception vector table is based at address 0x(00)00_0000 and that each ISR uses the same two-instruction prologue shown here. The resulting alternate entry point is a fixed offset (8 bytes) from the normal entry point defined in the exception vector table.

The ISR epilogue includes a three instruction sequence to restore the volatile registers from the stack and return from the interrupt exception.

This example is intentionally simple, but does show how performing the software IACK and passing control to an alternate entry point when there is a pending but masked interrupt request can avoid the execution of the ISR epilogue, another interrupt exception, and the ISR prologue.

Chapter 16
Low Leakage Wakeup Unit (LLWU)

16.1 Introduction

The LLWU module allows the user to select up to 16 external pin sources and up to 8 internal modules as a wakeup source from low-leakage power modes. The input sources are described in the device's chip configuration details. Each of the available wakeup sources can be individually enabled.

The $\overline{\text{RESET}}$ pin is an additional source for triggering an exit from low-leakage power modes, and causes the MCU to exit VLLS through a reset flow. The LLWU_RST[LLRSTE] bit must be set to allow an exit from low-leakage modes via the $\overline{\text{RESET}}$ pin. On a device where the $\overline{\text{RESET}}$ pin is shared with other functions, the explicit port mux control register must be set for the $\overline{\text{RESET}}$ pin before the $\overline{\text{RESET}}$ pin can be used as a low-leakage reset source.

The LLWU module also includes three optional digital pin filters: two for the external wakeup pins and one for the $\overline{\text{RESET}}$ pin.

16.1.1 Features

The LLWU module features include:

- Support for up to 16 external input pins and up to 8 internal modules with individual enable bits
- Input sources may be external pins or from internal peripherals capable of running in VLLS. See the chip configuration information for wakeup input sources for this device.
- External pin wakeup inputs, each of which is programmable as falling-edge, rising-edge, or any change

- Wakeup inputs that are activated if enabled after MCU enters a low-leakage power mode
- Optional digital filters provided to qualify an external pin detect and $\overline{\text{RESET}}$ pin detect.

16.1.2 Modes of operation

The LLWU module becomes functional on entry into a low-leakage power mode. After recovery from VLLS, the LLWU continues to detect wakeup events until the user has acknowledged the wakeup via a write to the PMC_REGSC[ACKISO] bit.

16.1.2.1 VLLS modes

The LLWU module provides up to 16 external wakeup inputs and up to 8 internal module wakeup inputs. A VLLS reset event can be initiated via assertion of the $\overline{\text{RESET}}$ pin. All wakeup and reset events result in VLLS exit via a reset flow.

16.1.2.2 Non-low leakage modes

The LLWU is not active in all non-low leakage modes where detection and control logic are in a static state. The LLWU registers are accessible in non-low leakage modes and are available for configuring and reading status when bus transactions are possible.

When the$\overline{\text{RESET}}$ pin filter or wakeup pin filters are enabled, filter operation begins immediately. If a low leakage mode is entered within 5 LPO clock cycles of an active edge, the edge event will be detected by the LLWU. For $\overline{\text{RESET}}$ pin filtering, this means that there is no restart to the minimum LPO cycle duration as the filtering transitions from a non-low leakage filter, which is implemented in the RCM, to the LLWU filter.

16.1.2.3 Debug mode

When the chip is in Debug mode and then enters a VLLSx mode, no debug logic works in the fully-functional low-leakage mode. Upon an exit from the VLLSx mode, the LLWU becomes inactive.

16.1.3 Block diagram

The following figure is the block diagram for the LLWU module.

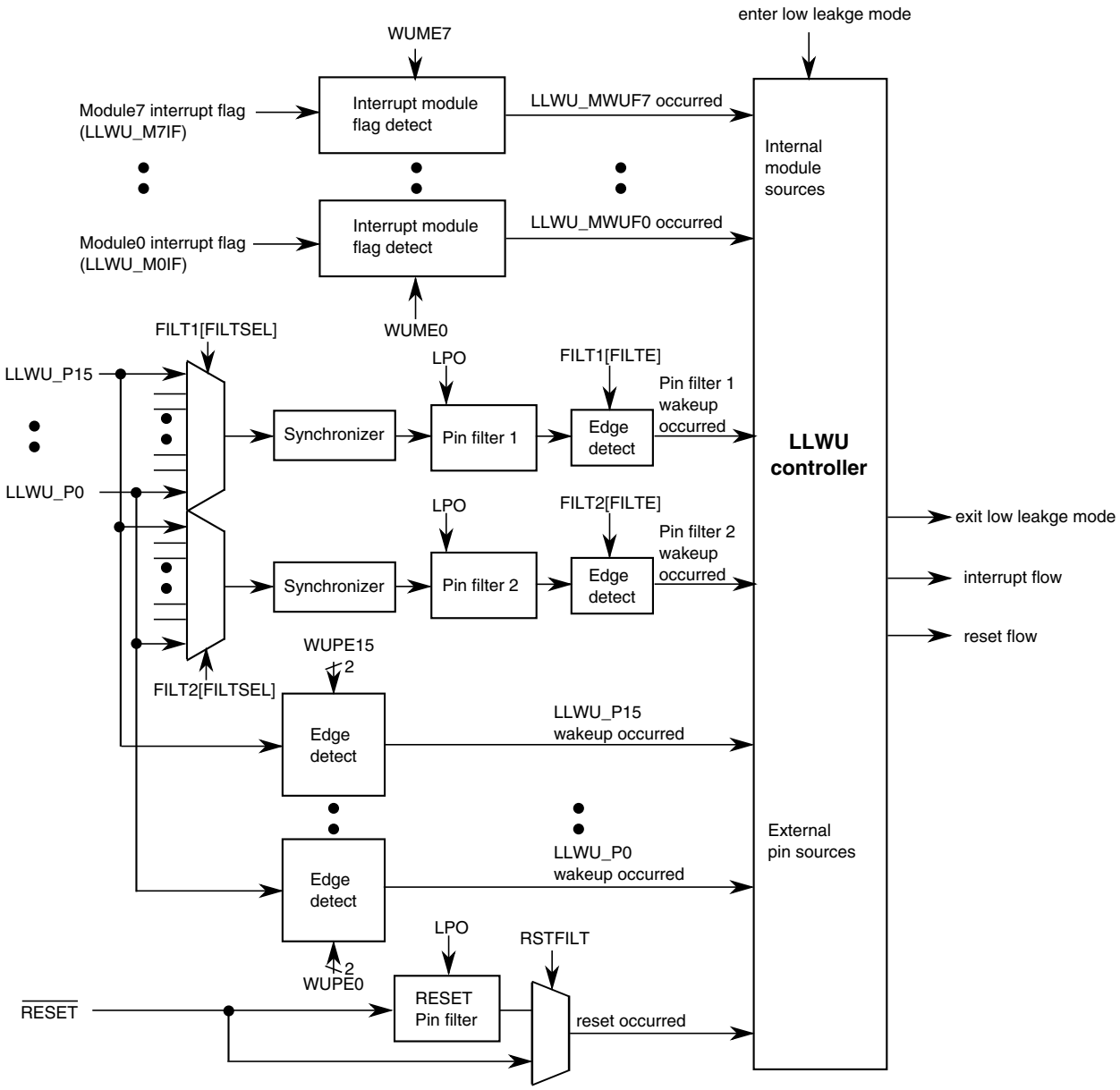


Figure 16-1. LLWU block diagram

16.2 LLWU signal descriptions

The signal properties of LLWU are shown in the following table. The external wakeup input pins can be enabled to detect either rising-edge, falling-edge, or on any change.

Table 16-1. LLWU signal descriptions

Signal	Description	I/O
LLWU_Pn	Wakeup inputs (n = 0-15)	I

16.3 Memory map/register definition

The LLWU includes the following registers:

- Five 8-bit wakeup source enable registers
  - Enable external pin input sources
  - Enable internal peripheral sources
- Three 8-bit wakeup flag registers
  - Indication of wakeup source that caused exit from a low-leakage power mode includes external pin or internal module interrupt
- Two 8-bit wakeup pin filter enable registers
- One 8-bit $\overline{\text{RESET}}$ pin filter enable register

NOTE

The LLWU registers can be written only in supervisor mode. Write accesses in user mode are blocked and will result in a bus error.

All LLWU registers are reset by Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. Each register's displayed reset value represents this subset of reset types. LLWU registers are unaffected by reset types that do not trigger Chip Reset not VLLS. For more information about the types of reset on this chip, refer to the Introduction details.

LLWU memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_8090	LLWU Pin Enable 1 register (LLWU_PE1)	8	R/W	00h	16.3.1/289
FFFF_8091	LLWU Pin Enable 2 register (LLWU_PE2)	8	R/W	00h	16.3.2/290
FFFF_8092	LLWU Pin Enable 3 register (LLWU_PE3)	8	R/W	00h	16.3.3/291
FFFF_8093	LLWU Pin Enable 4 register (LLWU_PE4)	8	R/W	00h	16.3.4/292
FFFF_8094	LLWU Module Enable register (LLWU_ME)	8	R/W	00h	16.3.5/293

Table continues on the next page...

LLWU memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_8095	LLWU Flag 1 register (LLWU_F1)	8	R/W	00h	16.3.6/295
FFFF_8096	LLWU Flag 2 register (LLWU_F2)	8	R/W	00h	16.3.7/296
FFFF_8097	LLWU Flag 3 register (LLWU_F3)	8	R	00h	16.3.8/298
FFFF_8098	LLWU Pin Filter 1 register (LLWU_FILT1)	8	R/W	00h	16.3.9/300
FFFF_8099	LLWU Pin Filter 2 register (LLWU_FILT2)	8	R/W	00h	16.3.10/301
FFFF_809A	LLWU Reset Enable register (LLWU_RST)	8	R/W	02h	16.3.11/302

16.3.1 LLWU Pin Enable 1 register (LLWU_PE1)

LLWU_PE1 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P3-LLWU_P0.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: FFFF_8090h base + 0h offset = FFFF_8090h

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
1–0 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

16.3.2 LLWU Pin Enable 2 register (LLWU_PE2)

LLWU_PE2 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P7-LLWU_P4.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: FFFF_8090h base + 1h offset = FFFF_8091h

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
1–0 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

16.3.3 LLWU Pin Enable 3 register (LLWU_PE3)

LLWU_PE3 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P11-LLWU_P8.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: FFFF_8090h base + 2h offset = FFFF_8092h

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
1–0 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

16.3.4 LLWU Pin Enable 4 register (LLWU_PE4)

LLWU_PE4 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P15-LLWU_P12.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: FFFF_8090h base + 3h offset = FFFF_8093h

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
1–0 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

16.3.5 LLWU Module Enable register (LLWU_ME)

LLWU_ME contains the bits to enable the internal module flag as a wakeup input source for inputs MWUF7-MWUF0.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: FFFF_8090h base + 4h offset = FFFF_8094h

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. 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.

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

16.3.6 LLWU Flag 1 register (LLWU_F1)

LLWU_F1 contains the wakeup flags indicating which wakeup source caused the MCU to exit VLLS mode. 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: FFFF_8090h base + 5h offset = FFFF_8095h

Bit	7	6	5	4	3	2	1	0
Read	WUF7	WUF6	WUF5	WUF4	WUF3	WUF2	WUF1	WUF0
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

LLWU_F1 field descriptions

Field	Description
7 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

Table continues on the next page...

LLWU_F1 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 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 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

16.3.7 LLWU Flag 2 register (LLWU_F2)

LLWU_F2 contains the wakeup flags indicating which wakeup source caused the MCU to exit or VLLS mode. 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: FFFF_8090h base + 6h offset = FFFF_8096h

Bit	7	6	5	4	3	2	1	0
Read	WUF15	WUF14	WUF13	WUF12	WUF11	WUF10	WUF9	WUF8
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

LLWU_F2 field descriptions

Field	Description
7 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. 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

Table continues on the next page...

LLWU_F2 field descriptions (continued)

Field	Description
	Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF10. 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

16.3.8 LLWU Flag 3 register (LLWU_F3)

LLWU_F3 contains the wakeup flags indicating which internal wakeup source caused the MCU to exit VLLS mode. 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 iRTC or CMP modules, 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: FFFF_8090h base + 7h offset = FFFF_8097h

Bit	7	6	5	4	3	2	1	0
Read	MWUF7	MWUF6	MWUF5	MWUF4	MWUF3	MWUF2	MWUF1	MWUF0
Write								
Reset	0	0	0	0	0	0	0	0

LLWU_F3 field descriptions

Field	Description
7 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_F3 field descriptions (continued)

Field	Description
	Indicates that an enabled internal peripheral was a source of exiting a low-leakage power mode. To clear the flag, follow the internal peripheral flag clearing mechanism. 0 Module 0 input was not a wakeup source 1 Module 0 input was a wakeup source

16.3.9 LLWU Pin Filter 1 register (LLWU_FILT1)

LLWU_FILT1 is a control and status register that is used to enable/disable the digital filter 1 features for an external pin.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: FFFF_8090h base + 8h offset = FFFF_8098h

Bit	7	6	5	4	3	2	1	0
Read	FILTF	FILTE		0	FILTSEL			
Write	w1c							
Reset	0	0	0	0	0	0	0	0

LLWU_FILT1 field descriptions

Field	Description
7 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
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

LLWU_FILT1 field descriptions (continued)

Field	Description
3–0 FILTSEL	Filter Pin Select Selects 1 out of the 16 wakeup pins to be muxed into the filter. 0000 Select LLWU_P0 for filter ... ... 1111 Select LLWU_P15 for filter

16.3.10 LLWU Pin Filter 2 register (LLWU_FILT2)

LLWU_FILT2 is a control and status register that is used to enable/disable the digital filter 2 features for an external pin.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: FFFF_8090h base + 9h offset = FFFF_8099h

Bit	7	6	5	4	3	2	1	0
Read	FILTF	FILTE		0	FILTSEL			
Write	w1c							
Reset	0	0	0	0	0	0	0	0

LLWU_FILT2 field descriptions

Field	Description
7 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
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

LLWU_FILT2 field descriptions (continued)

Field	Description
3–0 FILTSEL	Filter Pin Select Selects 1 out of the 16 wakeup pins to be muxed into the filter. 0000 Select LLWU_P0 for filter ... ... 1111 Select LLWU_P15 for filter

16.3.11 LLWU Reset Enable register (LLWU_RST)

LLWU_RST is a control register that is used to enable/disable the digital filter for the external pin detect and RESET pin.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: FFFF_8090h base + Ah offset = FFFF_809Ah

Bit	7	6	5	4	3	2	1	0
Read	0						LLRSTE	RSTFILT
Write								
Reset	0	0	0	0	0	0	1	0

LLWU_RST field descriptions

Field	Description
7–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 LLRSTE	Low-Leakage Mode RESET Enable This bit must be set to allow the device to be reset while in a low-leakage power mode. On devices where Reset is not a dedicated pin, the RESET pin must also be enabled in the explicit port mux control. 0 RESET pin not enabled as a leakage mode exit source 1 RESET pin enabled as a low leakage mode exit source
0 RSTFILT	Digital Filter On RESET Pin Enables the digital filter for the RESET pin during VLLS3, VLLS2, or VLLS1 modes. 0 Filter not enabled 1 Filter enabled

16.4 Functional description

This on-chip peripheral module is called a low-leakage wakeup unit (LLWU) module because it allows internal peripherals and external input pins as a source of wakeup from low-leakage modes. It is operational only in VLLSx modes.

The LLWU module contains pin enables for each external pin and internal module. For each external pin, the user can disable or select the edge type for the wakeup. Type options are:

- Falling-edge
- Rising-edge
- Either-edge

When an external pin is enabled as a wakeup source, the pin must be configured as an input pin.

The LLWU implements optional 3-cycle glitch filters, based on the LPO clock. A detected external pin, either wakeup or $\overline{\text{RESET}}$, is required to remain asserted until the enabled glitch filter times out. Additional latency of up to 2 cycles is due to synchronization, which results in a total of up to 5 cycles of delay before the detect circuit alerts the system to the wakeup or reset event when the filter function is enabled. Two wakeup detect filters are available to detect up to two external pins. A separate reset filter is on the $\overline{\text{RESET}}$ pin. Glitch filtering is not provided on the internal modules.

For internal module wakeup operation, the WUMEx bit enables the associated module as a wakeup source.

16.4.1 VLLS modes

In the case of a wakeup due to external pin or internal module wakeup, recovery is always via a reset flow and the RCM_SRS[WAKEUP] is set indicating the low-leakage mode was active. State retention data is lost and I/O will be restored after PMC_REGSC[ACKISO] has been written.

A VLLS exit event due to $\overline{\text{RESET}}$ pin assertion causes an exit via a system reset. State retention data is lost and the I/O states immediately return to their reset state. The RCM_SRS[WAKEUP] and RCM_SRS[PIN] bits are set and the system executes a reset flow before CPU operation begins with a reset vector fetch.

16.4.2 Initialization

For an enabled peripheral wakeup input, the peripheral flag must be cleared by software before entering 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 VLLSx mode.

After enabling an external pin filter or changing the source pin, wait at least 5 LPO clock cycles before entering VLLSx mode to allow the filter to initialize.

NOTE

After recovering from a VLLS mode, user must restore chip configuration before clearing ACKISO. In particular, pin configuration for enabled LLWU wakeup pins must be restored to avoid any LLWU flag from being falsely set when ACKISO is cleared.

The signal selected as a wakeup source pin must be a digital pin, as selected in the pin mux control.

Chapter 17
Reset Control Module (RCM)

17.1 Introduction

This chapter describes the registers of the Reset Control Module (RCM). The RCM implements many of the reset functions for the chip. See the chip's reset chapter for more information.

17.2 Reset memory map and register descriptions

The Reset Control Module (RCM) registers provide reset status information and reset filter control.

NOTE

The RCM registers can be written only in supervisor mode. Write accesses in user mode are blocked and will result in a bus error.

RCM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_8080	System Reset Status Register 0 (RCM_SRS0)	8	R	82h	17.2.1/306
FFFF_8081	System Reset Status Register 1 (RCM_SRS1)	8	R	00h	17.2.2/307
FFFF_8084	Reset Pin Filter Control register (RCM_RPFC)	8	R/W	00h	17.2.3/309
FFFF_8085	Reset Pin Filter Width register (RCM_RPFW)	8	R/W	00h	17.2.4/310
FFFF_8087	Mode Register (RCM_MR)	8	R	00h	17.2.5/311

17.2.1 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: FFFF_8080h base + 0h offset = FFFF_8080h

Bit	7	6	5	4	3	2	1	0
Read	POR	PIN	WDOG	ILOP	ILAD	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
5 WDOG	Watchdog Indicates a reset has been caused by the watchdog timer Computer Operating Properly (COP) timing out. This reset source can be blocked by disabling the COP watchdog: write 00 to the SIM's COPC[COPT] field. 0 Reset not caused by watchdog timeout 1 Reset caused by watchdog timeout
4 ILOP	Illegal opcode

Table continues on the next page...

RCM_SRS0 field descriptions (continued)

Field	Description
	Indicates a reset has been caused by an attempt to execute an unimplemented or illegal opcode. The STOP instruction is considered illegal if stop is disabled by ((SOPT4[STOPE] = 0) && (SOPT4[WAITE] = 0)) in the SIM. The HALT instruction is considered illegal if the BDM interface is disabled by XCSR[ENBDM] = 0. 0 Reset not caused by an illegal opcode 1 Reset caused by an illegal opcode
3 ILAD	Illegal address Indicates a reset has been caused by an attempt to access an illegal address in the memory map. 0 Reset not caused by an illegal access 1 Reset caused by an illegal access
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 the LVDRE bit is set and the supply drops below the LVD trip voltage, an LVD reset occurs. This bit is also set by POR. 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. 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

17.2.2 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: FFFF_8080h base + 1h offset = FFFF_8081h

Bit	7	6	5	4	3	2	1	0
Read	0	0	SACKERR	EZPT	BDFR	0	0	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 EZPT	EzPort Reset Indicates a reset has been caused by EzPort receiving the RESET command while the device is in EzPort mode. 0 Reset not caused by EzPort receiving the RESET command while the device is in EzPort mode 1 Reset caused by EzPort receiving the RESET command while the device is in EzPort mode
3 BDFR	Background Debug Force Reset Indicates a reset has been caused by the host debugger system setting of CSR2[BDFR] in the ColdFire core. 0 Reset not caused by host debugger system setting of CSR2[BDFR] 1 Reset caused by host debugger system setting of CSR2[BDFR]
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.

17.2.3 Reset Pin Filter Control register (RCM_RPFC)

NOTE

The reset values of bits 2-0 are for Chip POR only. They are unaffected by other reset types.

NOTE

The bus clock filter is reset when disabled or when entering stop mode. The LPO filter is reset when disabled or when entering any low leakage stop mode .

Address: FFFF_8080h base + 4h offset = FFFF_8084h

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 0 All filtering disabled 1 LPO clock filter enabled
1–0 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

17.2.4 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: FFFF_8080h base + 5h offset = FFFF_8085h

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.
4–0 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

17.2.5 Mode Register (RCM_MR)

This register includes read-only status flags to indicate the state of the mode pins during the last Chip Reset.

Address: FFFF_8080h base + 7h offset = FFFF_8087h

Bit	7	6	5	4	3	2	1	0
Read	0						EZP_MS	MS
Write								
Reset	0	0	0	0	0	0	0	0

RCM_MR field descriptions

Field	Description
7–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 EZP_MS	EZP_MS_B pin state Reflects the state of the $\overline{\text{EZP\_MS}}$ pin during the last Chip Reset 0 Pin deasserted (logic 1) 1 Pin asserted (logic 0)
0 MS	MS_B Pin State Reflects the state of the $\overline{\text{MS}}$ pin during the last Chip Reset 0 Pin deasserted (logic 1) 1 Pin asserted (logic 0)

Chapter 18
System Mode Controller (SMC)

18.1 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.

18.2 Modes of operation

The V1 ColdFire CPU has two primary modes of operation: run and stop. The STOP instruction is used to invoke both stop and wait modes. The CPU does not differentiate between stop and wait modes.

In addition, Freescale MCUs also augment stop, wait, and run modes in a number of ways. The power management controller (PMC) contains a run and a stop mode regulator. Run regulation is used in normal run, wait and stop modes. Stop mode regulation is used during all very low power and low leakage modes. During stop mode regulation, the bus frequencies are limited in the very low power modes.

The SMC provides the user with multiple power options. The Very Low Power Run (VLPR) mode can drastically reduce run time power when maximum bus frequency is not required to handle the application needs. From Normal Run mode, the Run Mode (RUNM) field can be modified to change the MCU into VLPR mode when limited frequency is sufficient for the application. From VLPR mode, a corresponding wait (VLPW) and stop (VLPS) mode can be entered.

Depending on the needs of the user application, a variety of stop modes are available that allow the state retention, partial power down or full power down of certain logic and/or memory. I/O states are held in all modes of operation. Several registers are used to configure the various modes of operation for the device.

The following table describes the power modes available for the device.

Table 18-1. Power modes

Mode	Description
RUN	The MCU can be run at full speed and the internal supply is fully regulated, that is, in run regulation. This mode is also referred to as Normal Run mode.
WAIT	The core clock is gated off. The system clock continues to operate. Bus clocks, if enabled, continue to operate. Run regulation is maintained.
STOP	The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid.
VLPR	The core, system, bus, and flash clock maximum frequencies are restricted in this mode. See the Power Management chapter for details about the maximum allowable frequencies.
VLPW	The core clock is gated off. The system, bus, and flash clocks continue to operate, although their maximum frequency is restricted. See the Power Management chapter for details on the maximum allowable frequencies.
VLPS	The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid.
VLLS3	The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid. The MCU is placed in a low leakage mode by powering down the internal logic. All system RAM contents are retained and I/O states are held. Internal logic states are not retained.
VLLS2	The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid.The MCU is placed in a low leakage mode by powering down the internal logic and the system RAM3 partition. The system RAM2 partition can be optionally retained using VLLSCTRL[RAM2PO]. The system RAM1 partition contents are retained in this mode. Internal logic states are not retained. [1]
VLLS1	The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid. The MCU is placed in a low leakage mode by powering down the internal logic and all system RAM. I/O states are held. Internal logic states are not retained.

1. See the devices' chip configuration details for the size and location of the system RAM partitions.

18.3 Memory map and register descriptions

Details follow about the registers related to the system mode controller.

Different SMC registers reset on different reset types. Each register's description provides details.

NOTE

The SMC registers can be written only in supervisor mode. Write accesses in user mode are blocked and will result in a bus error.

SMC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_80B0	Power Mode Protection register (SMC_PMPROT)	8	R/W	00h	18.3.1/315
FFFF_80B1	Power Mode Control register (SMC_PMCTRL)	8	R/W	00h	18.3.2/316
FFFF_80B2	VLLS Control register (SMC_VLLSCTRL)	8	R/W	03h	18.3.3/318
FFFF_80B3	Power Mode Status register (SMC_PMSTAT)	8	R	01h	18.3.4/319

18.3.1 Power Mode Protection register (SMC_PMPROT)

This register provides protection for entry into any low-power run or stop mode. The enabling of the low-power run or stop mode occurs by configuring the Power Mode Control register (PMCTRL).

The PMPROT register can be written only once after any system reset.

If the MCU is configured for a disallowed or reserved power mode, the MCU remains in its current power mode. For example, if the MCU is in normal RUN mode and AVLP is 0, an attempt to enter VLPR mode using PMCTRL[RUNM] is blocked and the RUNM bits remain 00b, indicating the MCU is still in Normal Run mode.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

Address: FFFF_80B0h base + 0h offset = FFFF_80B0h

Bit	7	6	5	4	3	2	1	0
Read	0	0	AVLP	0	0	0	AVLLS	0
Write								
Reset	0	0	0	0	0	0	0	0

SMC_PMPROT 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 AVLP	Allow Very-Low-Power Modes Provided the appropriate control bits are set up in PMCTRL, this write-once bit allows the MCU to enter any very-low-power mode (VLPR, VLPW, and VLPS). 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 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 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.

18.3.2 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.

Address: FFFF_80B0h base + 1h offset = FFFF_80B1h

Bit	7	6	5	4	3	2	1	0
Read	LPWUI	RUNM		0	STOPA	STOPM		
Write								
Reset	0	0	0	0	0	0	0	0

SMC_PMCTRL field descriptions

Field	Description
7 LPWUI	Low-Power Wake Up On Interrupt Causes the SMC to exit to normal RUN mode when any active MCU interrupt occurs while in a VLP mode (VLPR, VLPW or VLPS). **NOTE:** If VLPS mode was entered directly from RUN mode, the SMC will always exit back to normal RUN mode regardless of the LPWUI setting. **NOTE:** LPWUI must be modified only while the system is in RUN mode, that is, when PMSTAT=RUN. 0 The system remains in a VLP mode on an interrupt 1 The system exits to Normal RUN mode on an interrupt
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. This field is cleared by hardware on any exit to normal RUN mode. **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. 00 Normal Run mode (RUN) 01 Reserved 10 Very-Low-Power Run mode (VLPR) 11 Reserved
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 or reset occured during the previous stop mode entry sequence, preventing the system from entering that mode. This bit is cleared by hardware at the beginning of any stop mode entry sequence and is set if the sequence was aborted. 0 The previous stop mode entry was successsful. 1 The previous stop mode entry was aborted.
2–0 STOPM	Stop Mode Control When written, controls entry into the selected stop mode when the next STOP instruction is executed with STOPE=1 and WAITE=0 . 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 VLLSx, the in the VLLSCTRL register is used to further select the particular VLLS submode which will be entered. **NOTE:**

Table continues on the next page...

SMC_PMCTRL field descriptions (continued)

Field	Description
	000 Normal Stop (STOP) 001 Reserved 010 Very-Low-Power Stop (VLPS) 011 Reserved 100 Very-Low-Leakage Stop (VLLSx) 101 Reserved 110 Reseved 111 Reserved

18.3.3 VLLS Control register (SMC_VLLSCTRL)

The VLLSCTRL register controls features related to VLLS modes.

NOTE

This register is reset on Chip POR not VLLS and by reset types that trigger Chip POR not VLLS. It is unaffected by reset types that do not trigger Chip POR not VLLS.

Address: FFFF_80B0h base + 2h offset = FFFF_80B2h

Bit	7	6	5	4	3	2	1	0
Read	0		0	RAM2PO	0	VLLSM		
Write								
Reset	0	0	0	0	0	0	1	1

SMC_VLLSCTRL field descriptions

Field	Description
7–6 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 RAM2PO	RAM2 Power Option Controls powering of RAM partition 2 in VLLS2 mode. **NOTE:** See the device's chip configuration details for the size and location of RAM parition 2 0 RAM2 not powered in VLLS2 1 RAM2 powered in VLLS2
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–0 VLLSM	VLLS Mode Control Controls which VLLS sub-mode to enter if STOPM=VLLS. 000 Reserved

Table continues on the next page...

SMC_VLLSCTRL field descriptions (continued)

Field	Description
	001 VLLS1 010 VLLS2 011 VLLS3 100 Reserved 101 Reserved 110 Reserved 111 Reserved

18.3.4 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.

Address: FFFF_80B0h base + 3h offset = FFFF_80B3h

Bit	7	6	5	4	3	2	1	0
Read	0	PMSTAT						
Write								
Reset	0	0	0	0	0	0	0	1

SMC_PMSTAT field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–0 PMSTAT	**NOTE:** When debug is enabled, the PMSTAT will not update to STOP or VLPS 000_0001 Current power mode is RUN 000_0010 Current power mode is STOP 000_0100 Current power mode is VLPR 000_1000 Current power mode is VLPW 001_0000 Current power mode is VLPS 010_0000 Reserved 100_0000 Current power mode is VLLS

18.4 Functional description

18.4.1 Power mode transitions

The following figure shows the power mode state transitions available on the chip. Any reset always brings the MCU back to the normal run state.

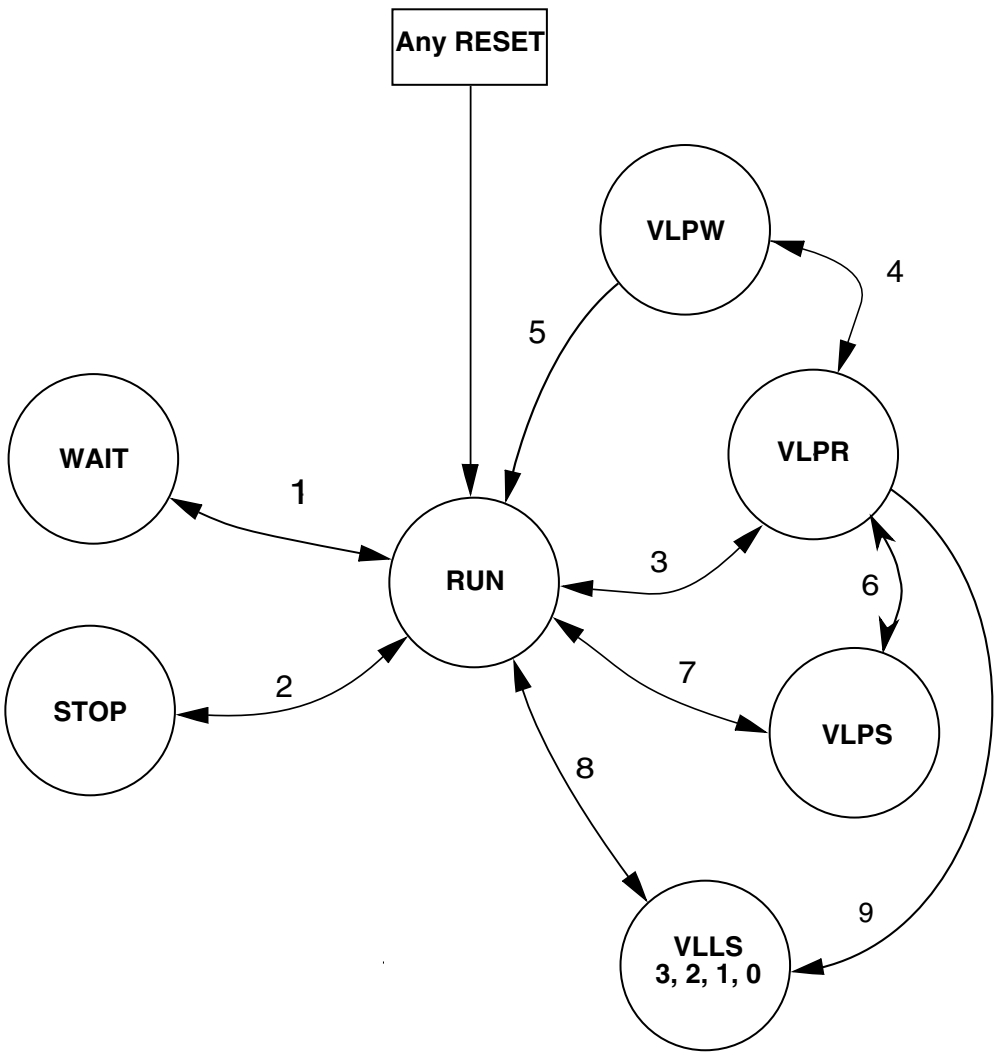


Figure 18-5. Power mode state diagram

The following table defines triggers for the various state transitions shown in the previous figure.

Table 18-7. Power mode transition triggers

Transition #	From	To	Trigger conditions
1	RUN	WAIT	WAITE=1, Issue STOP instruction. See note.[1]
	WAIT	RUN	Interrupt or Reset
2	RUN	STOP	PMCTRL[RUNM]=00, PMCTRL[STOPM]=000 , STOPE=1, WAITE=0, Issue STOP instruction. 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.
	VLPR	RUN	Set PMCTRL[RUNM]=00 or Interrupt with PMCTRL[LPWUI] =1 or Reset.
4	VLPR	VLPW	WAITE=1, Issue STOP instruction. See note.[1]
	VLPW	VLPR	Interrupt with PMCTRL[LPWUI]=0
5	VLPW	RUN	Interrupt with PMCTRL[LPWUI]=1 or Reset
6	VLPR	VLPS	PMCTRL[STOPM]=000 or 010, STOPE=1, WAITE=0, Issue STOP instruction. See note.[1]
	VLPS	VLPR	Interrupt with PMCTRL[LPWUI]=0 **NOTE:** If VLPS was entered directly from RUN, hardware will not allow this transition and will force exit back to RUN
7	RUN	VLPS	PMPROT[AVLP]=1, PMCTRL[STOPM]=010, STOPE=1, WAITE=0, Issue STOP instruction. See note.[1]
	VLPS	RUN	Interrupt with PMCTRL[LPWUI]=1 or Interrupt with PMCTRL[LPWUI]=0 and VLPS mode was entered directly from RUN or Reset

Table continues on the next page...

Table 18-7. Power mode transition triggers (continued)

Transition #	From	To	Trigger conditions
8	RUN	VLLSx	PMPROT[AVLLS]=1, PMCTRL[STOPM]=100, VLLSCTRL[VLLSM]=x (VLLSx), STOPE=1, WAITE=0 Issue STOP instruction.
	VLLSx	RUN	Wakeup from enabled input source or RESET pin
9	VLPR	VLLSx	PMPROT[AVLLS]=1, PMCTRL[STOPM]=100, VLLSCTRL[VLLSM]=x (VLLSx), STOPE=1, WAITE=0 Issue STOP instruction.

1. If debug is enabled, the core clock remains to support debug.

18.4.2 Power mode entry/exit sequencing

When entering or exiting low-power modes, the system must conform to an orderly sequence to manage transitions safely. The SMC manages the system's entry into and exit from all power modes. The following diagram illustrates the connections of the SMC with other system components in the chip that are necessary to sequence the system through all power modes.

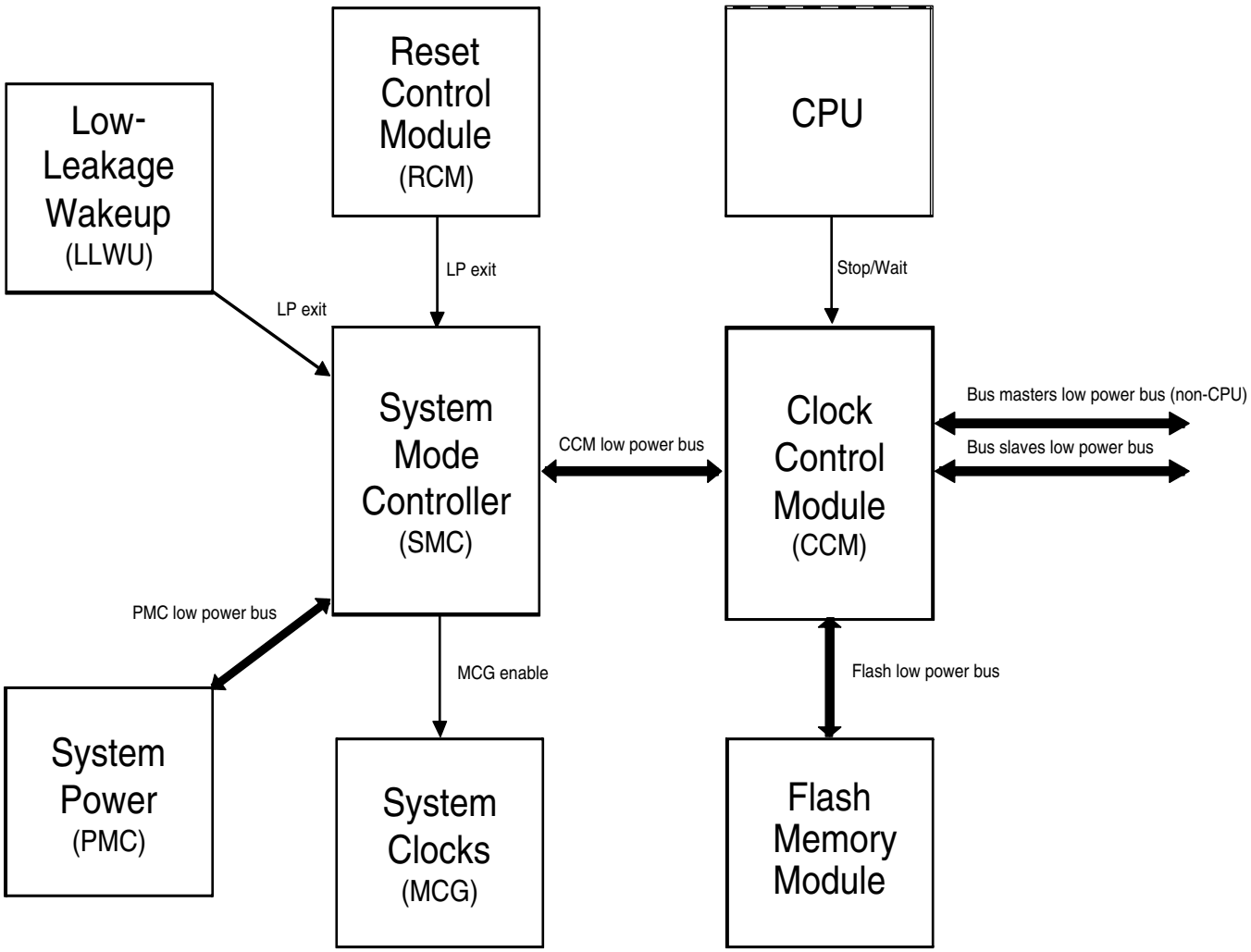


Figure 18-6. Low-power system components and connections

18.4.2.1 Stop mode entry sequence

Entry into a low-power stop mode (Stop, VLPS, VLLSx) is initiated by CPU execution of the 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.

18.4.2.2 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.

18.4.2.3 Aborted stop mode entry

If an interrupt or a reset occurs during a stop entry sequence, the SMC can abort the transition early and return to RUN mode without completely entering the stop mode. An aborted entry is possible only if the reset or interrupt occurs before the PMC begins the transition to stop mode regulation. After this point, the interrupt or reset is ignored until the PMC has completed its transition to stop mode regulation. When an aborted stop mode entry sequence occurs, the SMC's PMCTRL[STOPA] is set to 1.

18.4.2.4 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.

18.4.2.5 Transition from stop modes to Debug mode

The debugger module supports a transition from STOP, WAIT, VLPS, and VLPW back to a Halted state when the debugger has been enabled, that is, ENBDM is 1. As part of this transition, system clocking is re-established and is equivalent to the normal RUN and VLPR mode clocking configuration.

18.4.3 Run modes

The device supports the following run modes:

- Run
- Very Low-Power Run (VLPR)

18.4.3.1 RUN mode

This is the normal operating mode for the device.

This mode is selected after any internal reset including LVD and when the BKGD/MS pin is high after a POR exit or a BDM-initiated force reset. When the ColdFire processor exits reset, it fetches initial 32-bit values for the supervisor stack pointer and program counter from locations 0x00_0000 and 0x00_0004 respectively and user code execution begins.

- The processor reads the start SP (SP_main) from vector-table offset 0x000
- The processor reads the start PC from vector-table offset 0x004
- LR is set to 0xFFFF_FFFF.

To reduce power in this mode, disable the clocks to unused modules using their corresponding clock gating control bits in the SIM's registers.

18.4.3.2 Very-Low Power Run (VLPR) mode

In VLPR mode, the on-chip voltage regulator is put into a stop mode regulation state. In this state, the regulator is designed to supply enough current to the MCU over a reduced frequency. To further reduce power in this mode, disable the clocks to unused modules using their corresponding clock gating control bits in the SIM's registers.

Before entering this mode, the following conditions must be met:

- The MCG must be configured in a mode which is supported during VLPR. See the Power Management details for information about these MCG modes.
- All clock monitors in the MCG must be disabled.
- The maximum frequencies of the system, bus, flash, and core are restricted. See the Power Management details about which frequencies are supported.
- Mode protection must be set to allow VLP modes, that is, PMPROT[AVLP] is 1.
- PMCTRL[RUNM] is set to 10b to enter VLPR.
- Flash programming/erasing is not allowed.

NOTE

Do not change the clock frequency while in VLPR mode, because the regulator is slow responding and cannot manage fast load transitions. In addition, do not modify the clock source in the MCG module, the module clock enables in the SIM, or any clock divider registers.

To reenter Normal Run mode, clear RUNM. The PMSTAT register is a read-only status register that can be used to determine when the system has completed an exit to RUN mode. When PMSTAT=RUN, the system is in run regulation and the MCU can run at full speed in any clock mode. If a higher execution frequency is desired, poll the PMSTAT register until it is set to RUN when returning from VLPR mode.

VLPR mode also provides the option to return to run regulation if any interrupt occurs. Implement this option by setting Low-Power Wakeup On Interrupt (LPWUI) in the PMCTRL register. Any reset always causes an exit from VLPR and returns the device to RUN mode after the MCU exits its reset flow. The RUNM bits are cleared by hardware on any interrupt when LPWUI is set or on any reset.

18.4.3.3 BDM in Run and VLPR Mode

If the MCU is unsecure and BDM mode is enabled, then the MCU can be fully debugged using the BDM in RUN and VLPR modes. If XCSR[ENBDM] = 0, before entering active BDM mode, the host must write XCSR[ENBDM] = 1 before sending a BACKGROUND command.

18.4.4 Wait modes

This device contains two different wait modes:

- Wait
- Very-Low Power Wait (VLPW)

18.4.4.1 WAIT mode

WAIT mode is entered by executing a STOP instruction after configuring the device appropriately. Upon execution of the STOP instruction, the CPU enters a low-power state in which it is not clocked.

The V1 ColdFire core does not differentiate between STOP and WAIT modes. Both are considered STOP mode from the core's perspective. The difference between the two is at the device level. In STOP mode, most peripheral clocks are shut down. In WAIT mode, the global peripheral clocks continue to run and can be enabled or disabled on a per peripheral basis using clock gating control bits in the SIM.

When an interrupt request occurs, the CPU exits WAIT mode and resumes processing in RUN mode, beginning with the stacking operations leading to the interrupt service routine.

A system reset will cause an exit from WAIT mode, returning the device to normal RUN mode.

18.4.4.2 Very-Low-Power Wait (VLPW) mode

VLPW mode is entered by executing a STOP instruction while the MCU is in very low power run (VLPR) mode and configured as per Table 18-7.

In VLPW, the on-chip voltage regulator remains in its stop regulation state. In this state, the regulator is designed to supply enough current to the MCU over a reduced frequency. To further reduce power in this mode, disable the clocks to unused modules by clearing the peripherals' corresponding clock gating control bits in the SIM.

VLPR mode restrictions also apply to VLPW.

VLPW mode provides the option to return to fully-regulated normal RUN mode if any enabled interrupt occurs. This is done by setting PMCTRL[LPWUI]. Wait for the PMSTAT register to set to RUN before increasing the frequency.

If the LPWUI bit is clear, when an interrupt from VLPW occurs, the device returns to VLPR mode to execute the interrupt service routine.

A system reset will cause an exit from VLPW mode, returning the device to normal RUN mode.

18.4.4.3 BDM in Wait and VLPW Mode

If the MCU is unsecure, BDM mode is enabled, and XCSR[ENBDM] is set prior to entering wait then the MCU can support debugging using the BDM.

While the MCU is in wait mode, there are some restrictions on which background debug commands can be used. Only the BACKGROUND command and memory-access-with-status commands are available when the MCU is in wait mode. The memory-access-with-status commands do not allow memory access, but they report an error indicating that the MCU is in either stop or wait mode.

The BACKGROUND command can be used to wake the MCU from wait mode and enter active background mode. After entering halt mode, all background commands are available.

18.4.5 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 may be traded off.

NOTE

All clock monitors must be disabled before entering these low-power modes: Stop, VLPS, VLPR, VLPW, and VLLSx.

The various stop modes are selected by setting the appropriate fields in PMPROT and PMCTRL. The selected mode is entered following the execution of a STOP instruction.

The available stop modes are:

- Normal Stop (STOP)
- Very-Low Power Stop (VLPS)
- Very-Low-Leakage Stop (VLLSx)

18.4.5.1 STOP mode

STOP mode is entered by executing a STOP instruction after configuring the device as per Table 18-7. In STOP mode, the bus and CPU clocks are halted. If the ENBDM is set prior to entering stop, only the peripheral clocks are halted.

The MCG module can be configured to leave the reference clocks running.

NOTE

If neither the WAITE or STOPE bit is set when the CPU executes a STOP instruction, the MCU will not enter either of the stop modes. Executing a STOP instruction under this condition results in either a system reset if the instruction-related reset disable bit in the CPU control register is cleared

(CPUCR[IRD]=0) or an illegal instruction exception if it is set (CPUCR[IRD]=1).

A module capable of providing an asynchronous interrupt to the device takes the device out of STOP mode and returns the device to normal RUN mode. Refer to the device's Power Management chapter for peripheral, I/O, and memory operation in STOP mode. When an interrupt request occurs, the CPU exits STOP mode and resumes processing, beginning with the stacking operations leading to the interrupt service routine.

A system reset will cause an exit from STOP mode, returning the device to normal RUN mode via an MCU reset.

18.4.5.2 Very-Low-Power Stop (VLPS) mode

VLPS mode can be entered in one of two ways:

- Executing a STOP instruction while the MCU is in VLPR mode and STOPM=010 or 000 in the PMCTRL register.
- Executing a STOP instruction while the MCU is in normal RUN mode and STOPM=010 in the PMCTRL register. Note, when VLPS is entered directly from RUN mode, exit to VLPR is disabled by hardware and the system will always exit back to RUN.

In VLPS, the on-chip voltage regulator remains in its stop regulation state as in VLPR.

A module capable of providing an asynchronous interrupt to the device takes the device out of VLPS and returns the device to VLPR mode, provided LPWUI is clear.

If LPWUI is set, the device returns to normal RUN mode upon an interrupt request. PMSTAT must be set to RUN before allowing the system to return to a frequency higher than that allowed in VLPR mode.

A system reset will also cause a VLPS exit, returning the device to normal RUN mode.

18.4.5.3 BDM in Stop and VLPS Modes

If the MCU is unsecure, BDM is enabled, XCSR[ENBDM] is set prior to entering stop, and the current mode is RUN (debug from VLPR is not supported), then the MCU can support debugging using BDM. To support debugging, the CPU clock remains running after the STOP instruction is executed. While the MCU is in stop or VLPS mode, some restrictions affect which background debug commands can be used. Only the BACKGROUND command and memory-access-with-status commands are available

when the MCU is in stop mode. The memory-access-with-status commands do not allow memory access, but they report an error indicating that the MCU is in either stop or wait mode.

The BACKGROUND command can wake the MCU from stop or VLPS mode and cause it to enter active background mode. After entering halt mode, all background commands are available.

NOTE

When the chip is attempting to enter VLPS mode directly from RUN mode while BDM is enabled, the MCU regulator remains in full regulation. In other words, when BDM is enabled, the regulation of the chip retains its previous state when executing STOP.

18.4.5.4 Very-Low-Leakage Stop (VLLSx) modes

This device contains these very low leakage modes:

- VLLS3
- VLLS2
- VLLS1

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.

By executing a STOP instruction while the MCU is in a run mode and configured as per Table 18-7 the MCU will enter the configured VLLS mode.

In VLLS, the on-chip voltage regulator is in its stop-regulation state while most digital logic is powered off.

Before entering VLLS mode, the user should configure the low-leakage wakeup (LLWU) module to enable the desired wakeup sources. The available wakeup sources in VLLS are detailed in the chip configuration details for this device.

After wakeup from VLLS, the device returns to normal RUN mode with a pending LLWU interrupt. In the LLWU interrupt service routine (ISR), the user can poll the LLWU module wakeup flags to determine the source of the wakeup.

When entering VLLS, each I/O pin is latched as configured before executing VLLS. Because all digital logic in the MCU is powered off, all port and peripheral data is lost during VLLS. This information must be restored before the ACKISO bit in the PMC is set.

An asserted $\overline{\text{RESET}}$ pin will cause an exit from any VLLS mode, returning the device to normal RUN mode. When exiting VLLS via the $\overline{\text{RESET}}$ pin, the PIN and WAKEUP bits are set in the SRS0 register of the reset control module (RCM).

18.4.5.5 BDM in VLLSx Modes

No debug is available while the MCU is in VLLSx mode.

Entering a VLLSx mode causes all the BDM and debug controls and settings to be powered off. Therefore, any breakpoints or other debug triggers set prior to entering the VLLSx mode are lost.

Chapter 19
Power Management Controller (PMC)

19.1 Introduction

The power management controller (PMC) contains the internal voltage regulator, power on reset (POR), and low voltage detect system.

19.2 Features

The PMC features include:

- Internal voltage regulator
- Active POR providing brown-out detect
- Low-voltage detect supporting two low-voltage trip points with four warning levels per trip point

19.3 Low-voltage detect (LVD) system

This device includes a system to guard against low-voltage conditions. This protects memory contents and controls MCU system states during supply voltage variations. The system is comprised of a power-on reset (POR) circuit and a LVD circuit with a user-selectable trip voltage: high (V_{LVDH}) or low (V_{LVDL}). The trip voltage is selected by the LVDSC1[LVDV] bits. The LVD is disabled upon entering VLPx and VLLSx modes.

Two flags are available to indicate the status of the low-voltage detect system:

- The low voltage detect flag (LVDF) operates in a level sensitive manner. The LVDF bit is set when the supply voltage falls below the selected trip point (VLVD). The

LVDF bit is cleared by writing one to the LVDACK bit, but only if the internal supply has returned above the trip point; otherwise, the LVDF bit remains set.

- The low voltage warning flag (LVWF) operates in a level sensitive manner. The LVWF bit is set when the supply voltage falls below the selected monitor trip point (VLVW). The LVWF bit is cleared by writing one to the LVWACK bit, but only if the internal supply has returned above the trip point; otherwise, the LVWF bit remains set.

19.3.1 LVD reset operation

By setting the LVDRE bit, the LVD generates a reset upon detection of a low voltage condition. The low voltage detection threshold is determined by the LVDV bits. After an LVD reset occurs, the LVD system holds the MCU in reset until the supply voltage rises above this threshold. The LVD bit in the SRS register is set following an LVD or power-on reset.

19.3.2 LVD interrupt operation

By configuring the LVD circuit for interrupt operation (LVDIE set and LVDRE clear), LVDSC1[LVDF] is set and an LVD interrupt request occurs upon detection of a low voltage condition. The LVDF bit is cleared by writing one to the LVDSC1[LVDACK] bit.

19.3.3 Low-voltage warning (LVW) interrupt operation

The LVD system contains a low-voltage warning flag (LVWF) to indicate that the supply voltage is approaching, but is above, the LVD voltage. The LVW also has an interrupt, which is enabled by setting the LVDSC2[LVWIE] bit. If enabled, an LVW interrupt request occurs when the LVWF is set. LVWF is cleared by writing one to the LVDSC2[LVWACK] bit.

The LVDSC2[LVWV] bits select one of four trip voltages:

- Highest: V_{LVW4}
- Two mid-levels: V_{LVW3} and V_{LVW2}
- Lowest: V_{LVW1}

19.4 I/O retention

When in VLLS modes, the I/O states are held on a wakeup event (with the exception of wakeup by reset event) until the wakeup has been acknowledged via a write to the ACKISO bit. In the case of VLLS exit via a RESET pin, the I/O are released and default to their reset state. In this case, no write to the ACKISO is needed.

19.5 Memory map and register descriptions

PMC register details follow.

NOTE

Different portions of PMC registers are reset only by particular reset types. Each register's description provides details.

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
FFFF_80A0	Low Voltage Detect Status And Control 1 register (PMC_LVDSC1)	8	R/W	10h	19.5.1/335
FFFF_80A1	Low Voltage Detect Status And Control 2 register (PMC_LVDSC2)	8	R/W	00h	19.5.2/337
FFFF_80A2	Regulator Status And Control register (PMC_REGSC)	8	R/W	04h	19.5.3/338

19.5.1 Low Voltage Detect Status And Control 1 register (PMC_LVDSC1)

This register contains status and control bits to support the low voltage detect function. This register should be written during the reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

While the device is in the very low power or low leakage modes, the LVD system is disabled regardless of LVDSC1 settings. To protect systems that must have LVD always on, configure the SMC's power mode protection register (PMPROT) to disallow any very low power or low leakage modes from being enabled.

See the device's data sheet for the exact LVD trip voltages.

NOTE

The LVDV bits are reset solely on a POR Only event. The register's other bits are reset on Chip Reset Not VLLS.

Address: FFFF_80A0h base + 0h offset = FFFF_80A0h

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 bit 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 bit 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
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.
1–0 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 ($V_{LVD} = V_{LVDH}$)

Table continues on the next page...

PMC_LVDSC1 field descriptions (continued)

Field	Description
	10 Reserved 11 Reserved

19.5.2 Low Voltage Detect Status And Control 2 register (PMC_LVDSC2)

This register contains status and control bits to support the low voltage warning function.

While the device is in the very low power or low leakage modes, the LVD system is disabled regardless of LVDSC2 settings.

See the device's data sheet for the exact LVD trip voltages.

NOTE

The LVW trip voltages depend on LVWV and LVDV bits.

NOTE

The LVWV bits are reset solely on a POR Only event. The register's other bits are reset on Chip Reset Not VLLS.

Address: FFFF_80A0h base + 1h offset = FFFF_80A1h

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 bit indicates a low-voltage warning event. LVWF is set when V_{Supply} transitions below the trip point, or after reset and V_{Supply} is already below V_{LVW}. LVWF bit may be 1 after power on reset, therefore, to use LVW interrupt function, before enabling LVWIE, LVWF must be cleared by writing LVWACK first. 0 Low-voltage warning event not detected 1 Low-voltage warning event detected
6 LVWACK	Low-Voltage Warning Acknowledge This write-only bit is used to acknowledge low voltage warning errors. Write 1 to clear LVWF. Reads always return 0.
5 LVWIE	Low-Voltage Warning Interrupt Enable

Table continues on the next page...

PMC_LVDSC2 field descriptions (continued)

Field	Description
	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.
1–0 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 ($V_{LVW} = V_{LVW4}$)

19.5.3 Regulator Status And Control register (PMC_REGSC)

The PMC contains an internal voltage regulator. The voltage regulator design uses a bandgap reference that is also available through a buffer as input to certain internal peripherals, such as the CMP and ADC. The internal regulator provides a status bit (REGONS) indicating the regulator is in run regulation.

NOTE

This register is reset on Chip Reset Not VLLS and by reset types that trigger Chip Reset not VLLS.

Address: FFFF_80A0h base + 2h offset = FFFF_80A2h

Bit	7	6	5	4	3	2	1	0
Read	0		Reserved	BGEN	ACKISO	REGONS	Reserved	BGBE
Write					w1c			
Reset	0	0	0	0	0	1	0	0

PMC_REGSC field descriptions

Field	Description
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
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, 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.

Table continues on the next page...

PMC_REGSC field descriptions (continued)

Field	Description
	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 , and VLLSx modes 1 Bandgap voltage reference is enabled in VLPx , and VLLSx modes
3 ACKISO	Acknowledge Isolation Reading this bit indicates whether certain peripherals and the I/O pads are in a latched state as a result of having been in a VLLS mode. Writing one to this bit when it is set releases the I/O pads and certain peripherals to their normal run mode state. **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 This read-only bit 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

Chapter 20
DMA Controller

20.1 Introduction

This chapter describes the direct memory access (DMA) controller module. It provides an overview of the module and describes in detail its signals and programming model. The latter sections of this chapter describe operations, features, and supported data transfer modes in detail.

Note

The designation *n* is used throughout this section to refer to registers or signals associated with one of the four identical DMA channels: DMA0, DMA1, DMA2, or DMA3.

20.1.1 Overview

The DMA controller module enables fast transfers of data, providing an efficient way to move blocks of data with minimal processor interaction. The DMA module, shown in the following figure, has four channels that allow 8-bit, 16-bit, or 32-bit data transfers. Each channel has a dedicated source address register (SAR*n*), destination address register (DAR*n*), status register (DSR*n*), byte count register (BCR*n*), and control register (DCR*n*). Collectively, the combined program-visible registers associated with each channel define a transfer control descriptor (TCD). All transfers are dual address, moving data from a source memory location to a destination memory location with the module operating as a 32-bit bus master connected to the system bus. The programming model is accessed through a 32-bit connection with the slave peripheral bus. DMA data transfers may be explicitly initiated by software or by peripheral hardware requests.

The following figure is a simplified block diagram of the 4-channel DMA controller.

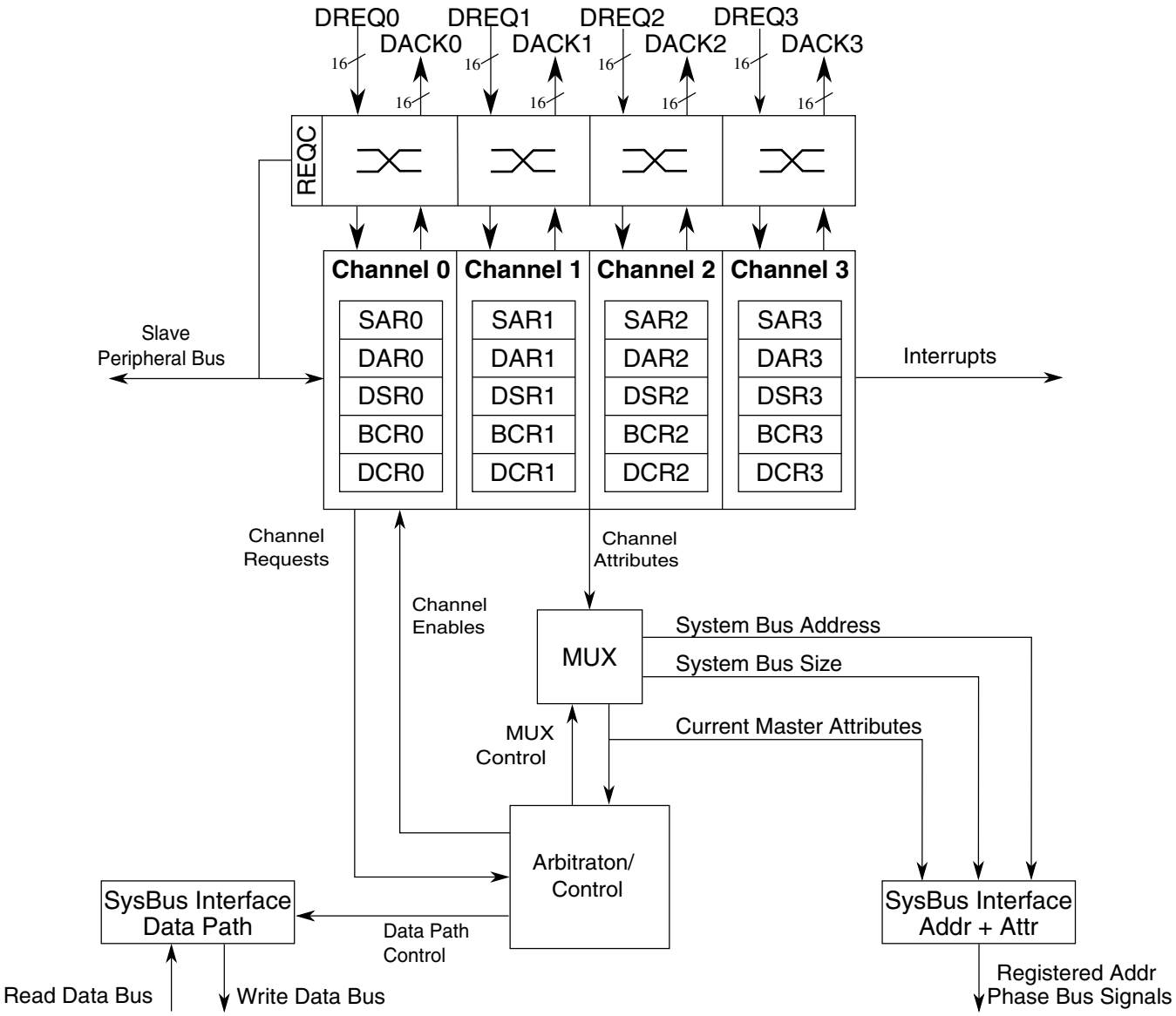


Figure 20-1. 4-Channel DMA Block Diagram

The terms *peripheral request* and *DREQ* refer to a DMA request from one of the on-chip peripherals or package pins. The DMA provides hardware handshake signals: either a DMA acknowledge (DACK) or a done indicator back to the peripheral.For details on the connections associated with DMA request inputs, see the register definition for DMA Request Control (DMAREQC).

20.1.2 Features

The DMA controller module features:

- Four independently programmable DMA controller channels

- Dual-address transfers via 32-bit master connection to the system bus
- Data transfers in 8-, 16-, or 32-bit blocks
- Continuous-mode or cycle-steal transfers from software or peripheral initiation
- One programmable input selected from 16 possible peripheral requests per channel
- Automatic hardware acknowledge/done indicator from each channel
- Independent source and destination address registers
- Optional modulo addressing and automatic updates of source and destination addresses
- Independent transfer sizes for source and destination
- Optional auto-alignment feature for source or destination accesses
- Optional automatic single or double channel linking
- Programming model accessed via 32-bit slave peripheral bus
- Channel arbitration on transfer boundaries using fixed priority scheme

20.2 DMA Transfer Overview

The DMA module can move data within system memory (including memory and peripheral devices) with minimal processor intervention, greatly improving overall system performance. The DMA module consists of four independent, functionally equivalent channels, so references to DMA in this chapter apply to any of the channels. It is not possible to address all four channels at once.

As soon as a channel has been initialized, it may be started by setting DCR*n*[START] or a properly-selected peripheral DMA request, depending on the status of DCR*n*[ERQ]. Each channel can be programmed to select one peripheral request from a set of 16 possible request inputs.

The DMA controller supports dual-address transfers using its bus master connection to the system bus. The DMA channels support transfers up to 32 data bits in size and have the same memory map addressibility as the processor.

- Dual-address transfers—A dual-address transfer consists of a read followed by a write and is initiated by a request using the DCRn[START] bit or by a peripheral DMA request. The read data is temporarily held in the DMA channel hardware until the write operation. Two types of single transfers occur: a read from a source address followed by a write to a destination address. See the following figure.

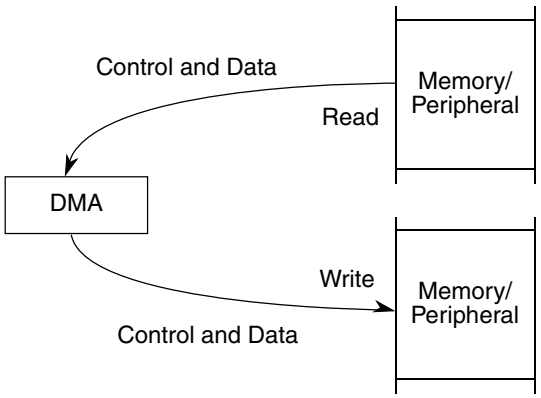


Figure 20-2. Dual-Address Transfer

Any operation involving a DMA channel follows the same three steps:

1. Channel initialization—The transfer control descriptor, contained in the channel registers, is loaded with address pointers, a byte-transfer count, and control information using accesses from the slave peripheral bus.
2. Data transfer—The DMA accepts requests for data transfers. Upon receipt of a request, it provides address and bus control for the transfers via its master connection to the system bus and temporary storage for the read data. The channel performs one or more source read and destination write data transfers.
3. Channel termination—Occurs after the operation is finished successfully or due to an error. The channel indicates the operation status in the channel's DSR, described in the definitions of the DMA Status Registers (DSRn) and Byte Count Registers (BCRn).

20.3 Memory Map and Registers

Descriptions of each register and its bit assignments follow. Modifying DMA control registers during a transfer can result in undefined operation. The following table shows the mapping of DMA controller registers. The DMA programming model is accessed via

the slave peripheral bus. The concatenation of the source and destination address registers, the status and byte count register, and the control register create a 128-bit transfer control descriptor (TCD) that defines the operation of each DMA channel.

DMA memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_E400	DMA Request Control Register (DMA_REQC)	32	R/W	0000_0000h	20.3.1/345
FFFF_E500	Source Address Register (DMA_SAR0)	32	R/W	0000_0000h	20.3.2/349
FFFF_E504	Destination Address Register (DMA_DAR0)	32	R/W	0000_0000h	20.3.3/350
FFFF_E508	DMA Status Register / Byte Count Register (DMA_DSR_BCR0)	32	R/W	0000_0000h	20.3.4/350
FFFF_E50C	DMA Control Register (DMA_DCR0)	32	R/W	0000_0000h	20.3.5/353
FFFF_E510	Source Address Register (DMA_SAR1)	32	R/W	0000_0000h	20.3.2/349
FFFF_E514	Destination Address Register (DMA_DAR1)	32	R/W	0000_0000h	20.3.3/350
FFFF_E518	DMA Status Register / Byte Count Register (DMA_DSR_BCR1)	32	R/W	0000_0000h	20.3.4/350
FFFF_E51C	DMA Control Register (DMA_DCR1)	32	R/W	0000_0000h	20.3.5/353
FFFF_E520	Source Address Register (DMA_SAR2)	32	R/W	0000_0000h	20.3.2/349
FFFF_E524	Destination Address Register (DMA_DAR2)	32	R/W	0000_0000h	20.3.3/350
FFFF_E528	DMA Status Register / Byte Count Register (DMA_DSR_BCR2)	32	R/W	0000_0000h	20.3.4/350
FFFF_E52C	DMA Control Register (DMA_DCR2)	32	R/W	0000_0000h	20.3.5/353
FFFF_E530	Source Address Register (DMA_SAR3)	32	R/W	0000_0000h	20.3.2/349
FFFF_E534	Destination Address Register (DMA_DAR3)	32	R/W	0000_0000h	20.3.3/350
FFFF_E538	DMA Status Register / Byte Count Register (DMA_DSR_BCR3)	32	R/W	0000_0000h	20.3.4/350
FFFF_E53C	DMA Control Register (DMA_DCR3)	32	R/W	0000_0000h	20.3.5/353

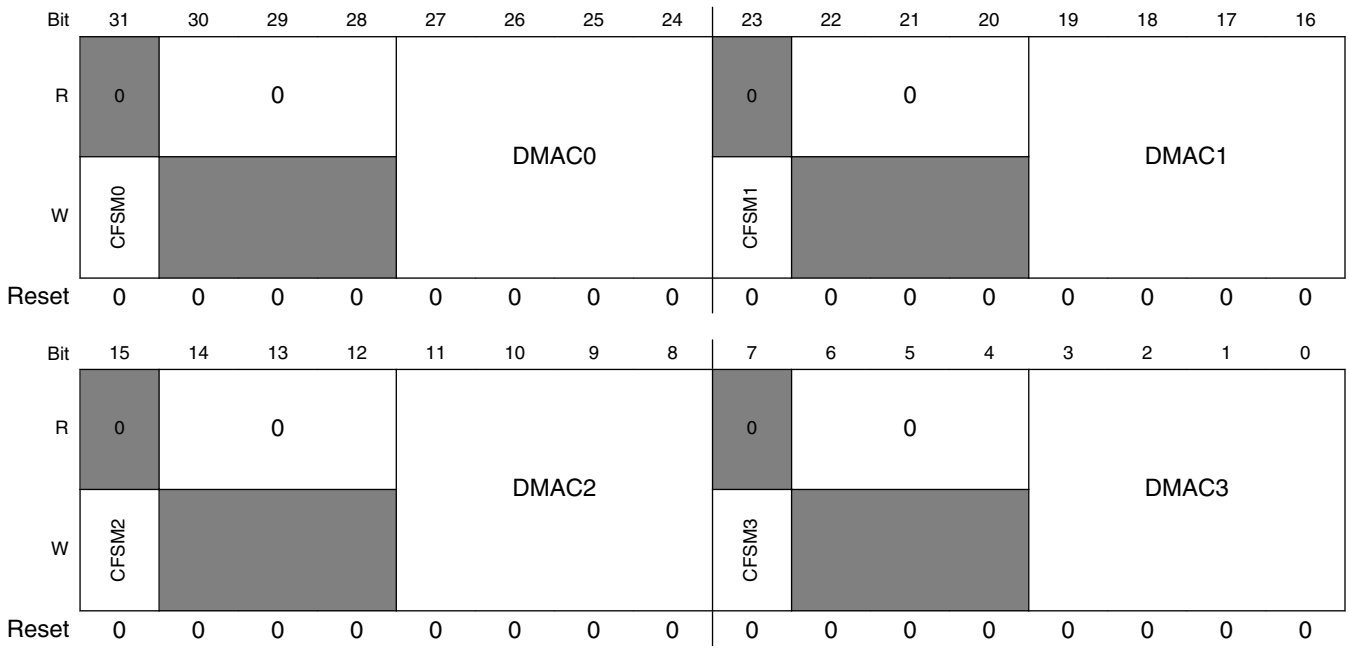
20.3.1 DMA Request Control Register (DMA_REQC)

This register provides a software-controlled connection matrix for DMA requests and acknowledges. Each channel supports 16 possible peripheral requests. The register is programmed to select one peripheral request from the available sources for each channel of the DMA controller. Additionally, the register routes a DMA acknowledge from the channel back to the appropriate peripheral. Writing to this register determines the exact routing of the DMA requests to each of the four channels of the DMA module.

If DCRn[ERQ] is set and the channel is idle, the assertion of the appropriate DREQn signal activates channel n.

The connections of the DMA request sources to the specific channels are device-specific. Refer to the Chip Configuration details for more information.

Address: FFFF_E400h base + 0h offset = FFFF_E400h



DMA_REQC field descriptions

Field	Description
31 CFSM0	Clear state machine control 0 This bit clears the state machine for DMA channel 0. When changing the DMAC0 field to select a different requester, set (write 1) to the CFSM0 bit to clear the channel's state machine. Writing 0 to this bit has no effect. The bit always reads as 0. 0 No effect 1 Clear state machine for DMA channel 0
30–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–24 DMAC0	DMA channel 0 This four-bit field defines the logical connection between the DMA requesters and DMA channel 0. There are sixteen possible requesters per channel and any request from the possible sources can be routed to the DMA channel 0. Effectively, the DMAREQC register provides a software-controlled routing matrix of the DMA request signals to the 4 channels of the DMA module. DMAC0 controls DMA channel 0. The DMA also uses this register to control the broadcasting of acknowledge/done signals back to the selected peripheral to complete the hardware-initiated data transfer. The definition of the 16 possible DMA request sources for each channel is device specific. Refer to the Chip Configuration details for more information. 0000 Select request 0 as the source 0001 Select request 1 as the source 0010 Select request 2 as the source 0011 Select request 3 as the source 0100 Select request 4 as the source 0101 Select request 5 as the source 0110 Select request 6 as the source

Table continues on the next page...

DMA_REQC field descriptions (continued)

Field	Description
	0111 Select request 7 as the source 1000 Select request 8 as the source 1001 Select request 9 as the source 1010 Select request 10 as the source 1011 Select request 11 as the source 1100 Select request 12 as the source 1101 Select request 13 as the source 1110 Select request 14 as the source 1111 Select request 15 as the source
23 CFSM1	Clear state machine control 1 This bit clears the state machine for DMA channel 1. When changing the DMAC1 field to select a different requester, set (write 1) to the CFSM1 bit to clear the channel's state machine. Writing 0 to this bit has no effect. The bit always reads as 0. 0 No effect 1 Clear state machine for DMA channel 1
22–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–16 DMAC1	DMA channel 1 This four-bit field defines the logical connection between the DMA requesters and DMA channel 1. There are sixteen possible requesters per channel and any request from the possible sources can be routed to the DMA channel 1. Effectively, the DMAREQC register provides a software-controlled routing matrix of the DMA request signals to the 4 channels of the DMA module. DMAC1 controls DMA channel 1. The DMA also uses this register to control the broadcasting of acknowledge/done signals back to the selected peripheral to complete the hardware-initiated data transfer. The definition of the 16 possible DMA request sources for each channel is device specific. Refer to the Chip Configuration details for more information. 0000 Select request 0 as the source 0001 Select request 1 as the source 0010 Select request 2 as the source 0011 Select request 3 as the source 0100 Select request 4 as the source 0101 Select request 5 as the source 0110 Select request 6 as the source 0111 Select request 7 as the source 1000 Select request 8 as the source 1001 Select request 9 as the source 1010 Select request 10 as the source 1011 Select request 11 as the source 1100 Select request 12 as the source 1101 Select request 13 as the source 1110 Select request 14 as the source 1111 Select request 15 as the source
15 CFSM2	Clear state machine control 2

Table continues on the next page...

DMA_REQC field descriptions (continued)

Field	Description
	This bit clears the state machine for DMA channel 2. When changing the DMAC2 field to select a different requester, set (write 1) to the CFSM2 bit to clear the channel's state machine. Writing 0 to this bit has no effect. The bit always reads as 0. 0 No effect 1 Clear state machine for DMA channel 2
14–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–8 DMAC2	DMA channel 2 This four-bit field defines the logical connection between the DMA requesters and DMA channel 2. There are sixteen possible requesters per channel and any request from the possible sources can be routed to the DMA channel 2. Effectively, the DMAREQC register provides a software-controlled routing matrix of the DMA request signals to the 4 channels of the DMA module. DMAC2 controls DMA channel 2. The DMA also uses this register to control the broadcasting of acknowledge/done signals back to the selected peripheral to complete the hardware-initiated data transfer. The definition of the 16 possible DMA request sources for each channel is device specific. Refer to the Chip Configuration details for more information. 0000 Select request 0 as the source 0001 Select request 1 as the source 0010 Select request 2 as the source 0011 Select request 3 as the source 0100 Select request 4 as the source 0101 Select request 5 as the source 0110 Select request 6 as the source 0111 Select request 7 as the source 1000 Select request 8 as the source 1001 Select request 9 as the source 1010 Select request 10 as the source 1011 Select request 11 as the source 1100 Select request 12 as the source 1101 Select request 13 as the source 1110 Select request 14 as the source 1111 Select request 15 as the source
7 CFSM3	Clear state machine control 3 This bit clears the state machine for DMA channel 3. When changing the DMAC3 field to select a different requester, set (write 1) to the CFSM3 bit to clear the channel's state machine. Writing 0 to this bit has no effect. The bit always reads as 0. 0 No effect 1 Clear state machine for DMA channel 3
6–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–0 DMAC3	DMA channel 3 This four-bit field defines the logical connection between the DMA requesters and DMA channel 3. There are sixteen possible requesters per channel and any request from the possible sources can be routed to

Table continues on the next page...

DMA_REQC field descriptions (continued)

Field	Description
	the DMA channel 3. Effectively, the DMAREQC register provides a software-controlled routing matrix of the DMA request signals to the 4 channels of the DMA module. DMAC3 controls DMA channel 3. The DMA also uses this register to control the broadcasting of acknowledge/done signals back to the selected peripheral to complete the hardware-initiated data transfer. The definition of the 16 possible DMA request sources for each channel is device specific. Refer to the Chip Configuration details for more information. 0000 Select request 0 as the source 0001 Select request 1 as the source 0010 Select request 2 as the source 0011 Select request 3 as the source 0100 Select request 4 as the source 0101 Select request 5 as the source 0110 Select request 6 as the source 0111 Select request 7 as the source 1000 Select request 8 as the source 1001 Select request 9 as the source 1010 Select request 10 as the source 1011 Select request 11 as the source 1100 Select request 12 as the source 1101 Select request 13 as the source 1110 Select request 14 as the source 1111 Select request 15 as the source

20.3.2 Source Address Register (DMA_SAR*n*)

Address: FFFF_E400h 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	SAR																															
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

DMA_SAR*n* field descriptions

Field	Description
31–0 SAR	Each SAR contains the byte address used by the DMA controller to read data. The SARn is typically aligned on a 0-modulo-ssize boundary—that is, on the natural alignment of the source data. Because the system supports only 24-bit addresses, SARn[31:24] is ignored.

20.3.3 Destination Address Register (DMA_DAR*n*)

Address: FFFF_E400h 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 W	DAR																															
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

DMA_DAR*n* field descriptions

Field	Description
31–0 DAR	Each DAR contains the byte address used by the DMA controller to write data. The DARn is typically aligned on a 0-modulo-dsize boundary—that is, on the natural alignment of the destination data. Because the system supports only 24-bit addresses, DARn[31:24] is ignored.

20.3.4 DMA Status Register / Byte Count Register (DMA_DSR_BCR*n*)

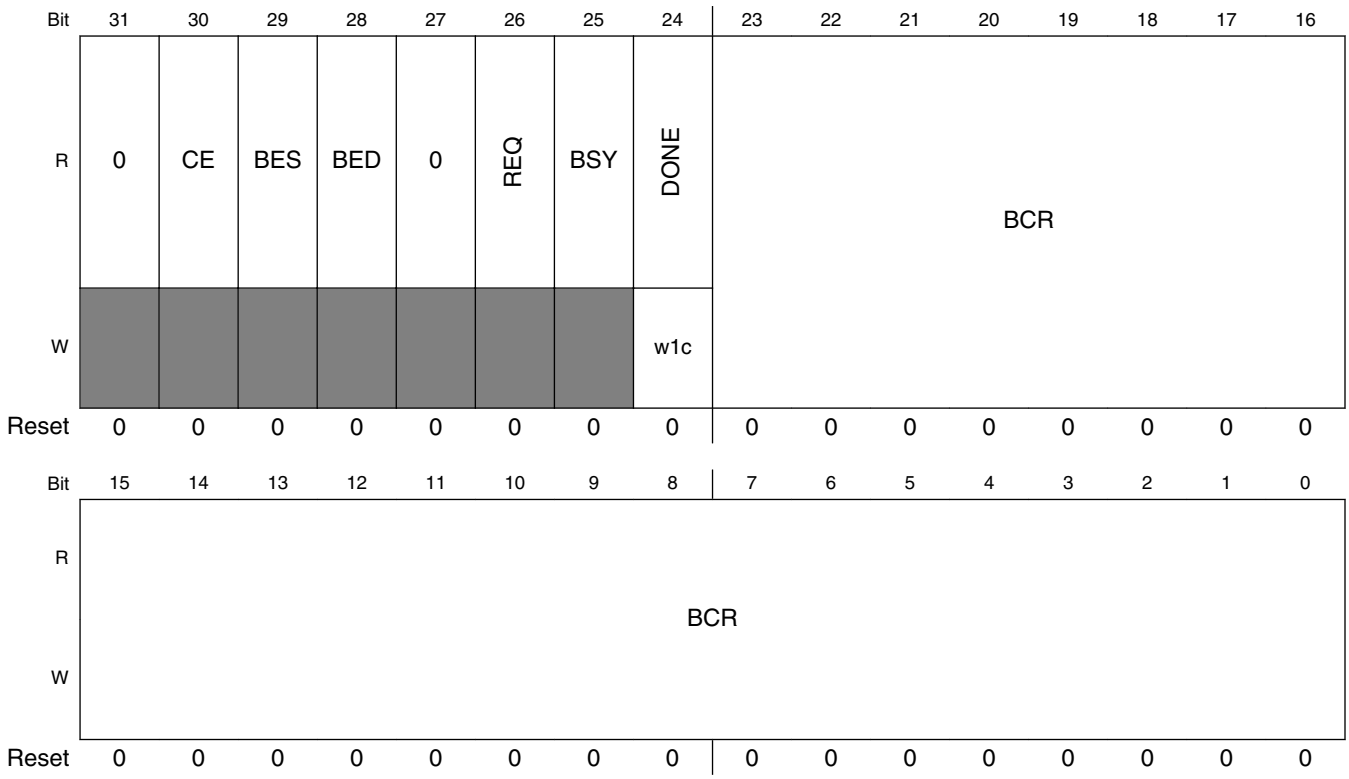
DSR and BCR are two logical registers that occupy one 32-bit address. DSRn occupies bits 31–24, and BCRn occupies bits 23–0. DSRn contains flags indicating the channel status, and BCRn contains the number of bytes yet to be transferred for a given block.

On the successful completion of the write transfer, BCRn decrements by 1, 2, or 4 for 8-bit, 16-bit, or 32-bit accesses, respectively. BCRn is cleared if a 1 is written to DSR[DONE].

In response to an event, the DMA controller writes to the appropriate DSRn bit. Only a write to DSRn[DONE] results in action. DSRn[DONE] is set when the block transfer is complete.

When a transfer sequence is initiated and BCRn[BCR] is not a multiple of 4 or 2 when the DMA is configured for 32-bit or 16-bit transfers, respectively, DSRn[CE] is set and no transfer occurs.

Address: FFFF_E400h base + 108h offset + (16d × i), where i=0d to 3d



DMA_DSR_BCR*n* field descriptions

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 CE	Configuration error Any of the following conditions causes a configuration error: • BCR, SAR, or DAR does not match the requested transfer size. • SSIZE or DSIZE is set to an unsupported value. • BCR equals 0 when the DMA receives a start condition. CE is cleared at hardware reset or by writing a 1 to the DONE bit. 0 No configuration error exists. 1 A configuration error has occurred.
29 BES	Bus error on source BES is cleared at hardware reset or by writing a 1 to the DONE bit. 0 No bus error occurred. 1 The DMA channel terminated with a bus error during the read portion of a transfer.
28 BED	Bus error on destination BED is cleared at hardware reset or by writing a 1 to the DONE bit. 0 No bus error occurred. 1 The DMA channel terminated with a bus error during the write portion of a transfer.

Table continues on the next page...

DMA_DSR_BCR*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 REQ	Request 0 No request is pending or the channel is currently active. Cleared when the channel is selected. 1 The DMA channel has a transfer remaining and the channel is not selected.
25 BSY	Busy 0 DMA channel is inactive. Cleared when the DMA has finished the last transaction. 1 BSY is set the first time the channel is enabled after a transfer is initiated.
24 DONE	Transactions done Set when all DMA controller transactions complete as determined by transfer count, or based on error conditions. When BCR reaches zero, DONE is set when the final transfer completes successfully. DONE can also be used to abort a transfer by resetting the status bits. When a transfer completes, software must clear DONE before reprogramming the DMA. 0 DMA transfer is not yet complete. Writing a 0 has no effect. 1 DMA transfer completed. Writing a 1 to this bit clears all DMA status bits and should be used in an interrupt service routine to clear the DMA interrupt and error bits.
23–0 BCR	This field contains the number of bytes yet to be transferred for a given block.

20.3.5 DMA Control Register (DMA_DCR*n*)

Address: FFFF_E400h base + 10Ch offset + (16d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EINT	ERQ	CS	AA	0			Reserved	Reserved	SINC	SSIZE		DINC	DSIZE		0
W																START
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	SMOD				DMOD				D_REQ	0	LINKCC		LCH1		LCH2	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_DCR*n* field descriptions

Field	Description
31 EINT	Enable interrupt on completion of transfer Determines whether an interrupt is generated by completing a transfer or by the occurrence of an error condition. 0 No interrupt is generated. 1 Interrupt signal is enabled.
30 ERQ	Enable peripheral request **CAUTION:** Be careful: a collision can occur between the START bit and D_REQ when the ERQ bit is 1. 0 Peripheral request is ignored. 1 Enables peripheral request, defined by the appropriate REQC[DMACn] field, to initiate transfer. A software-initiated request (setting the START bit) is always enabled.
29 CS	Cycle steal 0 DMA continuously makes read/write transfers until the BCR decrements to 0. 1 Forces a single read/write transfer per request.
28 AA	Auto-align

Table continues on the next page...

DMA_DCR*n* field descriptions (continued)

Field	Description
	AA and SIZE bits determine whether the source or destination is auto-aligned; that is, transfers are optimized based on the address and size. 0 Auto-align disabled 1 If SSIZE indicates a transfer no smaller than DSIZE, source accesses are auto-aligned; otherwise, destination accesses are auto-aligned. Source alignment takes precedence over destination alignment. If auto-alignment is enabled, the appropriate address register increments, regardless of DINC or SINC.
27–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 Reserved	This field is reserved. **CAUTION:** Must be written as zero; otherwise, undefined behavior results.
23 Reserved	This field is reserved. **CAUTION:** Must be written as zero; otherwise, undefined behavior results.
22 SINC	Source increment Controls whether the source address increments after each successful transfer. 0 No change to SAR after a successful transfer. 1 The SAR increments by 1, 2, 4 as determined by the transfer size.
21–20 SSIZE	Source size Determines the data size of the source bus cycle for the DMA controller. 00 32-bit 01 8-bit 10 16-bit 11 Reserved (generates a configuration error (DSRn[CE]) if incorrectly specified at time of channel activation)
19 DINC	Destination increment Controls whether the destination address increments after each successful transfer. 0 No change to the DAR after a successful transfer. 1 The DAR increments by 1, 2, 4 depending upon the size of the transfer.
18–17 DSIZE	Destination size Determines the data size of the destination bus cycle for the DMA controller. 00 32-bit 01 8-bit 10 16-bit 11 Reserved (generates a configuration error (DSRn[CE]) if incorrectly specified at time of channel activation)
16 START	Start transfer 0 DMA inactive 1 The DMA begins the transfer in accordance to the values in the TCDn. START is cleared automatically after one module clock and always reads as logic 0.

Table continues on the next page...

DMA_DCR*n* field descriptions (continued)

Field	Description
15–12 SMOD	Source address modulo Defines the size of the source data circular buffer used by the DMA Controller. If enabled (SMOD is non-zero), the buffer base address is located on a boundary of the buffer size. The value of this boundary is based upon the initial source address (SAR). The base address should be aligned to a 0-modulo-(circular buffer size) boundary. Misaligned buffers are not possible. The boundary is forced to the value determined by the upper address bits in the field selection. 0000 Buffer disabled 0001 Circular buffer size is 16 bytes 0010 Circular buffer size is 32 bytes 0011 Circular buffer size is 64 bytes 0100 Circular buffer size is 128 bytes 0101 Circular buffer size is 256 bytes 0110 Circular buffer size is 512 bytes 0111 Circular buffer size is 1 KB 1000 Circular buffer size is 2 KB 1001 Circular buffer size is 4 KB 1010 Circular buffer size is 8 KB 1011 Circular buffer size is 16 KB 1100 Circular buffer size is 32 KB 1101 Circular buffer size is 64 KB 1110 Circular buffer size is 128 KB 1111 Circular buffer size is 256 KB
11–8 DMOD	Destination address modulo Defines the size of the destination data circular buffer used by the DMA Controller. If enabled (DMOD value is non-zero), the buffer base address is located on a boundary of the buffer size. The value of this boundary depends on the initial destination address (DAR). The base address should be aligned to a 0-modulo-(circular buffer size) boundary. Misaligned buffers are not possible. The boundary is forced to the value determined by the upper address bits in the field selection. 0000 Buffer disabled 0001 Circular buffer size is 16 bytes 0010 Circular buffer size is 32 bytes 0011 Circular buffer size is 64 bytes 0100 Circular buffer size is 128 bytes 0101 Circular buffer size is 256 bytes 0110 Circular buffer size is 512 bytes 0111 Circular buffer size is 1 KB 1000 Circular buffer size is 2 KB 1001 Circular buffer size is 4 KB 1010 Circular buffer size is 8 KB 1011 Circular buffer size is 16 KB 1100 Circular buffer size is 32 KB 1101 Circular buffer size is 64 KB 1110 Circular buffer size is 128 KB 1111 Circular buffer size is 256 KB
7 D_REQ	Disable request

Table continues on the next page...

DMA_DCR*n* field descriptions (continued)

Field	Description
	DMA hardware automatically clears the corresponding DCRn[ERQ] bit when the byte count register reaches zero. 0 ERQ bit is not affected. 1 ERQ bit is cleared when the BCR is exhausted.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–4 LINKCC	Link channel control Allows DMA channels to have their transfers linked. The current DMA channel triggers a DMA request to the linked channels (LCH1 or LCH2) depending on the condition described by the LINKCC bits. If not in cycle steal mode (DCRn[CS]=0) and LINKCC equals 01 or 10, no link to LCH1 occurs. If LINKCC equals 01, a link to LCH1 is created after each cycle-steal transfer performed by the current DMA channel is completed. As the last cycle-steal is performed and the BCR reaches zero, then the link to LCH1 is closed and a link to LCH2 is created. 00 No channel-to-channel linking 01 Perform a link to channel LCH1 after each cycle-steal transfer followed by a link to LCH2 after the BCR decrements to zero 10 Perform a link to channel LCH1 after each cycle-steal transfer 11 Perform a link to channel LCH1 after the BCR decrements to zero
3–2 LCH1	Link channel 1 Indicates the DMA channel assigned as link channel 1. The link channel number cannot be the same as the currently executing channel, and generates a configuration error if this is attempted (DSRn[CE] is set). 00 DMA Channel 0 01 DMA Channel 1 10 DMA Channel 2 11 DMA Channel 3
1–0 LCH2	Link channel 2 Indicates the DMA channel assigned as link channel 2. The link channel number cannot be the same as the currently executing channel, and generates a configuration error if this is attempted (DSRn[CE] is set). 00 DMA Channel 0 01 DMA Channel 1 10 DMA Channel 2 11 DMA Channel 3

20.4 Functional Description

In the following discussion, the term DMA request implies that DCR*n*[START] is set, or DCRn[ERQ] is set and then followed by assertion of the properly selected DMA peripheral request. The START bit is cleared when the channel is activated.

Before initiating a dual-address access, the DMA module verifies that DCR*n*[SSIZE] and DCR*n*[DSIZE] are consistent with the source and destination addresses. If they are not consistent, the configuration error bit, DSR*n*[CE], is set. If misalignment is detected, no transfer occurs, DSR*n*[CE] is set, and, depending on the DCR configuration, an interrupt event may be issued. If the auto-align bit, DCR*n*[AA], is set, error checking is performed on the appropriate registers.

A read/write transfer sequence reads data from the source address and writes it to the destination address. The number of bytes transferred is the largest of the sizes specified by DCR*n*[SSIZE] and DCR*n*[DSIZE] in the DMA Control Registers (DCRn).

Source and destination address registers (SAR*n* and DAR*n*) can be programmed in the DCR*n* to increment at the completion of a successful transfer.

20.4.1 Transfer Requests (Cycle-Steal and Continuous Modes)

The DMA channel supports software-initiated or peripheral-initiated requests. A request is issued by setting DCR*n*[START] or when the selected peripheral request asserts and DCRn[ERQ] is set. Setting DCR*n*[ERQ] enables recognition of the peripheral DMA requests. Selecting between cycle-steal and continuous modes minimizes bus usage for either type of request.

- Cycle-steal mode (DCR*n*[CS] = 1)—Only one complete transfer from source to destination occurs for each request. If DCRn[ERQ] is set, the request is peripheral initiated. A software-initiated request is enabled by setting DCR*n*[START].
- Continuous mode (DCR*n*[CS] = 0)—After a software-initiated or peripheral request, the DMA continuously transfers data until BCRn reaches zero. The DMA performs the specified number of transfers, then retires the channel.

In either mode, the crossbar switch performs independent arbitration on each slave port after each transaction.

20.4.2 Channel Initialization and Startup

Before a data transfer starts, the channel's transfer control descriptor must be initialized with information describing configuration, request-generation method, and pointers to the data to be moved.

20.4.2.1 Channel Prioritization

The four DMA channels are prioritized based on number, with channel 0 having highest priority and channel 3 having the lowest, that is, channel 0 > channel 1 > channel 2 > channel 3.

Simultaneous peripheral requests activate the channels based on this priority order. Once activated, a channel runs to completion as defined by DCR*n*[CS] and BCR*n*.

20.4.2.2 Programming the DMA Controller Module

CAUTION

During a channel's execution, writes to programming model registers can corrupt the data transfer. The DMA module itself does not have a mechanism to prevent writes to registers during a channel's execution.

General guidelines for programming the DMA are:

- The REQC register is configured to select the peripheral DMA requests and assign them to the individual DMA channels.
- TCD*n* is initialized.
  - SAR*n* is loaded with the source (read) address. If the transfer is from a peripheral device to memory or to another peripheral, the source address is the location of the peripheral data register. If the transfer is from memory to a peripheral device or to memory, the source address is the starting address of the data block. This can be any appropriately aligned address.
  - DAR*n* is initialized with the destination (write) address. If the transfer is from a peripheral device to memory, or from memory to memory, DAR*n* is loaded with the starting address of the data block to be written. If the transfer is from memory to a peripheral device, or from a peripheral device to a peripheral device, DAR*n* is loaded with the address of the peripheral data register. This address can be any appropriately aligned address.

- SAR*n* and DAR*n* change after each data transfer depending on DCR*n*[SSIZE, DSIZE, SINC, DINC, SMOD, DMOD] and the starting addresses. Increment values can be 1, 2, or 4 for 8-bit, 16-bit, or 32-bit transfers, respectively. If the address register is programmed to remain unchanged, the register is not incremented after the data transfer.
- BCR*n*[BCR] must be loaded with the total number of bytes to be transferred. It is decremented by 1, 2, or 4 at the end of each transfer, depending on the transfer size. DSR*n*[DONE] must be cleared for channel startup.

- After the channel has been initialized, it may be started by setting DCR*n*[START] or a properly selected peripheral DMA request, depending on the status of DCR*n*[ERQ]. For a software-initiated transfer, the channel can be started by setting DCR*n*[START] as part of a single 32-bit write to the last 32 bits of the TCD*n*; that is, it is not required to write the DCR*n* with START cleared and then perform a second write to explicitly set START.
- Programming the channel for a software-initiated request causes the channel to request the system bus and start transferring data immediately. If the channel is programmed for peripheral-initiated request, a properly selected peripheral DMA request must be asserted before the channel begins the system bus transfers.
- The hardware can automatically clear DCR*n*[ERQ], disabling the peripheral request, when BCR*n* reaches zero by setting DCR*n*[D_REQ].
- Changes to DCR*n* are effective immediately while the channel is active. To avoid problems with changing a DMA channel setup, write a one to DSR*n*[DONE] to stop the DMA channel.

20.4.3 Dual-Address Data Transfer Mode

Each channel supports dual-address transfers. Dual-address transfers consist of a source data read and a destination data write. The DMA controller module begins a dual-address transfer sequence after a DMA request. If no error condition exists, DSR*n*[REQ] is set.

- Dual-address read—The DMA controller drives the SAR*n* value onto the system address bus. If DCR*n*[SINC] is set, the SAR*n* increments by the appropriate number of bytes upon a successful read cycle. When the appropriate number of read cycles complete (multiple reads if the destination size is larger than the source), the DMA initiates the write portion of the transfer.

If a termination error occurs, DSR*n*[BES, DONE] are set and DMA transactions stop.

- Dual-address write—The DMA controller drives the DAR*n* value onto the system address bus. When the appropriate number of write cycles complete (multiple writes if the source size is larger than the destination), DAR*n* increments by the appropriate number of bytes if DCRn[DINC] is set. BCRn decrements by the appropriate number of bytes. DSR*n*[DONE] is set when BCR*n* reaches zero. If the BCR*n* is greater than zero, another read/write transfer is initiated if continuous mode is enabled (DCRn[CS] = 0).

  If a termination error occurs, DSR*n*[BED, DONE] are set and DMA transactions stop.

20.4.4 Advanced Data Transfer Controls: Auto-Alignment

Typically, auto-alignment for DMA transfers applies for transfers of large blocks of data. As a result, it does not apply for peripheral-initiated cycle-steal transfers.

Auto-alignment allows block transfers to occur at the optimal size based on the address, byte count, and programmed size. To use this feature, DCR*n*[AA] must be set. The source is auto-aligned if DCR*n*[SSIZE] indicates a transfer size larger than DCR*n*[DSIZE]. Source alignment takes precedence over the destination when the source and destination sizes are equal. Otherwise, the destination is auto-aligned. The address register chosen for alignment increments regardless of the increment value. Configuration error checking is performed on registers not chosen for alignment.

If BCR*n* is greater than 16, the address determines transfer size. Transfers of 8 bits, 16 bits, or 32 bits are transferred until the address is aligned to the programmed size boundary, at which time accesses begin using the programmed size. If BCR*n* is less than 16 at the start of a transfer, the number of bytes remaining dictates transfer size.

Consider this example:

- AA equals 1.
- SAR*n* equals 0x(00)80_0001.
- BCR*n* equals 0x00_00F0.
- SSIZE equals 00 (32 bits).
- DSIZE equals 01 (8 bits).

Because SSIZE > DSIZE, the source is auto-aligned. Error checking is performed on destination registers. The access sequence is as follows:

1. Read 1 byte from 0x(00)80_0001, increment SAR*n*, write 1 byte (using DAR*n*).
2. Read 2 bytes from 0x(00)80_0002, increment SAR*n*, write 2 bytes.

3. Read 4 bytes from 0x(00)80_0004, increment SAR*n*, write 4 bytes.
4. Repeat 4-byte operations until SAR*n* equals 0x(00)80_00F0.
5. Read byte from 0x(00)80_00F0, increment SAR*n*, write byte.

If DSIZE is another size, data writes are optimized to write the largest size allowed based on the address, but not exceeding the configured size.

20.4.5 Termination

An unsuccessful transfer can terminate for one of the following reasons:

- Error conditions—When the DMA encounters a read or write cycle that terminates with an error condition, DSR*n*[BES] is set for a read and DSR*n*[BED] is set for a write before the transfer is halted. If the error occurred in a write cycle, data in the internal holding registers is lost.
- Interrupts—If DCR*n*[EINT] is set, the DMA drives the appropriate interrupt request signal. The processor can read DSR*n* to determine whether the transfer terminated successfully or with an error. DSR*n*[DONE] is then written with a one to clear the interrupt, the DONE, and error status bits.

Chapter 21
Multipurpose Clock Generator (MCG)

21.1 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 of the FLL or PLL output clocks, or either of the internal or external reference clocks as a source for the MCU system clock. The MCG operates in conjuction with a crystal oscillator, which allows an external crystal, ceramic resonator, or another external clock source to produce the external reference clock.

21.1.1 Features

Key features of the MCG module are:

- Frequency-locked loop (FLL):
  - Digitally-controlled oscillator (DCO)
  - DCO frequency range is programmable for up to four different frequency ranges.
  - Option to program and maximize DCO output frequency for a low frequency external reference clock source.
  - Option to prevent FLL from resetting its current locked frequency when switching clock modes if FLL reference frequency is not changed.
  - Internal or external reference clock can be used as the FLL source.
  - Can be used as a clock source for other on-chip peripherals.
- Phase-locked loop (PLL):

- Voltage-controlled oscillator (VCO)
- External reference clock is used as the PLL source.
- Modulo VCO frequency divider
- Phase/Frequency detector
- Integrated loop filter
- Can be used as a clock source for other on-chip peripherals.

- Internal reference clock generator:
  - Slow clock with nine trim bits for accuracy
  - Fast clock with four trim bits
  - Can be used as source clock for the FLL. In FEI mode, only the slow Internal Reference Clock (IRC) can be used as the FLL source.
  - Either the slow or the fast clock can be selected as the clock source for the MCU.
  - Can be used as a clock source for other on-chip peripherals.
- 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 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 Background Debug Controller Clock (MCGBDCCLK) is provided as a clock source for the Background Debug Controller (BDC)

- MCG PLL Clock (MCGPLLCLK) is provided as a clock source for other on-chip peripherals
- MCG FLL Clock (MCGFLLCLK) is provided as a clock source for other on-chip peripherals
- MCG Fixed Frequency Clock (MCGFFCLK) is provided as a clock source for other on-chip peripherals
- MCG Internal Reference Clock (MCGIRCLK) is provided as a clock source for other on-chip peripherals

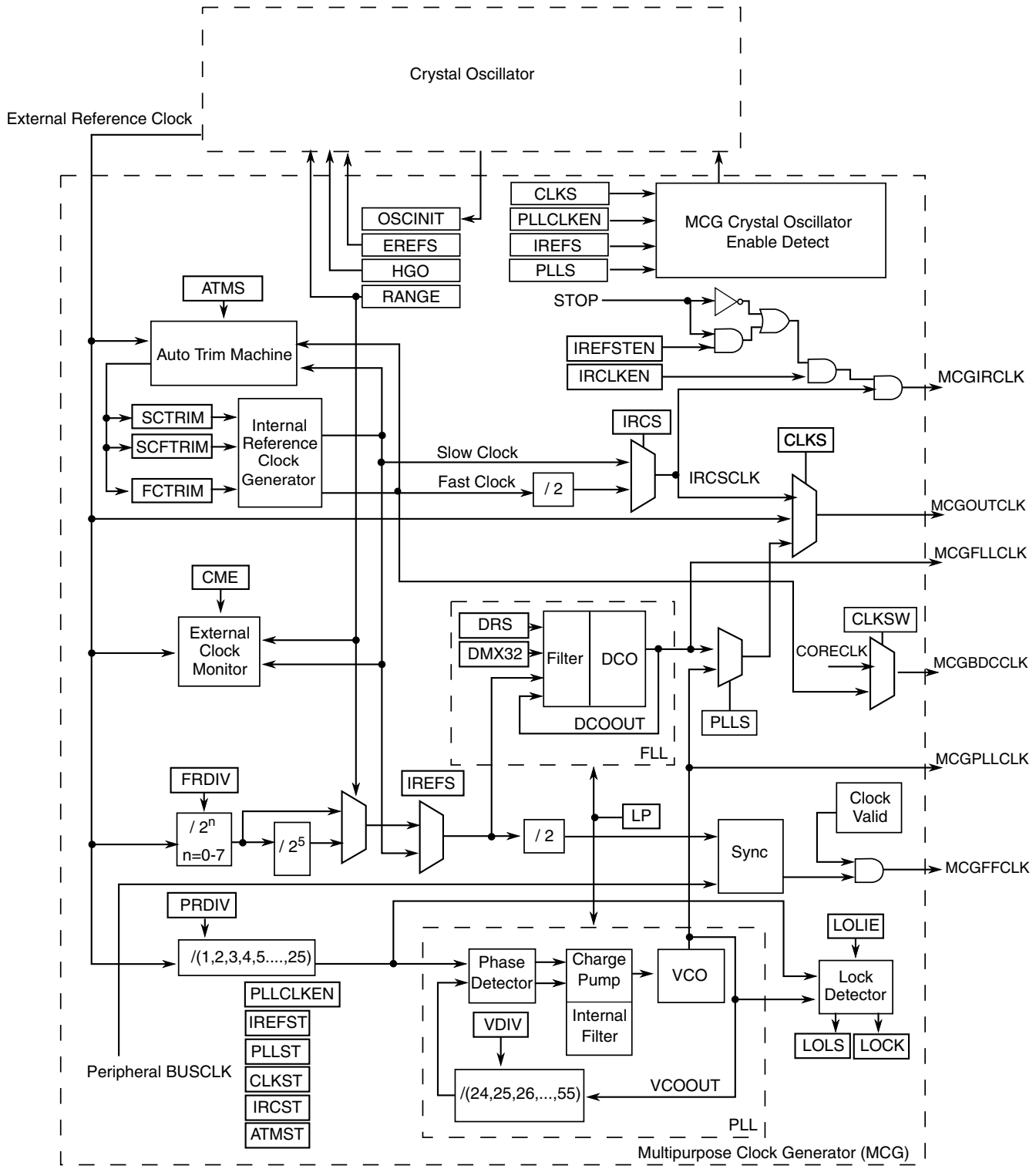


Figure 21-1. Multipurpose Clock Generator (MCG) block diagram

21.1.2 Modes of Operation

The MCG has the following modes of operation: FEI, FEE, FBI, FBE, PBE, PEE, BLPI, BLPE, and Stop. For details, see MCG modes of operation.

21.2 External Signal Description

There are no MCG signals that connect off chip.

21.3 Memory Map/Register Definition

This section includes the memory map and register definition.

MCG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_81E0	MCG Control 1 Register (MCG_C1)	8	R/W	04h	21.3.1/367
FFFF_81E1	MCG Control 2 Register (MCG_C2)	8	R/W	See section	21.3.2/369
FFFF_81E2	MCG Control 3 Register (MCG_C3)	8	R/W	Undefined	21.3.3/370
FFFF_81E3	MCG Control 4 Register (MCG_C4)	8	R/W	Undefined	21.3.4/370
FFFF_81E4	MCG Control 5 Register (MCG_C5)	8	R/W	00h	21.3.5/372
FFFF_81E5	MCG Control 6 Register (MCG_C6)	8	R/W	00h	21.3.6/373
FFFF_81E6	MCG Status Register (MCG_S)	8	R	10h	21.3.7/374
FFFF_81E8	MCG Auto Trim Control Register (MCG_ATC)	8	R/W	00h	21.3.8/376
FFFF_81EA	MCG Auto Trim Compare Value High Register (MCG_ATCVH)	8	R/W	00h	21.3.9/376
FFFF_81EB	MCG Auto Trim Compare Value Low Register (MCG_ATCVL)	8	R/W	00h	21.3.10/377

21.3.1 MCG Control 1 Register (MCG_C1)

Address: FFFF_81E0h base + 0h offset = FFFF_81E0h

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 , Divide Factor is 1; for all other RANGE values, Divide Factor is 32. 001 If RANGE = 0 , Divide Factor is 2; for all other RANGE values, Divide Factor is 64. 010 If RANGE = 0 , Divide Factor is 4; for all other RANGE values, Divide Factor is 128. 011 If RANGE = 0 , Divide Factor is 8; for all other RANGE values, Divide Factor is 256. 100 If RANGE = 0 , Divide Factor is 16; for all other RANGE values, Divide Factor is 512. 101 If RANGE = 0 , Divide Factor is 32; for all other RANGE values, Divide Factor is 1024. 110 If RANGE = 0 , Divide Factor is 64; for all other RANGE values, Divide Factor is Reserved . 111 If RANGE = 0 , Divide Factor is 128; for all other RANGE values, Divide Factor is Reserved .
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. 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.

21.3.2 MCG Control 2 Register (MCG_C2)

Address: FFFF_81E0h base + 1h offset = FFFF_81E1h

Bit	7	6	5	4	3	2	1	0
Read	0	0	RANGE		HGO	EREFS	LP	IRCS
Write								
Reset	0	0	0	0	0	0	0	0

MCG_C2 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.
5–4 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.
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) unless BDM is active.
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.

21.3.3 MCG Control 3 Register (MCG_C3)

Address: FFFF_81E0h base + 2h offset = FFFF_81E2h

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
7–0 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.

21.3.4 MCG Control 4 Register (MCG_C4)

NOTE

Reset values for DRST and DMX32 bits are 0.

Address: FFFF_81E0h base + 3h offset = FFFF_81E3h

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.
- A value for FCTRIM is loaded during reset from a factory programmed location. 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.

Table continues on the next page...

MCG_C4 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>The following table identifies settings for the DCO frequency range.

NOTE: The system clocks derived from this source should not exceed their specified maximums.

<table>
<tr><th>DRST_DRS</th><th>DMX32</th><th>Reference Range</th><th>FLL Factor</th><th>DCO Range</th></tr>
<tr><td rowspan="2">00</td><td>0</td><td>31.25–39.0625 kHz</td><td>640</td><td>20–25 MHz</td></tr>
<tr><td>1</td><td>32.768 kHz</td><td>732</td><td>24 MHz</td></tr>
<tr><td rowspan="2">01</td><td>0</td><td>31.25–39.0625 kHz</td><td>1280</td><td>40–50 MHz</td></tr>
<tr><td>1</td><td>32.768 kHz</td><td>1464</td><td>48 MHz</td></tr>
<tr><td rowspan="2">10</td><td>0</td><td>31.25–39.0625 kHz</td><td>1920</td><td>60–75 MHz</td></tr>
<tr><td>1</td><td>32.768 kHz</td><td>2197</td><td>72 MHz</td></tr>
<tr><td rowspan="2">11</td><td>0</td><td>31.25–39.0625 kHz</td><td>2560</td><td>80–100 MHz</td></tr>
<tr><td>1</td><td>32.768 kHz</td><td>2929</td><td>96 MHz</td></tr>
</table>

0 DCO has a default range of 25%.
1 DCO is fine-tuned for maximum frequency with 32.768 kHz reference.</td></tr>
<tr><td>6–5
DRST_DRS</td><td>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).
01 Encoding 1 — Mid range.
10 Encoding 2 — Mid-high range.
11 Encoding 3 — High range.</td></tr>
<tr><td>4–1
FCTRIM</td><td>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.</td></tr>
<tr><td>0
SCFTRIM</td><td>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.</td></tr>
</table>

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 .

21.3.5 MCG Control 5 Register (MCG_C5)

Address: FFFF_81E0h base + 4h offset = FFFF_81E4h

Bit	7	6	5	4	3	2	1	0
Read	0	PLLCLKEN	PLLSTEN	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 the 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 the range of 2 - 4 MHz 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 bit should be checked to make sure it is set. 0 MCGPLLCLK is inactive. 1 MCGPLLCLK is active.
5 PLLSTEN	PLL Stop Enable 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 is disabled in any of the Stop modes. 1 MCGPLLCLK is enabled if system is in Normal Stop mode.
4–0 PRDIV	PLL External Reference Divider Selects the amount to divide down the external reference clock for the PLL. The resulting frequency must be in the range of 2 MHz to 4 MHz. After the PLL is enabled (by setting either PLLCLKEN or PLLS), the PRDIV value must not be changed when LOCK is zero.

Table 21-7. PLL External Reference Divide Factor

PRDIV	Divide Factor		PRDIV	Divide Factor		PRDIV	Divide Factor		PRDIV	Divide Factor
00000	1		01000	9		10000	17		11000	25
00001	2		01001	10		10001	18		11001	Reserved
00010	3		01010	11		10010	19		11010	Reserved
00011	4		01011	12		10011	20		11011	Reserved

Table continues on the next page...

MCG_C5 field descriptions (continued)

Field	Description
	Table 21-7. PLL External Reference Divide Factor (continued)

00100	5		01100	13		10100	21		11100	Reserved
00101	6		01101	14		10101	22		11101	Reserved
00110	7		01110	15		10110	23		11110	Reserved
00111	8		01111	16		10111	24		11111	Reserved

21.3.6 MCG Control 6 Register (MCG_C6)

Address: FFFF_81E0h base + 5h offset = FFFF_81E5h

Bit	7	6	5	4	3	2	1	0
Read Write	LOLIE	PLLS	CME	VDIV				
Reset	0	0	0	0	0	0	0	0

MCG_C6 field descriptions

Field	Description
7 LOLIE	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 is set. 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 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 need to be programmed to the correct divider to generate a PLL reference clock in the range of 2–4 MHz prior to setting the PLLS bit).
5 CME	Clock Monitor Enable Determines if a reset request is made following a loss of external clock indication. The CME bit should 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 CME bit is set to a logic 1, the value of the RANGE bits in the C2 register should not be changed. CME bit should be set to a logic 0 before the MCG enters any Stop mode. Otherwise, a reset request may occur when in Stop mode. CME should also be set to a logic 0 before entering VLPR or VLPW power modes if the MCG is in BLPE mode.

Table continues on the next page...

MCG_C6 field descriptions (continued)

Field	Description
	0 External clock monitor is disabled. 1 Generate a reset request on loss of external clock.
4–0 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 21-9. PLL VCO Divide Factor

VDIV	Multiply Factor		VDIV	Multiply Factor		VDIV	Multiply Factor		VDIV	Multiply Factor
00000	24		01000	32		10000	40		11000	48
00001	25		01001	33		10001	41		11001	49
00010	26		01010	34		10010	42		11010	50
00011	27		01011	35		10011	43		11011	51
00100	28		01100	36		10100	44		11100	52
00101	29		01101	37		10101	45		11101	53
00110	30		01110	38		10110	46		11110	54
00111	31		01111	39		10111	47		11111	55

21.3.7 MCG Status Register (MCG_S)

Address: FFFF_81E0h base + 6h offset = FFFF_81E6h

Bit	7	6	5	4	3	2	1	0
Read	LOLS	LOCK0	PLLST	IREFST	CLKST		OSCINIT	IRCST
Write								
Reset	0	0	0	1	0	0	0	0

MCG_S field descriptions

Field	Description
7 LOLS	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. 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 was last cleared. 1 PLL has lost lock since LOLS was last cleared.

Table continues on the next page...

MCG_S field descriptions (continued)

Field	Description
6 LOCK0	Lock Status This bit indicates whether the PLL has acquired lock. Lock detection is only enabled when the PLL is enabled (either through clock mode selection or PLLCLKEN0=1 setting). While the PLL clock is locking to the desired frequency, the MCG PLL clock (MCGPLLCLK) will be gated off until the LOCK bit gets asserted. If the lock status bit is set, changing the value of the PRDIV [4:0] bits in the C5 register or the VDIV0[4:0] bits in the C6 register causes the lock status bit to clear and stay cleared until the PLL has reacquired lock. Entry into VLPS, or regular Stop with PLLSTEN=0 also causes the lock status bit to clear and stay cleared until the Stop mode is exited and the PLL has reacquired lock. Any time the PLL is enabled and the LOCK0 bit is cleared, the MCGPLLCLK will be gated off until the LOCK0 bit is asserted again. 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.
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 OSCINIT	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).

21.3.8 MCG Auto Trim Control Register (MCG_ATC)

Address: FFFF_81E0h base + 8h offset = FFFF_81E8h

Bit	7	6	5	4	3	2	1	0
Read	ATME	ATMS	ATMF	0				
Write								
Reset	0	0	0	0	0	0	0	0

MCG_ATC 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 ATC registers or entering Stop mode aborts the auto trim operation and clears this bit. 0 Auto Trim Machine disabled. 1 Auto Trim Machine enabled.
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 ATC 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.
4–0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

21.3.9 MCG Auto Trim Compare Value High Register (MCG_ATCVH)

Address: FFFF_81E0h base + Ah offset = FFFF_81EAh

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
7–0 ATCVH	ATM Compare Value High Values are used by Auto Trim Machine to compare and adjust Internal Reference trim values during ATM SAR conversion.

21.3.10 MCG Auto Trim Compare Value Low Register (MCG_ATCVL)

Address: FFFF_81E0h base + Bh offset = FFFF_81EBh

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
7–0 ATCVL	ATM Compare Value Low Values are used by Auto Trim Machine to compare and adjust Internal Reference trim values during ATM SAR conversion.

21.4 Functional Description

21.4.1 MCG mode state diagram

The nine states of the MCG are shown in the following figure and are described in Table 21-14. The arrows indicate the permitted MCG mode transitions.

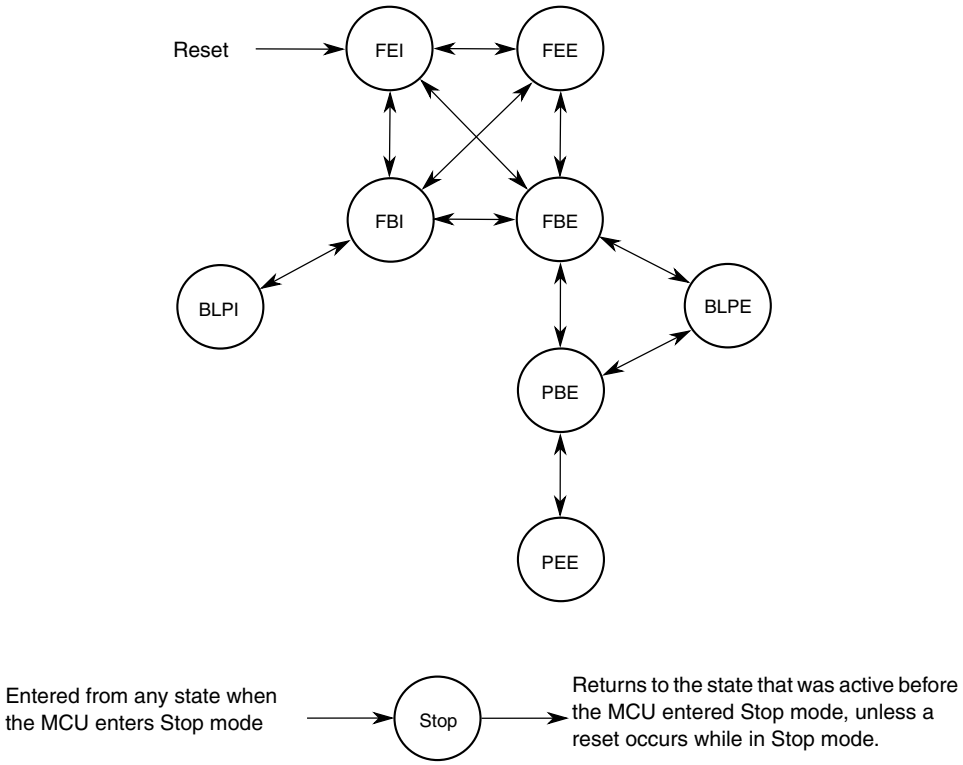


Figure 21-12. MCG mode state diagram

NOTE

- During exits from 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 C5[PLLSTEN]=0, the MCG will reset to PBE clock mode and C1[CLKS] and S[CLKST] will automatically be set to 2’b10.

21.4.1.1 MCG modes of operation

The MCG operates in one of the following modes.

Note

The MCG restricts transitions between modes. For the permitted transitions, see Figure 21-12.

Table 21-14. 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: • C1[CLKS] bits are written to 00 • C1[IREFS] bit is written to 1 • C6[PLLS] bit is written to 0 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: • C1[CLKS] bits are written to 00 • C1[IREFS] bit is written to 0 • C1[FRDIV] must be written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz • C6[PLLS] bit is written to 0 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: • C1[CLKS] bits are written to 01 • C1[IREFS] bit is written to 1 • C6[PLLS] is written to 0 • C2[LP] is written to 0 In FBI mode, the MCGOUTCLK is derived either from the slow (32 kHz IRC) or fast (2 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.

Table continues on the next page...

Table 21-14. MCG modes of operation (continued)

Mode	Description
FLL Bypassed External (FBE)	FLL bypassed external (FBE) mode is entered when all the following conditions occur: • C1[CLKS] bits are written to 10 • C1[IREFS] bit is written to 0 • C1[FRDIV] must be written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz. • C6[PLLS] bit is written to 0 • C2[LP] is written to 0 In FBE mode, the MCGOUTCLK is derived from the external reference clock. The FLL is operational but its output is not used. This mode is useful to allow the FLL to acquire its target frequency while the MCGOUTCLK is driven from the external reference clock. The FLL clock (DCOCLK) is controlled by the external reference clock, and the DCO clock frequency locks to a multiplication factor, as selected by C4[DRST_DRS] and C4[DMX32] bits, times the divided external reference frequency. See the C4[DMX32] bit description for more details. In FBI mode the PLL is disabled in a low-power state unless C5[PLLCLKEN] is set.
PLL Engaged External (PEE)	PLL Engaged External (PEE) mode is entered when all the following conditions occur: • C1[CLKS] bits are written to 00 • C1[IREFS] bit is written to 0 • C6[PLLS] bit is written to 1 In PEE mode, the MCGOUTCLK is derived from the PLL clock, which is controlled by the external reference clock. The PLL clock frequency locks to a multiplication factor, as specified by C6[VDIV], times the external reference frequency, as specified by C5[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: • C1[CLKS] bits are written to 10 • C1[IREFS] bit is written to 0 • C6[PLLS] bit is written to 1 • C2[LP] bit is written to 0 In PBE mode, MCGOUTCLK is derived from the 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: • C1[CLKS] bits are written to 01 • C1[IREFS] bit is written to 1 • C6[PLLS] bit is written to 0 • C2[LP] bit is written to 1 In BLPI mode, MCGOUTCLK is derived from the internal 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 21-14. MCG modes of operation (continued)

Mode	Description
Bypassed Low Power External (BLPE)	Bypassed Low Power External (BLPE) mode is entered when all the following conditions occur: • C1[CLKS] bits are written to 10 • C1[IREFS] bit is written to 0 • C2[LP] bit is written to 1 In BLPE mode, MCGOUTCLK is derived from the external reference clock. The FLL is disabled and PLL is disabled even if the C5[PLLCLKEN] is set to 1.
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:** • When entering Low Power Stop mode (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'b10 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. 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.

21.4.1.2 MCG mode switching

The C1[IREFS] bit can be changed at any time, but the actual switch to the newly selected reference clocks is shown by the S[IREFST] bit. When switching between engaged internal and engaged external modes, the FLL will begin locking again after the switch is completed.

The C1[CLKS] bits can also be changed at any time, but the actual switch to the newly selected clock is shown by the S[CLKST] bits. If the newly selected clock is not available, the previous clock will remain selected.

The C4[DRST_DRS] write bits can be changed at any time except when C2[LP] bit is 1. If the C4[DRST_DRS] write bits are changed while in FLL engaged internal (FEI) or FLL engaged external (FEE), the MCGOUTCLK will switch to the new selected DCO range within three clocks of the selected DCO clock. After switching to the new DCO, the FLL remains unlocked for several reference cycles. DCO startup time is equal to the FLL acquisition time. After the selected DCO startup time is over, the FLL is locked. The completion of the switch is shown by the C4[DRST_DRS] read bits.

21.4.2 Low Power Bit Usage

The C2[LP] bit is provided to allow the FLL or PLL to be disabled and thus conserve power when these systems are not being used. The C4[DRST_DRS] can not be written while C2[LP] bit is 1. However, in some applications, it may be desirable to enable the FLL or PLL and allow it to lock for maximum accuracy before switching to an engaged mode. Do this by writing C2[LP] to 0.

21.4.3 MCG Internal Reference Clocks

This module supports two internal reference clocks with nominal frequencies of 32 kHz (slow IRC) and 4 MHz (fast IRC).

21.4.3.1 MCG Internal Reference Clock

The MCG Internal Reference Clock (MCGIRCLK) provides a clock source for other on-chip peripherals and is enabled when C1[IRCLKEN]=1. When enabled, MCGIRCLK is driven by either the fast internal reference clock (2 MHz IRC) or the slow internal reference clock (32 kHz IRC). The IRCS clock frequency can be re-targeted by trimming the period of its IRCS selected internal reference clock. This can be done by writing a new trim value to the C3[SCTRIM]:C4[SCFTRIM] bits when the slow IRC clock is selected or by writing a new trim value to the C4[FCTRIM] bits when the fast IRC clock is selected. The internal reference clock period is proportional to the trim value written. C3[SCTRIM]:C4[SCFTRIM] (if C2[IRCS]=0) and C4[FCTRIM] (if C2[IRCS]=1) bits affect the MCGOUTCLK frequency if the MCG is in FBI or BLPI modes. C3[SCTRIM]:C4[SCFTRIM] (if C2[IRCS]=0) bits also affect the MCGOUTCLK frequency if the MCG is in FEI mode.

Additionally, this clock can be enabled in Stop mode by setting C1[IRCLKEN] and C1[IREFSTEN], otherwise this clock is disabled in Stop mode.

21.4.4 External Reference Clock

The MCG module can support an external reference clock in all modes. See the device datasheet for external reference frequency range. When C1[IREFS] is set, the external reference clock will not be used by the FLL or PLL. In these modes, the frequency can be equal to the maximum frequency the chip-level timing specifications will support.

If the CME is asserted the slow internal reference clock is enabled along with the enabled external clock monitor. For the case when C6[CME]=1, a loss of clock is detected if the OSC external reference falls below a minimum frequency (f_{loc_high} or f_{loc_low} depending on C2[RANGE]).

NOTE

All clock monitors must be disabled before entering these low-power modes: Stop, VLPS, VLPR, VLPW, and VLLSx.

Upon detect of a loss of clock event, the MCU generates a system reset if the respective LOCRE bit is set. Otherwise the MCG sets the respective LOCS bit and the MCG generates a LOCS interrupt request.

21.4.5 MCG Fixed frequency clock

The MCG Fixed Frequency Clock (MCGFFCLK) provides a fixed frequency clock source for other on-chip peripherals; see the block diagram. This clock is driven by either the slow clock from the internal reference clock generator or the external reference clock from the Crystal Oscillator, divided by the FLL reference clock divider. The source of MCGFFCLK is selected by C1[IREFS].

This clock is synchronized to the peripheral bus clock and is valid only when its frequency is not more than 1/8 of the MCGOUTCLK frequency. When it is not valid, it is disabled and held high. The MCGFFCLK is not available when the MCG is in BLPI mode. This clock is also disabled in Stop mode. The FLL reference clock must be set within the valid frequency range for the MCGFFCLK.

21.4.6 MCG Background Debug Controller Clock

The MCG Background Debug Controller Clock (MCGBDCCLK) provides a clock source to the Background Debug Controller (BDC). This clock is driven by either the SYSCLK or the 4 MHz internal reference clock (IRC), as selected by the XCSR[CLKSW] bit in the V1 ColdFire debug module.

21.4.7 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[LOCK] is set.

21.4.8 MCG Auto TRIM (ATM)

The MCG Auto Trim (ATM) is a MCG feature that when enabled, it configures the MCG hardware to automatically trim the MCG Internal Reference Clocks using an external clock as a reference. The selection between which MCG IRC clock gets tested and enabled is controlled by the ATC[ATMS] control bit (ATC[ATMS]=0 selects the 32 kHz IRC and ATC[ATMS]=1 selects the 4 MHz IRC). If 4 MHz IRC is selected for the ATM, a divide by 128 is enabled to divide down the 4 MHz IRC to a range of 31.250 kHz.

When MCG ATM is enabled by writing ATC[ATME] bit to 1, The ATM machine will start auto trimming the selected IRC clock. During the autotrim process, ATC[ATME] will remain asserted and will deassert after ATM is completed or an abort occurs. The MCG ATM is aborted if a write to any of the following control registers is detected : C1, C3, C4, or ATC or if Stop mode is entered. If an abort occurs, ATC[ATMF] fail flag is asserted.

The ATM machine uses the bus clock as the external reference clock to perform the IRC auto-trim. Therefore, it is required that the MCG is configured in a clock mode where the reference clock used to generate the system clock is the external reference clock such as FBE clock mode. The MCG must not be configured in a clock mode where selected IRC ATM clock is used to generate the system clock. The bus clock is also required to be running with in the range of 8–16 MHz.

To perform the ATM on the selected IRC, the ATM machine uses the successive approximation technique to adjust the IRC trim bits to generate the desired IRC trimmed frequency. The ATM SARs each of the ATM IRC trim bits starting with the MSB. For each trim bit test, the ATM uses a pulse that is generated by the ATM selected IRC clock to enable a counter that counts number of ATM external clocks. At end of each trim bit, the ATM external counter value is compared to the ATCV[15:0] register value. Based on the comparison result, the ATM trim bit under test will get cleared or stay asserted. This is done until all trim bits have been tested by ATM SAR machine.

Before the ATM can be enabled, the ATM expected count needs to be derived and stored into the ATCV register. The ATCV expected count is derived based on the required target Internal Reference Clock (IRC) frequency, and the frequency of the external reference clock using the following formula:

$$\text{ATCV Expected Count Value} = 21*(Fe/Fr)$$

- Fr = Target Internal Reference Clock (IRC) Trimmed Frequency
- Fe = External Clock Frequency

If the auto trim is being performed on the 4 MHz IRC, the calculated expected count value must be multiplied by 128 before storing it in the ATCV register. Therefore, the ATCV Expected Count Value for trimming the 4 MHz IRC is calculated using the following formula.

$$\text{Expected Count Value} = (Fe/Fr)*21*(128)$$

21.5 Initialization / Application information

This section describes how to initialize and configure the MCG module in an application. The following sections include examples on how to initialize the MCG and properly switch between the various available modes.

21.5.1 MCG module initialization sequence

The MCG comes out of reset configured for FEI mode. The internal reference will stabilize in $t_{irefsts}$ microseconds before the FLL can acquire lock. As soon as the internal reference is stable, the FLL will acquire lock in $t_{fll\_acquire}$ milliseconds.

21.5.1.1 Initializing the MCG

Because the MCG comes out of reset in FEI mode, the only MCG modes that can be directly switched to upon reset are FEE, FBE, and FBI modes (see Figure 21-12). Reaching any of the other modes requires first configuring the MCG for one of these three intermediate modes. Care must be taken to check relevant status bits in the MCG status register reflecting all configuration changes within each mode.

To change from FEI mode to FEE or FBE modes, follow this procedure:

1. Enable the external clock source by setting the appropriate bits in C2 register.

2. Write to C1 register to select the clock mode.
   - If entering FEE mode, set C1[FRDIV] appropriately, clear the C1[IREFS] bit to switch to the external reference, and leave the C1[CLKS] bits at 2'b00 so that the output of the FLL is selected as the system clock source.
   - If entering FBE, clear the C1[IREFS] bit to switch to the external reference and change the C1[CLKS] bits to 2'b10 so that the external reference clock is selected as the system clock source. The C1[FRDIV] bits should also be set appropriately here according to the external reference frequency to keep the FLL reference clock in the range of 31.25 kHz to 39.0625 kHz. Although the FLL is bypassed, it is still on in FBE mode.
   - The internal reference can optionally be kept running by setting the C1[IRCLKEN] bit. This is useful if the application will switch back and forth between internal and external modes. For minimum power consumption, leave the internal reference disabled while in an external clock mode.
3. Once the proper configuration bits have been set, wait for the affected bits in the MCG status register to be changed appropriately, reflecting that the MCG has moved into the proper mode.
   - If the MCG is in FEE, FBE, PEE, PBE, or BLPE mode, and C2[EREFS] was also set in step 1, wait here for S[OSCINIT] bit to become set indicating that the external clock source has finished its initialization cycles and stabilized.
   - If in FEE mode, check to make sure the S[IREFST] bit is cleared before moving on.
   - If in FBE mode, check to make sure the S[IREFST] bit is cleared and S[CLKST] bits have changed to 2'b10 indicating the external reference clock has been appropriately selected. Although the FLL is bypassed, it is still on in FBE mode.
4. Write to the C4 register to determine the DCO output (MCGFLLCLK) frequency range.
   - By default, with C4[DMX32] cleared to 0, the FLL multiplier for the DCO output is 640. For greater flexibility, if a mid-low-range FLL multiplier of 1280 is desired instead, set C4[DRST_DRS] bits to 2'b01 for a DCO output frequency of 40 MHz. If a mid high-range FLL multiplier of 1920 is desired instead, set the C4[DRST_DRS] bits to 2'b10 for a DCO output frequency of 60 MHz. If a high-range FLL multiplier of 2560 is desired instead, set the C4[DRST_DRS] bits to 2'b11 for a DCO output frequency of 80 MHz.

- When using a 32.768 kHz external reference, if the maximum low-range DCO frequency that can be achieved with a 32.768 kHz reference is desired, set C4[DRST_DRS] bits to 2'b00 and set C4[DMX32] bit to 1. The resulting DCO output (MCGOUTCLK) frequency with the new multiplier of 732 will be 24 MHz.
- When using a 32.768 kHz external reference, if the maximum mid-range DCO frequency that can be achieved with a 32.768 kHz reference is desired, set C4[DRST_DRS] bits to 2'b01 and set C4[DMX32] bit to 1. The resulting DCO output (MCGOUTCLK) frequency with the new multiplier of 1464 will be 48 MHz.
- When using a 32.768 kHz external reference, if the maximum mid high-range DCO frequency that can be achieved with a 32.768 kHz reference is desired, set C4[DRST_DRS] bits to 2'b10 and set C4[DMX32] bit to 1. The resulting DCO output (MCGOUTCLK) frequency with the new multiplier of 2197 will be 72 MHz.
- When using a 32.768 kHz external reference, if the maximum high-range DCO frequency that can be achieved with a 32.768 kHz reference is desired, set C4[DRST_DRS] bits to 2'b11 and set C4[DMX32] bit to 1. The resulting DCO output (MCGOUTCLK) frequency with the new multiplier of 2929 will be 96 MHz.

5. Wait for the FLL lock time to guarantee FLL is running at new C4[DRST_DRS] and C4[DMX32] programmed frequency.

To change from FEI clock mode to FBI clock mode, follow this procedure:

1. Change C1[CLKS] bits in C1 register to 2'b01 so that the internal reference clock is selected as the system clock source.
2. Wait for S[CLKST] bits in the MCG status register to change to 2'b01, indicating that the internal reference clock has been appropriately selected.
3. Write to the C2 register to determine the IRCS output (IRCSCLK) frequency range.
   - By default, with C2[IRCS] cleared to 0, the IRCS selected output clock is the slow internal reference clock (32 kHz IRC). If the faster IRC is desired, set C2[IRCS] bit to 1 for a IRCS clock derived from the 4 MHz IRC source.

21.5.2 Using a 32.768 kHz reference

In FEE and FBE modes, if using a 32.768 kHz external reference, at the default FLL multiplication factor of 640, the DCO output (MCGFLLCLK) frequency is 20.97 MHz at low-range. If C4[DRST_DRS] bits are set to 2'b01, the multiplication factor is doubled to 1280, and the resulting DCO output frequency is 41.94 MHz at mid-low-range. If C4[DRST_DRS] bits are set to 2'b10, the multiplication factor is set to 1920, and the resulting DCO output frequency is 62.91 MHz at mid high-range. If C4[DRST_DRS] bits are set to 2'b11, the multiplication factor is set to 2560, and the resulting DCO output frequency is 83.89 MHz at high-range.

In FBI and FEI modes, setting C4[DMX32] bit is not recommended. If the internal reference is trimmed to a frequency above 32.768 kHz, the greater FLL multiplication factor could potentially push the microcontroller system clock out of specification and damage the part.

21.5.3 MCG mode switching

When switching between operational modes of the MCG, certain configuration bits must be changed in order to properly move from one mode to another. Each time any of these bits are changed (C6[PLLS], C1[IREFS], C1[CLKS], C2[IRCS], or C2[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[PRDIV]) is set properly for the mode being switched to. For instance, in PEE mode, if using a 4 MHz crystal, C5[PRDIV] must be set to 5'b000 (divide-by-1) or 5'b001 (divide-by-2) to divide the external reference down to the required frequency between 2 and 4 MHz.

In FBE, FEE, FBI, and FEI modes, at any time, the application can switch the FLL multiplication factor between 640, 1280, 1920, and 2560 with C4[DRST_DRS] bits. Writes to C4[DRST_DRS] bits will be ignored if C2[LP]=1.

The table below shows MCGOUTCLK frequency calculations using C1[FRDIV], C5[PRDIV], and C6[VDIV] settings for each clock mode.

Table 21-15. MCGOUTCLK Frequency Calculation Options

Clock Mode	$f_{MCGOUTCLK}$[1]	Note
FEI (FLL engaged internal)	(f_{int} * F)	Typical $f_{MCGOUTCLK}$ = 21 MHz immediately after reset.
FEE (FLL engaged external)	(f_{ext} / FLL_R) *F	f_{ext} / FLL_R must be in the range of 31.25 kHz to 39.0625 kHz

Table continues on the next page...

Table 21-15. MCGOUTCLK Frequency Calculation Options (continued)

Clock Mode	$f_{MCGOUTCLK}$[1]	Note
FBE (FLL bypassed external)	OSCCLK	OSCCLK / FLL_R must be in the range of 31.25 kHz to 39.0625 kHz
FBI (FLL bypassed internal)	MCGIRCLK	Selectable between slow and fast IRC
PEE (PLL engaged external)	(OSCCLK / PLL_R) * M	OSCCLK / PLL_R must be in the range of 2 – 4 MHz
PBE (PLL bypassed external)	OSCCLK	OSCCLK / PLL_R must be in the range of 2 – 4 MHz
BLPI (Bypassed low power internal)	MCGIRCLK	Selectable between slow and fast IRC
BLPE (Bypassed low power external)	OSCCLK	

1. FLL_R is the reference divider selected by the C1[FRDIV] bits, PLL_R is the reference divider selected by C5[PRDIV] bits, F is the FLL factor selected by C4[DRST_DRS] and C4[DMX32] bits, and M is the multiplier selected by C6[VDIV] bits.

This section will include three mode switching examples using an 4 MHz external crystal. If using an external clock source less than 2 MHz, the MCG must not be configured for any of the PLL modes (PEE and PBE).

21.5.3.1 Example 1: Moving from FEI to PEE mode: External Crystal = 4 MHz, MCGOUTCLK frequency = 48 MHz

In this example, the MCG will move through the proper operational modes from FEI to PEE to achieve 48 MHz MCGOUTCLK frequency from 4 MHz external crystal reference. First, the code sequence will be described. Then there is a flowchart that illustrates the sequence.

1. First, FEI must transition to FBE mode:
   a. C2 = 0x2C
      - C2[RANGE] set to 2'b01 because the frequency of 4 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 = 0x90
      - C1[CLKS] set to 2'b10 to select external reference clock as system clock source

- C1[FRDIV] set to 3'b010, or divide-by-128 because 4 MHz / 128 = 31.25 kHz which is in the 31.25 kHz to 39.0625 kHz range required by the FLL
- C1[IREFS] cleared to 0, selecting the external reference clock and enabling the external oscillator.

c. Loop until S[OSCINIT] is 1, indicating the crystal selected by C2[EREFS] 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[PRDIV] to generate correct PLL reference frequency.

a. C5 = 0x01

- C5[PRDIV] set to 5'b001, or divide-by-2 resulting in a pll reference frequency of 4 MHz/2 = 2 MHz.

3. Then, FBE must transition either directly to PBE mode or first through BLPE mode and then to PBE mode:

a. BLPE: If a transition through BLPE mode is desired, first set C2[LP] to 1.

b. BLPE/PBE: C6 = 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 4 MHz/ 2 = 2 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-24 because 2 MHz reference * 24 = 48 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[LOCK] is set, indicating that the PLL has acquired lock.

4. Lastly, PBE mode transitions into PEE mode:

a. C1 = 0x10

- C1[CLKS] set to 2'b00 to select the output of the PLL as the system clock source.

b. Loop until S[CLKST] are 2'b11, indicating that the PLL output is selected to feed MCGOUTCLK in the current clock mode.

- Now, with PRDIV of divide-by-2, and C6[VDIV] of multiply-by-24, MCGOUTCLK = [(4 MHz / 2) * 24] = 48 MHz.

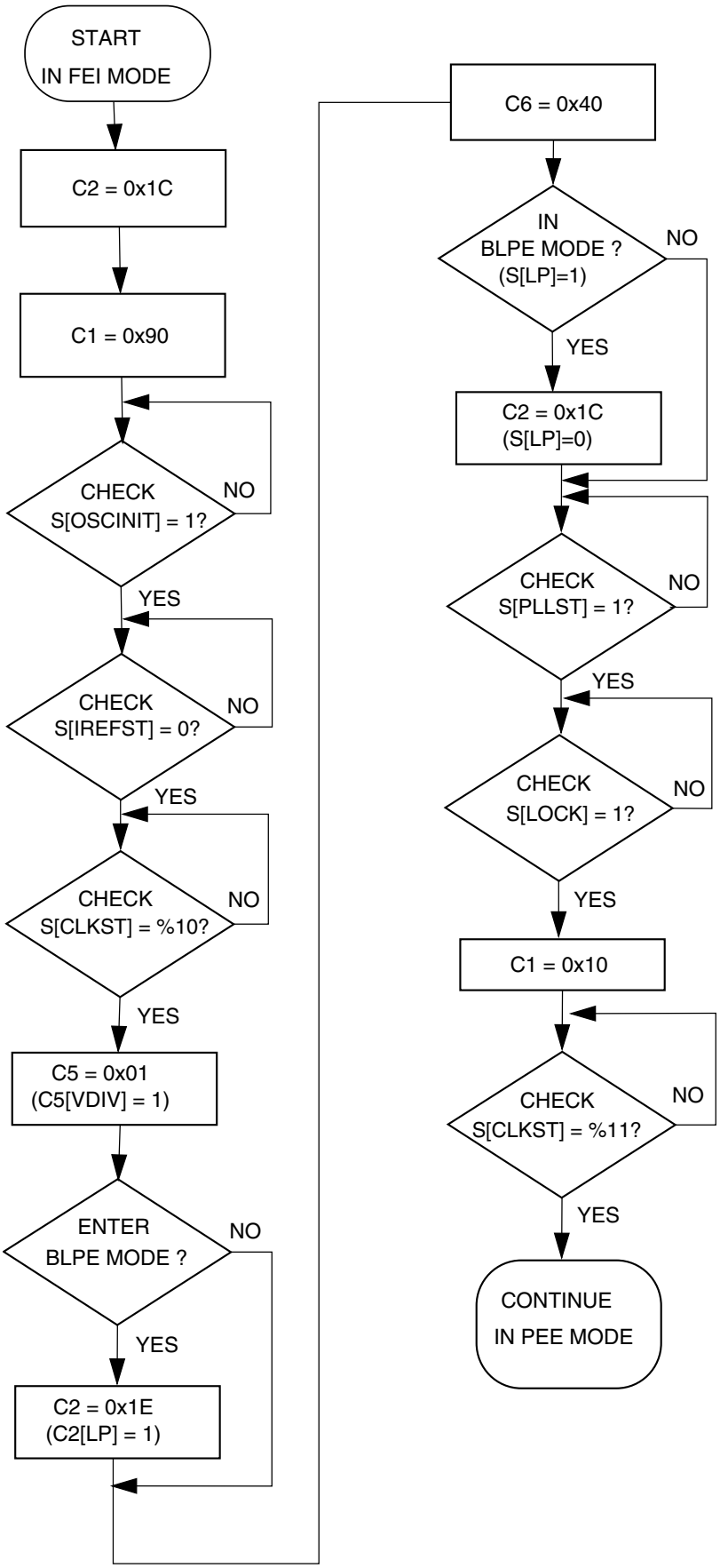


Figure 21-13. Flowchart of FEI to PEE mode transition using an 4 MHz crystal

21.5.3.2 Example 2: Moving from PEE to BLPI mode: MCGOUTCLK frequency =32 kHz

In this example, the MCG will move through the proper operational modes from PEE mode with a 4 MHz crystal configured for a 48 MHz MCGOUTCLK frequency (see previous example) to BLPI mode with a 32 kHz MCGOUTCLK frequency. First, the code sequence will be described. Then there is a flowchart that illustrates the sequence.

1. First, PEE must transition to PBE mode:
   a. C1 = 0x90
      - C1[CLKS] set to 2'b10 to switch the system clock source to the external reference clock.
   b. Loop until S[CLKST] are 2'b10, indicating that the external reference clock is selected to feed MCGOUTCLK.
2. Then, PBE must transition either directly to FBE mode or first through BLPE mode and then to FBE mode:
   a. BLPE: If a transition through BLPE mode is desired, first set C2[LP] to 1.
   b. BLPE/FBE: C6 = 0x00
      - C6[PLLS] clear to 0 to select the FLL. At this time, with C1[FRDIV] value of 3'b010, the FLL divider is set to 128, resulting in a reference frequency of 4 MHz / 128 = 31.25 kHz. If C1[FRDIV] was not previously set to 3'b010 (necessary to achieve required 31.25–39.06 kHz FLL reference frequency with an 4 MHz external source frequency), it must be changed prior to clearing C6[PLLS] bit. In BLPE mode,changing this bit only prepares the MCG for FLL usage in FBE mode. With C6[PLLS] = 0, the C6[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 = 0x54
      - C1[CLKS] set to 2'b01 to switch the system clock to the internal reference clock.

- C1[IREFS] set to 1 to select the internal reference clock as the reference clock source.
- C1[FRDIV] remain unchanged because the reference divider does not affect the internal reference.

b. Loop until S[IREFST] is 1, indicating the internal reference clock has been selected as the reference clock source.

c. Loop until S[CLKST] are 2'b01, indicating that the internal reference clock is selected to feed MCGOUTCLK.

4. Lastly, FBI transitions into BLPI mode.

a. C2 = 0x02

- C2[LP] is 1
- C2[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.

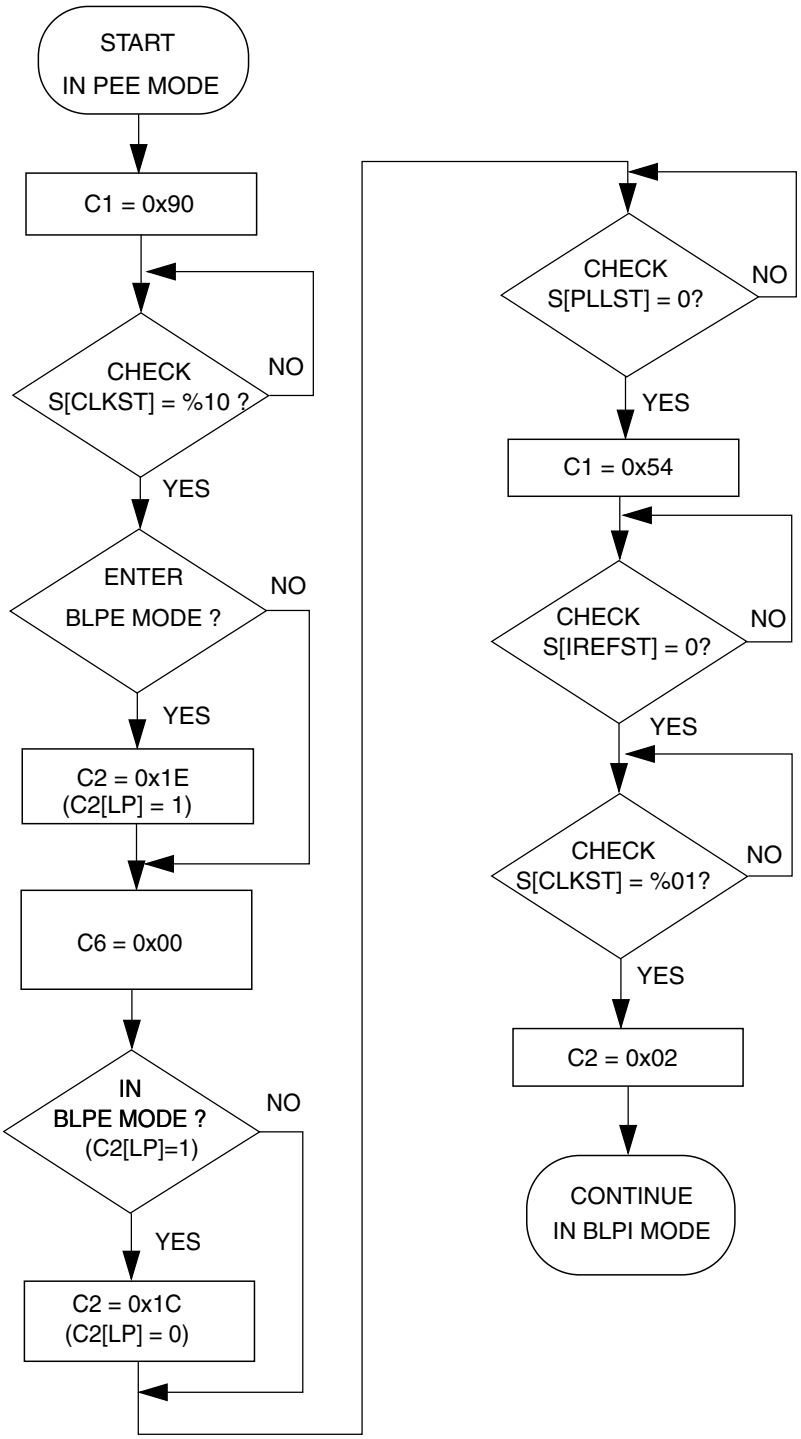


Figure 21-14. Flowchart of PEE to BLPI mode transition using an 4 MHz crystal

21.5.3.3 Example 3: Moving from BLPI to FEE mode

In this example, the MCG will move through the proper operational modes from BLPI mode at a 32 kHz MCGOUTCLK frequency running off the internal reference clock (see previous example) to FEE mode using a 4 MHz crystal configured for a 20 MHz MCGOUTCLK frequency. First, the code sequence will be described. Then there is a flowchart that illustrates the sequence.

1. First, BLPI must transition to FBI mode.
   a. C2 = 0x00
      - C2[LP] is 0
2. Next, FBI will transition to FEE mode.
   a. C2 = 0x1C
      - C2[RANGE] set to 2'b01 because the frequency of 4 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 = 0x10
      - C1[CLKS] set to 2'b00 to select the output of the FLL as system clock source.
      - C1[FRDIV] remain at 3'b010, or divide-by-128 for a reference of 4 MHz / 128 = 31.25 kHz.
      - C1[IREFS] cleared to 0, selecting the external reference clock.
   c. Loop until S[OSCINIT] is 1, indicating the crystal selected by the C2[EREFS] bit has been initialized.
   d. Loop until S[IREFST] is 0, indicating the external reference clock is the current source for the reference clock.
   e. Loop until S[CLKST] are 2'b00, indicating that the output of the FLL is selected to feed MCGOUTCLK.
   f. Now, with a 31.25 kHz reference frequency, a fixed DCO multiplier of 640, MCGOUTCLK = 31.25 kHz * 640 / 1 = 20 MHz.
   g. At this point, by default, the C4[DRST_DRS] bits are set to 2'b00 and C4[DMX32] is cleared to 0. If the MCGOUTCLK frequency of 40 MHz is desired instead, set the C4[DRST_DRS] bits to 0x01 to switch the FLL

multiplication factor from 640 to 1280. To return the MCGOUTCLK frequency to 20 MHz, set C4[DRST_DRS] bits to 2'b00 again, and the FLL multiplication factor will switch back to 640.

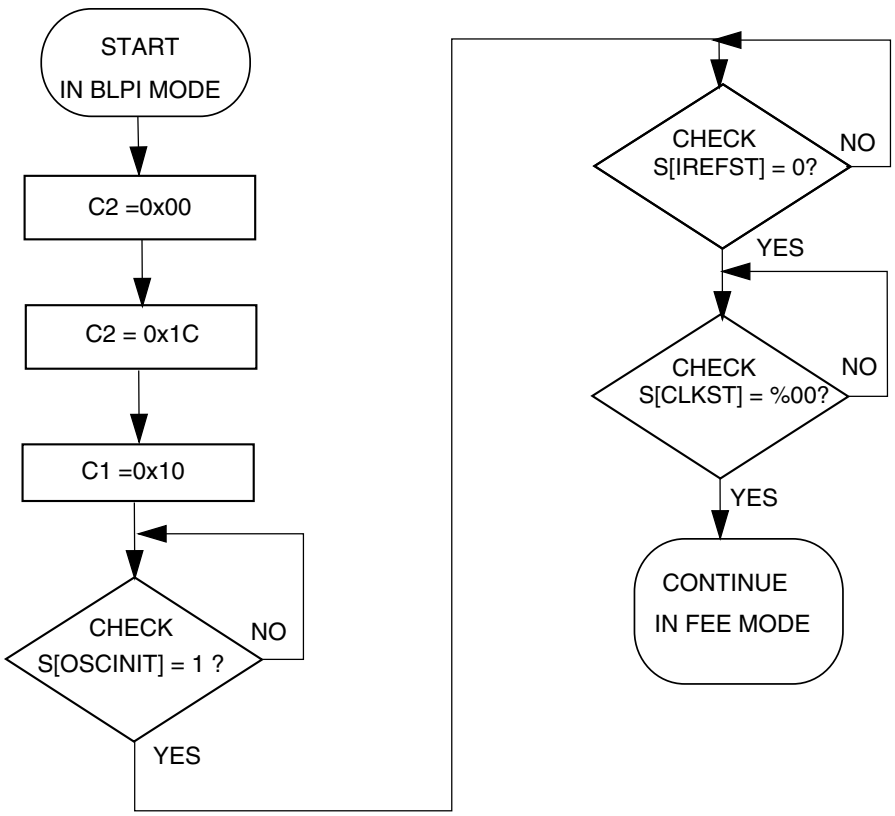


Figure 21-15. Flowchart of BLPI to FEE mode transition using an 4 MHz crystal

Chapter 22
Oscillator (OSC)

22.1 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.

22.2 Features and Modes

Key features of the module are:

- Supports 32 kHz crystals (Low Range mode)
- Supports 1–8 MHz, crystals and resonators (High Range mode)
- Automatic Gain Control (AGC) to optimize power consumption in high frequency ranges 1–8 MHz, using low-power mode
- High gain option in frequency ranges: 32 kHz, 1–8 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.

22.3 Block Diagram

The OSC module uses a crystal or resonator to generate three filtered oscillator clock signals. Three clocks are output from OSC module: OSCCLK for MCU system, OSCERCLK for on-chip peripherals, and OSC32KCLK. The OSCCLK can only work in run mode. OSCERCLK and OSC32KCLK can work in low power modes. For the clock source assignments, refer to the clock distribution information of this MCU.

Refer to the chip configuration chapter for the external reference clock source in this MCU.

The following figure shows the block diagram of the OSC module.

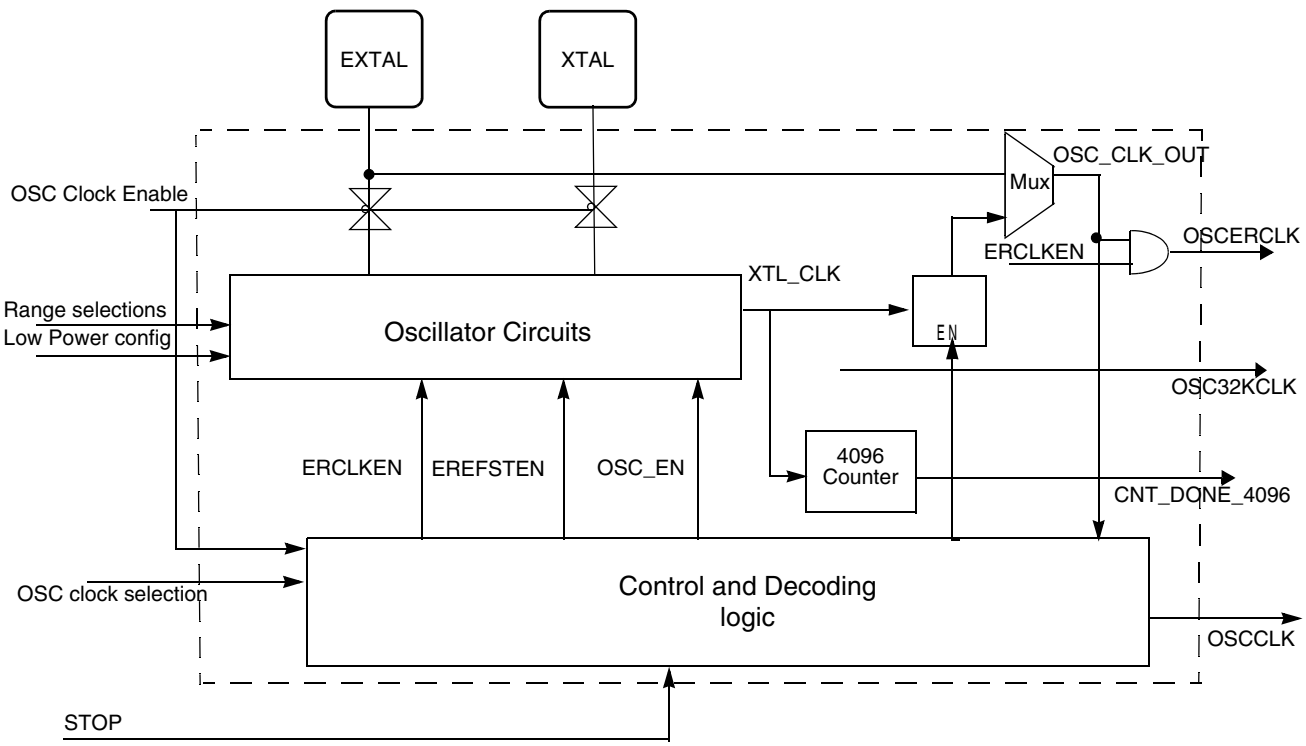


Figure 22-1. OSC Module Block Diagram

22.4 OSC Signal Descriptions

The following table shows the user-accessible signals available for the OSC module. Refer to signal multiplexing information for this MCU for more details.

Table 22-1. OSC Signal Descriptions

Signal	Description	I/O
EXTAL	External clock/Oscillator input	I
XTAL	Oscillator output	O

22.5 External Crystal / Resonator Connections

The connections for a crystal/resonator frequency reference are shown in the following figures. When using low-frequency, low-power mode, the only external component is the crystal or ceramic resonator itself. In the other oscillator modes, load capacitors (C_x, C_y) and feedback resistor (R_F) are required. In addition, a series resistor (R_S) may be used in high-gain modes. The following table shows all possible connections.

Table 22-2. External Caystal/Resonator Connections

Oscillator Mode	Connections
Low-frequency (32 kHz), low-power	Connection 1
Low-frequency (32 kHz), high-gain	Connection 2/Connection 4[1]/ Connection 3[2]
High-frequency (1~32 MHz), low-power	Connection 1/Connection 3[2,3]
High-frequency (1~32 MHz), high-gain	Connection 2/Connection 4[1]/ Connection 3[2]

1. When the frequency of the crystal is 32 kHz and the load capacitors (Cx, Cy) are less than 16 pF, use Connection 4.
2. When the load capacitors (Cx, Cy) are greater than 30 pF, use Connection 3.
3. With the low-power mode, the oscillator has the internal feedback resistor R_F. Therefore, the feedback resistor must not be externally with the Connection 3.

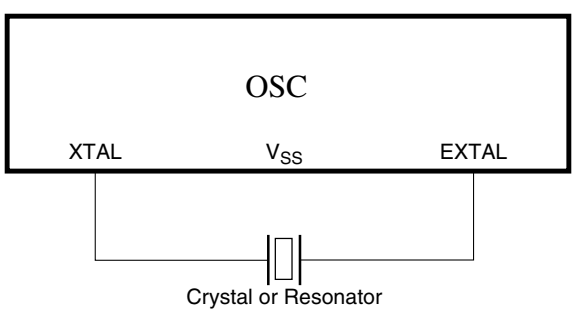


Figure 22-2. Crystal/Ceramic Resonator Connections - Connection 1

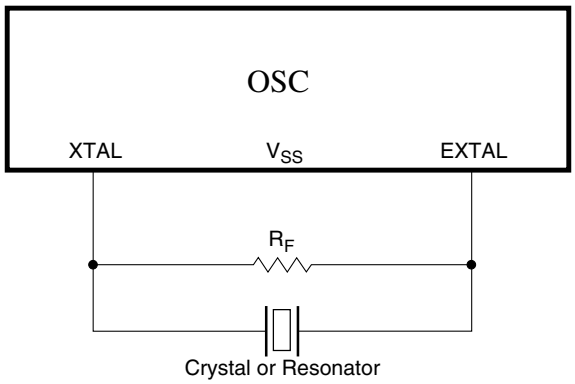


Figure 22-3. Crystal/Ceramic Resonator Connections - Connection 2

NOTE

Connection 1 and Connection 2 should use internal capacitors as the load of the oscillator by configuring the CR[SCxP] bits.

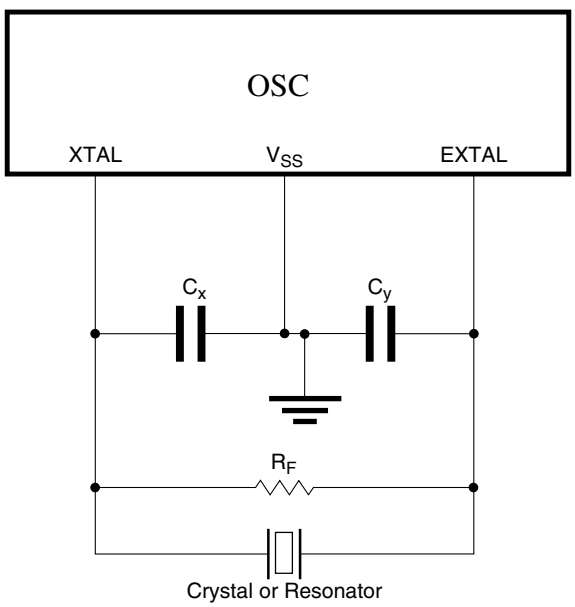


Figure 22-4. Crystal/Ceramic Resonator Connections - Connection 3

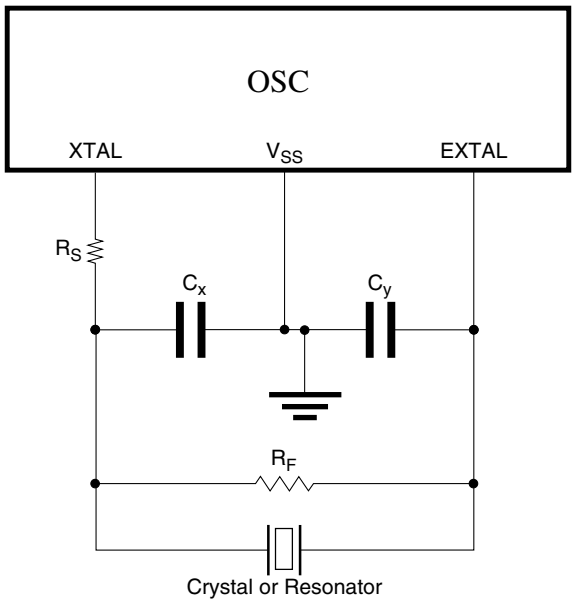


Figure 22-5. Crystal/Ceramic Resonator Connections - Connection 4

22.6 External Clock Connections

In external clock mode, the pins can be connected as shown below.

NOTE

XTAL can be used as a GPIO when the GPIO alternate function is configured for it.

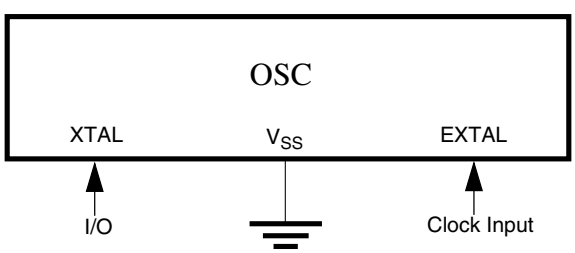


Figure 22-6. External Clock Connections

22.7 Memory Map/Register Definitions

Some oscillator module register bits are typically incorporated into other peripherals such as MCG or SIM.

22.7.1 OSC Memory Map/Register Definition

OSC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_8100	OSC Control Register (OSC1_CR)	8	R/W	00h	22.71.1/404
FFFF_8110	OSC Control Register (OSC2_CR)	8	R/W	00h	22.71.1/404

22.71.1 OSC Control Register (OSCx_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: Base address + 0h offset

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

OSCx_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

Table continues on the next page...

OSCx_CR field descriptions (continued)

Field	Description
	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.

22.8 Functional Description

This following sections provide functional details of the module.

22.8.1 OSC Module States

The states of the OSC module are shown in the following figure. The states and their transitions between each other are described in this section.

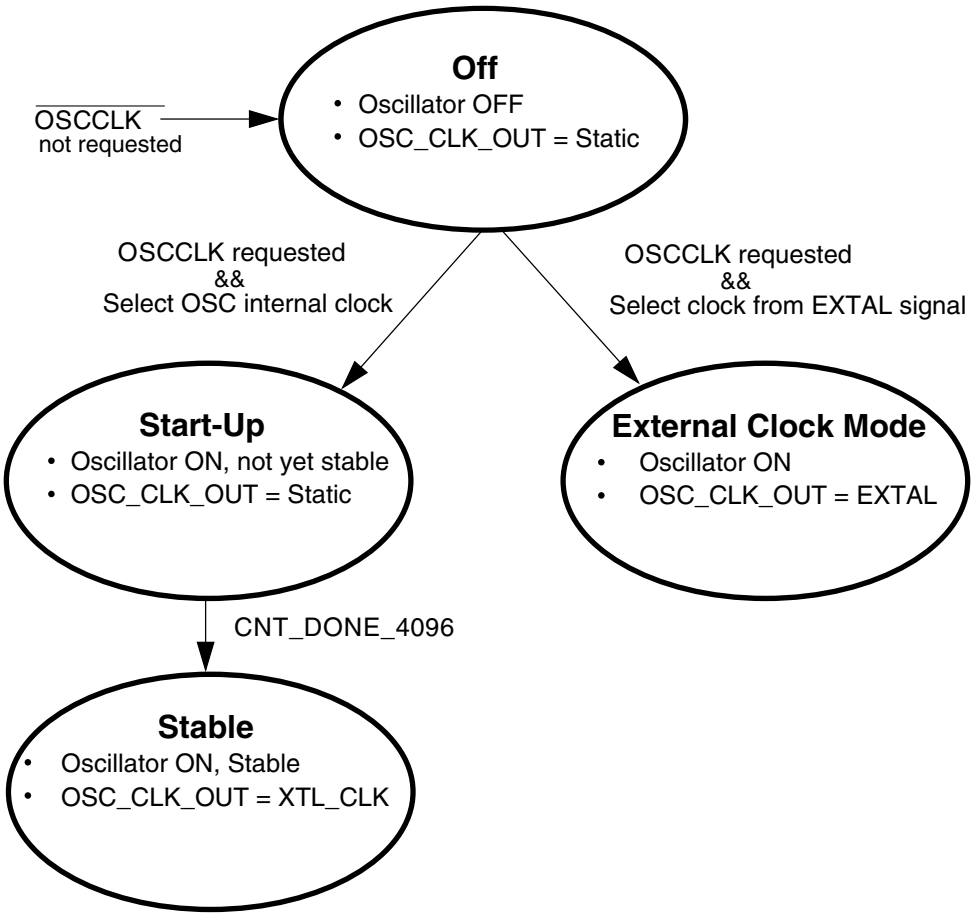


Figure 22-10. OSC Module State Diagram

NOTE

XTL_CLK is the clock generated internally from OSC circuits.

22.8.1.1 Off

The OSC enters the Off state when the system does not require OSC clocks. Upon entering this state, XTL_CLK is static unless OSC is configured to select the clock from the EXTAL pad by clearing the external reference clock selection bit. For details regarding the external reference clock source in this MCU, refer to the chip configuration chapter. The EXTAL and XTAL pins are also decoupled from all other oscillator circuitry in this state. The OSC module circuitry is configured to draw minimal current.

22.8.1.2 Oscillator Start-Up

The OSC enters start-up state when it is configured to generate clocks (internally the OSC_EN transitions high) using the internal oscillator circuits by setting the external reference clock selection bit. In this state, the OSC module is enabled and oscillations are starting up, but have not yet stabilized. When the oscillation amplitude becomes large enough to pass through the input buffer, XTL_CLK begins clocking the counter. When the counter reaches 4096 cycles of XTL_CLK, the oscillator is considered stable and XTL_CLK is passed to the output clock OSC_CLK_OUT.

22.8.1.3 Oscillator Stable

The OSC enters stable state when it is configured to generate clocks (internally the OSC_EN transitions high) using the internal oscillator circuits by setting the external reference clock selection bit and the counter reaches 4096 cycles of XTL_CLK (when CNT_DONE_4096 is high). In this state, the OSC module is producing a stable output clock on OSC_CLK_OUT. Its frequency is determined by the external components being used.

22.8.1.4 External Clock Mode

The OSC enters external clock state when it is enabled and external reference clock selection bit is cleared. For details regarding external reference clock source in this MCU, refer to the chip configuration chapter. In this state, the OSC module is set to buffer (with hysteresis) a clock from EXTAL onto the OSC_CLK_OUT. Its frequency is determined by the external clock being supplied.

22.8.2 OSC Module Modes

The OSC is a Pierce-type oscillator that supports external crystals or resonators operating over the frequency ranges shown in Table 22-9. 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 22-9. Oscillator Modes

Mode	Frequency Range
Low-frequency, high-gain	f_{osc_lo} (1 kHz) up to f_{osc_lo} (32.768 kHz)

Table continues on the next page...

Table 22-9. Oscillator Modes (continued)

Mode	Frequency Range
High-frequency mode1, high-gain	$f_{osc\_hi\_1}$ (1 MHz) up to $f_{osc\_hi\_1}$ (8 MHz)
High-frequency mode1, low-power	
High-frequency mode2, high-gain	$f_{osc\_hi\_2}$ (8 MHz) up to $f_{osc\_hi\_2}$ (32 MHz)
High-frequency mode2, low-power	

NOTE

For information about low power modes of operation used in this chip and their alignment with some OSC modes, refer to the chip's Power Management details.

22.8.2.1 Low-Frequency, High-Gain Mode

In Low-frequency, high-gain mode, the oscillator uses a simple inverter-style amplifier. The gain is set to achieve rail-to-rail oscillation amplitudes.

The oscillator input buffer in this mode is single-ended. It provides low pass frequency filtering as well as hysteresis for voltage filtering and converts the output to logic levels. In this mode, the internal capacitors could be used.

22.8.2.2 Low-Frequency, Low-Power Mode

In low-frequency, low-power mode, the oscillator uses a gain control loop to minimize power consumption. As the oscillation amplitude increases, the amplifier current is reduced. This continues until a desired amplitude is achieved at steady-state. This mode provides low pass frequency filtering as well as hysteresis for voltage filtering and converts the output to logic levels. In this mode, the internal capacitors could be used, the internal feedback resistor is connected, and no external resistor should be used.

In this mode, the amplifier inputs, gain-control input, and input buffer input are all capacitively coupled for leakage tolerance (not sensitive to the DC level of EXTAL).

Also in this mode, all external components except for the resonator itself are integrated, which includes the load capacitors and feeback resistor that biases EXTAL.

22.8.2.3 High-Frequency, High-Gain Mode

In high-frequency, high-gain mode, the oscillator uses a simple inverter-style amplifier. The gain is set to achieve rail-to-rail oscillation amplitudes. This mode provides low pass frequency filtering as well as hysteresis for voltage filtering and converts the output to logic levels. In this mode, the internal capacitors could be used.

22.8.2.4 High-Frequency, Low-Power Mode

In high-frequency, low-power mode, the oscillator uses a gain control loop to minimize power consumption. As the oscillation amplitude increases, the amplifier current is reduced. This continues until a desired amplitude is achieved at steady-state. In this mode, the internal capacitors could be used, the internal feedback resistor is connected, and no external resistor should be used.

The oscillator input buffer in this mode is differential. It provides low pass frequency filtering as well as hysteresis for voltage filtering and converts the output to logic levels.

22.8.3 Counter

The oscillator output clock (OSC_CLK_OUT) is gated off until the counter has detected 4096 cycles of its input clock (XTL_CLK). After 4096 cycles are completed, the counter passes XTL_CLK onto OSC_CLK_OUT. This counting time-out is used to guarantee output clock stability.

22.8.4 Reference Clock Pin Requirements

The OSC module requires use of both the EXTAL and XTAL pins to generate an output clock in Oscillator mode, but requires only the EXTAL pin in External clock mode. The EXTAL and XTAL pins are available for I/O. For the implementation of these pins on this device, refer to the Signal Multiplexing chapter.

22.9 Reset

There is no reset state associated with the OSC module. The counter logic is reset when the OSC is not configured to generate clocks.

There are no sources of reset requests for the OSC module.

22.10 Low Power Modes Operation

When the MCU enters Stop modes, the OSC is functional depending on ERCLKEN and EREFSETN bit settings. If both these bits are set, the OSC is in operation. After waking up from Very Low Leakage Stop (VLLSx) modes, all OSC register bits are reset and initialization is required through software.

22.11 Interrupts

The OSC module does not generate any interrupts.

Chapter 23
Real Time Clock (RTC)

23.1 Introduction

This block is a low power module that provides time keeping and calendaring functions.

23.1.1 Features

This block supports the following features:

- Designed for low power
  - Time and Date counters are rippled with respect to each other to prevent simultaneous toggling
- Basic Clock functions
  - Separate counters for Days, Hour, Minutes and Seconds
  - Calendaring support – Separate counters for Months, Year and Day of the Week
    - Automatic adjustment for Day Light Saving with user defined parameters
    - Automatic month and leap year adjustment
  - External clock support to run the counters in the case user wishes to provide externally compensated 1Hz clock.
- RTC utilizes 'local time' which implicitly contains the time zone offset
- Programmable alarm with interrupt. Alarm is output from RTC in case MCU wants to use it as a wake up event
- 8 periodic interrupts (Sampling Timer Interrupts)
- 16-bit CPU register programming interface

23.1.2 Modes of Operation

This section describes the RTC modes of operation.

23.1.2.1 Wait Mode

In Wait mode, the RTC is fully operational. Since the RTC runs off a 32.768 kHz clock, which is not gated in Wait mode; RTC 's time keeping functions and other functions that depend on the 32.768 kHz clock, continue to operate independent of CPU. Only the register block's clock is gated. No register contents are lost.

23.1.2.2 Stop Mode

In Stop mode, the RTC is fully operational and behaves in the same way as in Wait mode. The 1 Hz output clock is not maintained during CPU stop mode.

23.1.3 Design Overview

The RTC block provides basic time keeping functions through seconds, minutes, hours counters, and calendaring functions via date, day-of-week, month and year counters; along with automatic adjustments for leap year and day light saving. Reading these counters indicates the current date and time and writing to these registers sets the date and time as provided by the user.

The alarm is set for specific hour, minute and second. When the time counters match the configured alarm hour, minute and second settings, the alarm is set and an interrupt to CPU is generated, if the alarm interrupt is enabled. The alarm can additionally be configured to match days, months and year to generate the alarm interrupt. The alarm signal has been brought out from RTC for use by MCU to allow certain wake up events.

RTC module also provides 8 sampling timer interrupts apart from normal interrupts for alarm .

The registers in the RTC module are configured via the CPU register programming interface.

For detailed description on the complete functionality of RTC, refer to Functional Description.

23.2 Signal Description

This section lists all pad inputs to the RTC block.

23.2.1 EXTAL32K, XTAL32K

These pins serve as the connection pins to the external 32.768 kHz crystal.

23.3 Memory Map and Registers

The address of a register is the sum of a base address and an address offset. The base address is defined at the chip level. The address offset is defined at the module level.

NOTE

Initially, a 32 kHz clock is needed to initialize the RTC. This clock can later be gated while programming the RTC registers.

NOTE

For the General Purpose Data Register (RTC_GP_DATA_REG), the use of the fields is specific to the MCU. See the Chip Configuration chapter for a description of this register. Software reset has no effect on the contents of this register.

RTC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_8840	RTC Year and Month Counters Register (RTC_YEARMON)	16	R/W	0001h	23.3.1/414
FFFF_8842	RTC Days and Day-of-Week Counters Register (RTC_DAYS)	16	R/W	0001h	23.3.2/415
FFFF_8844	RTC Hours and Minutes Counters Register (RTC_HOURMIN)	16	R/W	0000h	23.3.3/416
FFFF_8846	RTC Seconds Counters Register (RTC_SECONDS)	16	R/W	0000h	23.3.4/417
FFFF_8848	RTC Year and Months Alarm Register (RTC_ALM_YEARMON)	16	R/W	0000h	23.3.5/418
FFFF_884A	RTC Days Alarm Register (RTC_ALM_DAYS)	16	R/W	0000h	23.3.6/418

Table continues on the next page...

RTC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_884C	RTC Hours and Minutes Alarm Register (RTC_ALM_HOURMIN)	16	R/W	0000h	23.3.7/419
FFFF_884E	RTC Seconds Alarm Register (RTC_ALM_SECONDS)	16	R/W	0000h	23.3.8/419
FFFF_8850	RTC Control Register (RTC_CTRL)	16	R/W	See section	23.3.9/421
FFFF_8852	RTC Status Register (RTC_STATUS)	16	R/W	0000h	23.3.10/422
FFFF_8854	RTC Interrupt Status Register (RTC_ISR)	16	w1c	0000h	23.3.11/423
FFFF_8856	RTC Interrupt Enable Register (RTC_IER)	16	R/W	0000h	23.3.12/426
FFFF_8862	RTC Daylight Saving Hour Register (RTC_DST_HOUR)	16	R/W	0000h	23.3.13/428
FFFF_8864	RTC Daylight Saving Month Register (RTC_DST_MONTH)	16	R/W	0000h	23.3.14/429
FFFF_8866	RTC Daylight Saving Day Register (RTC_DST_DAY)	16	R/W	0000h	23.3.15/430
FFFF_8868	RTC Prescaler Register (RTC_PRESCALER)	16	R/W	8000h	23.3.16/430

23.3.1 RTC Year and Month Counters Register (RTC_YEARMON)

This register stores the value of the month and year counters. The year field does not store the actual year value but the offset in years from a hardcoded BASE YEAR taken as the year 2112. This is a signed value and the ranges from –128 to +127. The software should program the offset from that base year into this register. For example, if the current year is 2007, then this will be represented in this register as –105 or 0x97. The actual year value can be found by adding the BASE YEAR and the offset in the YEARMON[15:8] register. Hence the range of year supported will be 1984 (2112 – 128) to 2239 (2112 + 127).

Hence for year calculation:

Actual Year = Base Year (for example, 2112) + Offset Year

The month register stores the count value of the months register. Writing to this register loads the months counter with this new value. The valid values are mentioned in table below. Both month and year are unaffected on software reset.

User software should first determine the state of the INVAL bit in the STATUS (bit 0) to determine that the counters are stable before their value can be read or changed. The assertion of INVAL bit ensures that no operation is done at the boundary of a second when counters change value.

Address: FFFF_8840h base + 0h offset = FFFF_8840h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	YROFST								0				MON_CNT			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

RTC_YEARMON field descriptions

Field	Description
15–8 YROFST	Year Offset Count Value These bits indicate the offset in years from the base year (hard coded as 2112) and do not show the actual year value. This is a signed value. Valid values are –128 to 127. With the Base Year as 2112 and if the value of YEAR field is 0x10, the actual year will be 2112 + 16 = 2128.
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–0 MON_CNT	These bits give the value of the Months Counter . Valid Values are: 0 Illegal Value 1 January 2 February 3 March 4 April 5 May 6 June 7 July 8 August 9 September 10 October 11 November 12 December 13 Illegal Value 14 Illegal Value 15 Illegal Value

23.3.2 RTC Days and Day-of-Week Counters Register (RTC_DAYS)

This read/write register shows the current value of the day-of-week counter and days counter. Reading this register returns the latest value of the counter s . Writing to this register loads the value to the day-of-week and days counters and the counters continue to count from this new value. The day-of-week is not calculated automatically and should be written by CPU. This register is unaffected by software reset.

User software should first determine the state of the INVAL bit in the STATUS (bit 0) to determine that the counters are stable before their value can be read or changed. The assertion of INVAL bit ensures that no operation is done at the boundary of a second when counters change value.

Address: FFFF_8840h base + 2h offset = FFFF_8842h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0					DOW			0			DAY_CNT				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

RTC_DAYS 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 DOW	Day of Week Counter Value. 0 Sunday 1 Monday 2 Tuesday 3 Wednesday 4 Thrusday 5 Friday 6 Saturday 7 Reserved
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–0 DAY_CNT	Days Counter Value. Valid values are ‘1’ to ‘31’.

23.3.3 RTC Hours and Minutes Counters Register (RTC_HOURMIN)

This register is used to program the hours and minutes counter. It can be read anytime to get the current value of the counters. Only power-on reset can reset this register. Hours counter can be set anything between 0 and 23. Minutes counter can be set anything between 0 and 59. This register is unaffected by software reset.

User software should first determine the state of the INVAL bit in the STATUS (bit 0) to determine that the counters are stable before their value can be read or changed. The assertion of INVAL bit ensures that no operation is done at the boundary of a second when counters change value.

Address: FFFF_8840h base + 4h offset = FFFF_8844h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0			HOUR_CNT					0		MIN_CNT					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RTC_HOURMIN field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 HOUR_CNT	Hours Counter Value. Valid count values are 0 to 23.
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–0 MIN_CNT	Minutes Counter Value. Valid count values are 0 to 59.

23.3.4 RTC Seconds Counters Register (RTC_SECONDS)

This register is used to program the seconds counter. It can be read anytime to get the current value of the counter. Only power-on reset can reset this register. Seconds counter can be set anything between 0 and 59 both included. This register is unaffected by software reset.

User software should first determine the state of the INVAL bit in the STATUS (bit 0) to determine that the counters are stable before their value can be read or changed. The assertion of INVAL bit ensures that no operation is done at the boundary of a second when counters change value.

Address: FFFF_8840h base + 6h offset = FFFF_8846h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								0		SEC_CNT					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RTC_SECONDS field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–0 SEC_CNT	Seconds Counter Value. Valid count values are 0 to 59.

23.3.5 RTC Year and Months Alarm Register (RTC_ALM_YEARMON)

This register is used to configure the months and year setting of the alarm. The alarm setting can be read or written anytime. This register is reset to its default state on software reset. Alarm interrupt bit is set when all values of alarm seconds, minutes, hours, days, month, and year match their respective counter values.

User software can configure the type of alarm using the ALM_TYPE bits in the CTRL register.

Address: FFFF_8840h base + 8h offset = FFFF_8848h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	ALM_YEAR								0				ALM_MON			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RTC_ALM_YEARMON field descriptions

Field	Description
15–8 ALM_YEAR	Year Value for Alarm. Same as Years Offset Value in RTC Year and Month Counters Register (RTC_YEARMON) .
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–0 ALM_MON	Months Value for Alarm. Same as Months Counter Value in RTC Year and Month Counters Register (RTC_YEARMON) .

23.3.6 RTC Days Alarm Register (RTC_ALM_DAYS)

The days alarm register is used to configure the day setting of the alarm. The alarm setting can be read or written anytime. This register is reset to its default state on software reset. Alarm interrupt bit is set when all values of alarm seconds, minutes, hours, days, month and year match their respective counter values.

User software can configure the type of alarm using the ALM_TYPE bits in the CTRL register.

Address: FFFF_8840h base + Ah offset = FFFF_884Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								0			ALM_DAY				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RTC_ALM_DAYS field descriptions

Field	Description
15–8 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.
4–0 ALM_DAY	Days Value for Alarm. Same as Days Counter Value in RTC Days and Day-of-Week Counters Register (RTC_DAYS) .

23.3.7 RTC Hours and Minutes Alarm Register (RTC_ALM_HOURMIN)

The hours and minutes alarm register is used to configure the hour and minute setting of the alarm. The alarm setting can be read or written anytime. This register is reset to default state on software reset.

User software can configure the type of alarm using the ALM_TYPE bits in the CTRL register.

Address: FFFF_8840h base + Ch offset = FFFF_884Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0			ALM_HOUR					0		ALM_MIN					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RTC_ALM_HOURMIN field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 ALM_HOUR	Hours Value for Alarm. Same as Hours Counter Value in RTC Hours and Minutes Counters Register (RTC_HOURMIN) .
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–0 ALM_MIN	Minutes Value for Alarm. Same as Minutes Counter Value in RTC Hours and Minutes Counters Register (RTC_HOURMIN) .

23.3.8 RTC Seconds Alarm Register (RTC_ALM_SECONDS)

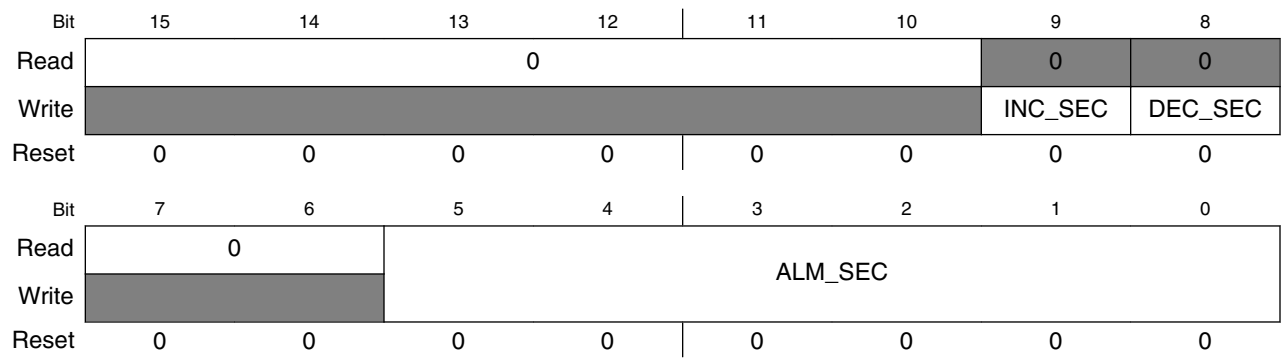
The seconds alarm register is used to configure the seconds setting of the alarm. The alarm setting can be read or written anytime. This register is reset to default value on software reset.

Bits 9:8 provide option to the user software to perform correction on seconds counter to compensate for the leap seconds. Write to these bits adds or subtracts 1 from the seconds counter and read returns zeros.

User software should first determine the state of the INVAL bit in the STATUS (bit 0) to determine that the counters are stable before they can be incremented or decremented. The assertion of INVAL bit ensures that no operation is done at the boundary of a second when counters change value.

User software can configure the type of alarm using the ALM_TYPE bits in the CTRL register.

Address: FFFF_8840h base + Eh offset = FFFF_884Eh



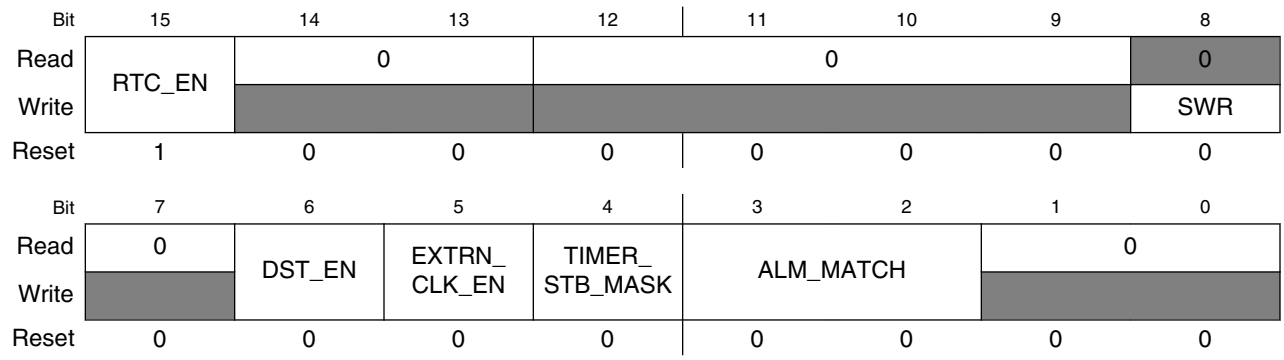
RTC_ALM_SECONDS field descriptions

Field	Description
15–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 INC_SEC	Increment Seconds Counter by 1. This bit controls the increment of seconds counter incase the software wants to make corrections to the seconds counter to compensate for the leap seconds or to perform fine trimming of time when needed. Write to this bit increments the seconds counter and then the bit gets cleared on next posedge.
8 DEC_SEC	Decrement Seconds Counter by 1. This bit controls the decrement of seconds counter incase the software wants to make corrections to the seconds counter to compensate for the leap seconds or to perform fine trimming of time when needed. Write to this bit has decrements the seconds counter and then the bit gets cleared on next posedge of bus clock.
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–0 ALM_SEC	Seconds Value for Alarm. Same as Seconds Counter Value in RTC Seconds Counters Register (RTC_SECONDS) .

23.3.9 RTC Control Register (RTC_CTRL)

This is the control register and governs all operations being done inside the RTC. This register is used to specify the software reset, daylight controls and the type of alarm function needed.

Address: FFFF_8840h base + 10h offset = FFFF_8850h



RTC_CTRL field descriptions

Field	Description
15 RTC_EN	RTC Enable bit. This bit is enabled on reset. Can be used to freeze RTC when desired by software. 0 RTC is disabled. 1 RTC is enabled.
14–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SWR	Software Reset bit. Self clearing bit. Asserting this bit clears the contents of alarm, interrupt (status and enable) registers and has no effect on DST, calendaring, time registers. 0 Software Reset cleared. 1 Software Reset asserted.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 DST_EN	Daylight Saving Enable. The date and time for daylight saving changes are stored in the Daylight Saving Registers. These registers can be changed when this bit is 0. Once this bit is set, those registers cannot be changed and when time and date match the values in those registers, daylight adjustment will happen. To disable Daylight Saving function, this bit should be set to 0.

Table continues on the next page...

RTC_CTRL field descriptions (continued)

Field	Description
	0 Disabled. Daylight saving changes are not applied. Daylight saving registers can be modified. 1 Enabled. Daylight saving changes are applied.
5 EXTRN_CLK_EN	External 1Hz clock enable signal The bitfleid enables/disables the external clock enable signal. 1'b1:- External 1Hz clock is enabled for the calendaring and counters clock. 1'b0:- External 1Hz clock is disabled for the calendaring and counters clock.
4 TIMER_STB_ MASK	Sampling timer clocks mask 1'b0:- Sampling clocks are not gated when in standby mode 1'b1:- Sampling clocks are gated in standby mode
3–2 ALM_MATCH	Alarm Match bits. These bits define the type of alarm function. These bits select which time and calendar counters will be used for matching and generate an alarm. 00 Only Seconds, Minutes, and Hours matched. 01 Only Seconds, Minutes, Hours, and Days matched. 10 Only Seconds, Minutes, Hours, Days, and Months matched. 3 Only Seconds, Minutes, Hours, Days, Months, and Year (offset) matched.
1–0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

23.3.10 RTC Status Register (RTC_STATUS)

This register indicates the status of various processes going inside the RTC. This register also helps the user software to read time or date register when their values are stable and not changing. Bus Error bit get cleared by writing ‘1’ to them. Software Reset resets the whole register to its default state.

Address: FFFF_8840h base + 12h offset = FFFF_8852h

Bit	15	14	13	12	11	10	9	8
Read	0				0	0	0	BUS_ERR
Write								w1c
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0		0	0	0		0	INVAL_BIT
Write								
Reset	0	0	0	0	0	0	0	0

RTC_STATUS field descriptions

Field	Description
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 BUS_ERR	Bus Error bit. This bit indicates that a read or write cycle was initiated by software when the INVAL_BIT was set. Write access to time /date registers gets nullified (terminate normally) and no register value gets changed. Read during INVAL bit asserted returns 16'hFFFF. No Transfer Error is asserted. This bit gets cleared by writing 1 to it. 0 Read and Write accesses are normal. 1 Read or Write accesses occurred when INVAL_BIT was asserted.
7–6 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.
3–2 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 INVAL_BIT	Invalidate CPU read/write access bit. This read-only bit indicates the time /date counters are invalid or changing and thus should not be read/written to. This bit is asserted for 1 oscillator clock cycle before and after the 1 Hz (seconds clock) boundary/edge. Write access to time /date registers gets nullified (terminate normally) and no register value gets changed. Read during INVAL bit asserted returns 0xFFFF. No Transfer Error is asserted. 0 Time /Date Counters can be read/written. Time /Date is valid. 1 Time /Date Counter values are changing or Time /Date is invalid and cannot be read or written.

23.3.11 RTC Interrupt Status Register (RTC_ISR)

NOTE

For the sampling timer interrupt status bits [14:6], refer to chip configuration chapter for the applicable sampling timer frequencies.

This register indicates the status of the various real-time clock interrupts. When an event of the types included in this register occurs then the bit will be set in this register regardless of its corresponding interrupt enable bit being set. The status bits are cleared by writing a value of 1, which also clears the interrupt. Interrupts may occur while the system clock is idle or in standby mode. When the system enters the active power mode, interrupt will be indicated to the CPU. The first event of the Sampling Timer interrupts after Power-on-Reset should not be used to qualify any periodic interval. However, the correct periodic interval (that is 512 Hz or 256 Hz, and so on) should be determined using two sampling timer interrupts. The time between two interrupt would always be the correct time period.

The status register is also cleared on software reset .

NOTE

Sampling interrupts from 512 Hz to 2 Hz are generated using uncompensated clock. Only 1 Hz is compensated.

Address: FFFF_8840h base + 14h offset = FFFF_8854h

Bit	15	14	13	12	11	10	9	8
Read	IS_512HZ	IS_256HZ	IS_128HZ	IS_64HZ	IS_32HZ	IS_16HZ	IS_8HZ	IS_4HZ
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	IS_2HZ	IS_1HZ	MIN_IS	HOUR_IS	DAY_IS	ALM_IS	0	0
Write	w1c	w1c	w1c	w1c	w1c	w1c		
Reset	0	0	0	0	0	0	0	0

RTC_ISR field descriptions

Field	Description
15 IS_512HZ	512 Hz Interval Interrupt Status bit. 0 Interrupt is de-asserted. 1 Interrupt is asserted.
14 IS_256HZ	256 Hz Interval Interrupt Status bit. 0 Interrupt is de-asserted. 1 Interrupt is asserted.
13 IS_128HZ	128 Hz Interval Interrupt Status bit. 0 Interrupt is de-asserted. 1 Interrupt is asserted.
12 IS_64HZ	64 Hz Interval Interrupt Status bit.

Table continues on the next page...

RTC_ISR field descriptions (continued)

Field	Description
	0 Interrupt is de-asserted. 1 Interrupt is asserted.
11 IS_32HZ	32 Hz Interval Interrupt Status bit. 0 Interrupt is de-asserted. 1 Interrupt is asserted.
10 IS_16HZ	16 Hz Interval Interrupt Status bit. 0 Interrupt is de-asserted. 1 Interrupt is asserted.
9 IS_8HZ	8 Hz Interval Interrupt Status bit. 0 Interrupt is de-asserted. 1 Interrupt is asserted.
8 IS_4HZ	4 Hz Interval Interrupt Status bit. 0 Interrupt is de-asserted. 1 Interrupt is asserted.
7 IS_2HZ	2 Hz Interval Interrupt Status bit. 0 Interrupt is de-asserted. 1 Interrupt is asserted.
6 IS_1HZ	1 Hz Interval Interrupt Status bit. 0 Interrupt is de-asserted. 1 Interrupt is asserted.
5 MIN_IS	Minutes Interrupt Status bit. 0 Interrupt is de-asserted. 1 Interrupt is asserted.
4 HOUR_IS	Hours Interrupt Status bit. 0 Interrupt is de-asserted. 1 Interrupt is asserted.
3 DAY_IS	Days Interrupt Status bit. 0 Interrupt is de-asserted. 1 Interrupt is asserted.
2 ALM_IS	Alarm Interrupt Status bit. This bit indicates that the alarm value programmed matches the counter values. 0 Interrupt is de-asserted. 1 Interrupt is asserted.

Table continues on the next page...

RTC_ISR field descriptions (continued)

Field	Description
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.

23.3.12 RTC Interrupt Enable Register (RTC_IER)

For the sampling timer interrupt enable bits [14:6], refer to chip configuration chapter for the applicable sampling timer frequencies.

The real-time clock interrupt enable register (IER) is used to enable/disable the various real-time clock interrupts. De-asserting an interrupt enable bit has no effect on the assertion of its corresponding status bit.

Alarm interrupt is asserted on counters matching the alarm setting done in the memory map. The counters matched for the alarm interrupt are selected based on the Alarm Type set in CTRL[3:2] bits. The various types of alarm available are as per the following table. Only one alarm type can be used at a time.

Table 23-14. Alarm Match Table

ALM_MATCH[1:0] (CTRL[3:2])	Counters Matched	Alarm Type
00	Seconds, Minutes, and Hours	Daily
01	Seconds, Minutes, Hours, and Days	Monthly
10	Seconds, Minutes, Hours, Days, and Months	Yearly
11	Seconds, Minutes, Hours, Days, Months, and Year	One Time

A common interrupt is generated by the block. The user software should read the status register in the interrupt service routine to determine which interrupt has occurred.

Address: FFFF_8840h base + 16h offset = FFFF_8856h

Bit	15	14	13	12	11	10	9	8
Read Write	IE_512Hz	IE_256Hz	IE_128Hz	IE_64Hz	IE_32Hz	IE_16Hz	IE_8Hz	IE_4Hz
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	IE_2Hz	IE_1Hz	MIN_IE	HOUR_IE	DAY_IE	ALM_IE	0	0
Reset	0	0	0	0	0	0	0	0

RTC_IER field descriptions

Field	Description
15 IE_512Hz	512 Hz Interval Interrupt Enable bit. 0 Interrupt is disabled. 1 Interrupt is enabled.
14 IE_256Hz	256 Hz Interval Interrupt Enable bit. 0 Interrupt is disabled. 1 Interrupt is enabled.
13 IE_128Hz	128 Hz Interval Interrupt Enable bit. 0 Interrupt is disabled. 1 Interrupt is enabled.
12 IE_64Hz	64 Hz Interval Interrupt Enable bit. 0 Interrupt is disabled. 1 Interrupt is enabled.
11 IE_32Hz	32 Hz Interval Interrupt Enable bit. 0 Interrupt is disabled. 1 Interrupt is enabled.
10 IE_16Hz	16 Hz Interval Interrupt Enable bit. 0 Interrupt is disabled. 1 Interrupt is enabled.
9 IE_8Hz	8 Hz Interval Interrupt Enable bit. 0 Interrupt is disabled. 1 Interrupt is enabled.
8 IE_4Hz	4 Hz Interval Interrupt Enable bit. 0 Interrupt is disabled. 1 Interrupt is enabled.
7 IE_2Hz	2 Hz Interval Interrupt Enable bit. 0 Interrupt is disabled. 1 Interrupt is enabled.
6 IE_1Hz	1 Hz Interval Interrupt Enable bit. 0 Interrupt is disabled. 1 Interrupt is enabled.
5 MIN_IE	Minutes Interrupt Enable bit. 0 Interrupt is disabled. 1 Interrupt is enabled.

Table continues on the next page...

RTC_IER field descriptions (continued)

Field	Description
4 HOUR_IE	Hours Interrupt Enable bit. 0 Interrupt is disabled. 1 Interrupt is enabled.
3 DAY_IE	Days Interrupt Enable bit. 0 Interrupt is disabled. 1 Interrupt is enabled.
2 ALM_IE	Alarm Interrupt Enable bit. This bit indicates that the alarm value programmed matches the counter values. 0 Interrupt is disabled. 1 Interrupt is enabled.
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.

23.3.13 RTC Daylight Saving Hour Register (RTC_DST_HOUR)

This register stores the time in hours when the Daylight Saving has to be applied or reversed. This register is programmable when the DST_EN bit in CTRL register is ‘Zero’. When DST_EN bit is set, the contents of this register cannot be changed. The user software should program the correct hour value (0 – 23) as per the regional settings. For example, if the Daylight Saving starts at 2:00 AM on March 25 and ends at 2:00 AM on October 28 in 2007 then the time at which the RTC advances or falls back is actually 1:59 AM. Hence the user software should program 1 for the hour count value (and not 2!) i.e. write 0x0101 in this register. 59 minute count is automatically checked inside RTC and hence not required to be programmed. This register has no effect on software reset.

Address: FFFF_8840h base + 22h offset = FFFF_8862h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0			DST_START_HOUR					0			DST_END_HOUR				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RTC_DST_HOUR field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 DST_START_HOUR	Daylight Saving Time (DST) Hours Start Value. This is the hour value for the time when DST comes into effect. Same as Hours Counter Value in the RTC Hours and Minutes Counters Register (RTC_HOURMIN) .
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–0 DST_END_HOUR	Daylight Saving Time (DST) Hours End Value. This is the hour value for the time when DST is reversed. Same as Hours Counter Value in the RTC Hours and Minutes Counters Register (RTC_HOURMIN) .

23.3.14 RTC Daylight Saving Month Register (RTC_DST_MONTH)

This register stores the month when the Daylight Saving has to be applied or reversed. This register is programmable when the DST_EN bit in CTRL register is Zero. When DST_EN bit is set, the contents of this register cannot be changed. The CPU should program the correct month value (1 – 12) as per the regional settings. For example, if the Daylight Saving starts at March 25 and ends at October 28 in 2007. Hence the CPU should write 0x030A in this register. This register has no effect on software reset.

Address: FFFF_8840h base + 24h offset = FFFF_8864h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0				DST_START_MONTH				0				DST_END_MONTH			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RTC_DST_MONTH field descriptions

Field	Description
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–8 DST_START_MONTH	Daylight Saving Time (DST) Month Start Value. This is the month value for the time when DST comes in to effect. See the RTC Year and Month Counters Register (RTC_YEARMON) .
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–0 DST_END_MONTH	Daylight Saving Time (DST) Month End Value. This is the month value for the time when DST is reversed. See the RTC Year and Month Counters Register (RTC_YEARMON) .

23.3.15 RTC Daylight Saving Day Register (RTC_DST_DAY)

This register stores the day when the Daylight Saving has to be applied or reversed. This register is programmable when the DST_EN bit in CTRL register is Zero. When DST_EN bit is set, the contents of this register cannot be changed. The CPU should program the correct day value (1 – 31) as per the regional settings. For example, if the Daylight Saving starts at March 25 and ends at October 28 in 2007. Hence the CPU should write 0x191C in this register. This register is unaffected by software reset.

Address: FFFF_8840h base + 26h offset = FFFF_8866h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0			DST_START_DAY					0			DST_END_DAY				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RTC_DST_DAY field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 DST_START_DAY	Daylight Saving Time (DST) Day Start Value. This is the day value for the time when DST comes into effect.
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–0 DST_END_DAY	Daylight Saving Time (DST) Day End Value. This is the day value for the time when DST is reversed.

23.3.16 RTC Prescaler Register (RTC_PRESCALER)

When the Compensation Block is not used, hence in these modes, the Prescaler Block generates the 1 Hz clock. Since this register controls the generation of 1 Hz clock and also provides a coarse compensation, hence the address offset value of this register is same as that of the RTC Compensation Register. The value programmed in this register can be modified by CPU in the event of change in oscillator clock frequency to perform some limited compensation.

Details on using this register can be found in RTC Prescaler.

Address: FFFF_8840h base + 28h offset = FFFF_8868h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	PRSCVAL_H								PRSCVAL_L							
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RTC_PRESCALER field descriptions

Field	Description
15–8 PRSCVAL_H	RTC Prescaler Count (Higher Byte) Value This value is used to compare against the value of the 1 Hz generation prescaler counter to generate 1 Hz clock. This register is used to store the Frequency of the 32.768 kHz in Hz and thus we can take care of any variation in frequency that might occur and have a near accurate 1 Hz clock.
7–0 PRSCVAL_L	RTC Prescaler Count (Lower Byte) Value This value is used to compare against the value of the 1 Hz generation prescaler counter to generate 1 Hz clock. This register is used to store the Frequency of the 32.768 kHz in Hz and thus we can take care of any variation in frequency that might occur and have a near accurate 1 Hz clock.

23.4 Functional Description

23.4.1 Time and Calendaring Functions

RTC performs and controls all chronological functions as mentioned below:

- Implements all counters for date and time and their related control logic
- Leap Year calculation and adjusting the day count accordingly
- Increment/Decrement of counters for adjustment of leap seconds
- Tracking the number of days in a month
- Automatic Daylight adjustment for time
- Alarm generation with selectable matching of different counters

Dynamic modifications to the date counters are done based on a) Leap Year b) Month and c) Daylight Saving. Additionally, the user software can add or subtract a second to take care of Leap Seconds Adjustment. All changes are hardware controlled and triggered by software. A leap year is defined as a year in which the year value is divisible by 4 and 400 and will have an extra day in February. Daylight savings are done in different regions of the world to shift the local time according to summer or winters. Time is shifted at a pre-defined time decided by the regional conditions and programmed by the user software into the RTC and RTC automatically adjusts the time using hardware alarms for daylight saving. Day counter is also adjusted for months having 28/29/30/31 days.

Alarm generation is done by the RTC block. The matching of counters is controlled by ALM_MATCH bits inside the control register to give various alarm options of daily, monthly, yearly or one-time alarms.

Note

Setting an hour alarm value that coincides with the Daylight Saving Start Hour value will not generate an alarm as when Daylight Saving comes into effect, the hour counter is incremented by 2 instead or normal 1. For example, setting ALM_HOURMIN = 0x1500 (hour alarm value = 15 or 3 PM) and DST_HOUR = 0x1411 (DST Start Hour value = 14 or 2 PM), will cause the hour counter to go from 14 to 16. The hour counter will not be 15 and hence the alarm which was set for 15 (or 3 PM) will not trigger. User software must take into account Daylight Saving changes when setting alarm values.

NOTE

The user software must program an alarm time equal to the daylight saving end time that is fallback time and when the alarm interrupt comes, the user software must clear the DSTEN bit in control register, else correct operation might not occur.

23.4.2 RTC Prescaler

This block generates the sampling timer frequencies and 1 Hz clock for the time keeping functions. The 1 Hz clock is generated when the prescaler counter value matches the value set in the PRESCALER register. Thus, the 1 Hz clock has a duty cycle of 1/32768. The user software must program the observed frequency (in Hz) of the 32.768 kHz clock input to the MCU, in the PRESCALER register. The user software can compensate for variations in the 32.768 kHz clock by periodically monitoring the 32.768 kHz clock any change in the oscillator frequency clock can be programmed to the closest 1 Hz value to provide a near accurate 1 Hz clock.Details on how compensation can be performed is described in "Compensation using the Prescaler Block".

The Prescaler block also includes a 512 Hz counter that counts on the 32.768 kHz clock to generate a 512 Hz clock output. The Sampling Timer counter runs on the 512 Hz clock and generates the frequencies from 512 Hz, 256 Hz and so on till 2 Hz which assert corresponding interrupt status bit in the ISR register. These bits will in turn set the RTC interrupt if corresponding interrupt enable is set.

23.4.2.1 Compensation using the Prescaler Block

23.4.2.1.1 Overview

The value of PRESCALER register represents the number of ticks of the input 32.768 kHz clock to RTC that represent a 1 Hz clock. When the board is able to provide an exact 32.768 kHz clock the number of ticks needed to create an accurate 1 Hz clock is 32768. This is also the reset value of this register.

However due to inherent error in the crystal used on board the clock reaching the design may not be accurately 32.768 kHz. Such variations lead to an inaccurate clock counted by RTC.

If the user is able to determine the exact frequency from crystal then it is easy to determine the value of the PRESCALER register. This is done as follows:

For example,

Actual crystal frequency 32767 Hz => PRESCALER = 32767

Actual crystal frequency 32768.8 Hz => PRESCALER = 32769 (rounded off)

User can program the closest absolute value to the value of crystal frequency in Hz. RTC cannot compensate less than 1 Hz error in crystal frequency in Basic & Standard Feature Sets.

23.4.2.1.2 Error Detection & Correction

If the user is seeing a deviation of the seconds counted by the RTC vs. the actual seconds counted then the user can determine the compensated value (i.e. value for the PRESCALER register) as follows; assuming the deviation is seen with PRESCALER=32768 (Reset value).

For example,

Expected seconds counted (From accurate clock source): 86400

Actual seconds counted (From RTC seconds register): 86397

Since PRESCALER=32768, it means only 32768*86397 pulses of the current supplied clock represent real 86400 seconds.

With ideal clock frequency = 32768 Hz, Current clock frequency = 32768*86397/86400 Hz = 32766.86 Hz

Hence, the closest value of PRESCALER to compensate for above loss would be 32767. The PPM error would still be 4.20479 PPM (less than 20 PPM) and the RTC time would be slightly more than accurate real time.

An accurate crystal can help user further lower the PPM error in clock and must be validated on the board used.

23.4.3 RTC Isolation

To prevent leakage and erratic behavior, RTC isolates its CPU register programming interface and other control signals. There are two levels of isolation provided:

- CPU is in certain low power modes where LVD is disabled (except VLPR).
- When VDD < LVD, both read and writes to all registers are blocked. Isolation is enabled to prevent leakage from signals coming from the powered off domain of SoC.

The voltage threshold detection is done by an analog block (outside digital RTC) which monitors the voltage levels.

Chapter 24
Flash Memory Controller (FMC)

24.1 Introduction

The Flash Memory Controller (FMC) is a memory acceleration unit that provides:

- an interface between the chip and the 32-bit program flash memory and FlexMemory (FlexNVM and FlexRAM used as EEPROM).
- a buffer and a cache that can accelerate program flash memory data transfers.

24.1.1 Overview

The Flash Memory Controller manages the interface between the chip and the 32-bit program flash memory and FlexMemory (FlexNVM as well as FlexRAM used as EEPROM). The FMC receives status information detailing the configuration of the flash memory and FlexMemory 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 and FlexNVM used as data flash memory. A write operation to program flash or FlexNVM used as data flash memory results in a bus error. The FMC interface to FlexNVM and FlexRAM when they are used as EEPROM allows both read and write 8-bit, 16-bit, and 32-bit operations.

In addition, for program flash memory, the FMC provides two separate mechanisms for accelerating the interface between the device and the flash. A 32-bit speculation buffer can prefetch the next 32-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.

24.1.2 Features

The FMC's features include:

- Interface between the device and the 32-bit program flash memory and FlexMemory:
  - 8-bit, 16-bit, and 32-bit read operations to nonvolatile memory (flash and FlexNVM used as data flash memory).
  - 8-bit, 16-bit, and 32-bit read and write operations to the FlexNVM and FlexRAM used as EEPROM.
  - Consecutive read accesses (such as 0x0,0x4) to program flash memory return the second read data with no wait states when the buffer or cache is enabled. The memory returns 32 bits via the 32-bit bus access.
- Acceleration of data transfer from the program flash memory to the device:
  - 32-bit prefetch speculation buffer for program flash accesses for crossbar switch master 0 (V1 ColdFire CPU) with controls for instruction/data access
  - 4-way, 4-set, 32-bit line size program flash memory cache for a total of sixteen 32-bit entries with invalidation control

24.2 Modes of operation

The FMC operates only when the chip accesses the program flash memory or FlexMemory. 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 or FlexMemory cannot be accessed, the FMC is disabled.

24.3 External signal description

The FMC has no external (off-chip) signals.

24.4 Memory map and register descriptions

The ColdFire CPU's programming model provides control and configuration of the flash controller's speculation and cache functions with bits in the CPU Control Register (CPUCR). For details, see the description of the CPUCR register.

NOTE

Program the Flash Memory Controller's configuration and control settings only while the Flash Memory Controller is idle. Changing settings while a flash access is in progress can lead to non-deterministic behavior.

NOTE

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

24.5 Functional description

The FMC is a flash acceleration unit with flexible buffers for user configuration. Besides managing the interface between the chip and the program flash memory and FlexMemory, 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. (The basic flash access time is two processor cycles.)

Upon system reset, the FMC is configured to provide buffering for transfers from the program flash memory. Prefetch support for data and instructions is enabled for program flash accesses for crossbar switch master 0 (V1 ColdFire CPU). The program flash memory cache is enabled and configured for instruction replacement.

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 controls enable buffering per access type (data or instruction). When reconfiguring the FMC, do not program the control and configuration inputs to the FMC while the program flash memory or FlexMemory is being accessed. Instead, change them with a routine executing from RAM in supervisor mode.

Chapter 25
Flash Memory Module (FTFL)

25.1 Introduction

The flash memory module includes the following accessible memory regions:

- Program flash memory for vector space and code store
- FlexNVM for data store and additional code store
- FlexRAM for high-endurance data store or traditional RAM

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.

25.1.1 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.

25.1.1.1 Program Flash Memory Features

- Sector size of 1 KB
- Program flash protection scheme prevents accidental program or erase of stored data
- Automated, built-in, program and erase algorithms with verify
- Section programming for faster bulk programming times
- Read access to one logical program flash block is possible while programming or erasing data in the other logical program flash block
- Read access to program flash memory possible while programming or erasing data in the data flash memory or FlexRAM

25.1.1.2 FlexNVM Memory Features

When FlexNVM is partitioned for data flash memory:

- Sector size of 1 KB
- Protection scheme prevents accidental program or erase of stored data
- Automated, built-in program and erase algorithms with verify
- Section programming for faster bulk programming times
- Read access to data flash memory possible while programming or erasing data in the program flash memory

25.1.1.3 FlexRAM Features

- Memory that can be used as traditional RAM or as high-endurance EEPROM storage

- Up to 2 KB of FlexRAM configured for EEPROM or traditional RAM operations
- When configured for EEPROM:
  - Protection scheme prevents accidental program or erase of data written for EEPROM
  - Built-in hardware emulation scheme to automate EEPROM record maintenance functions
  - Programmable EEPROM data set size and FlexNVM partition code facilitating EEPROM memory endurance trade-offs
  - Supports FlexRAM aligned writes of 1, 2, or 4 bytes at a time
  - Read access to FlexRAM possible while programming or erasing data in the program or data flash memory
- When configured for traditional RAM:
  - Read and write access possible to the FlexRAM while programming or erasing data in the program or data flash memory

25.1.1.4 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

25.1.2 Block Diagram

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

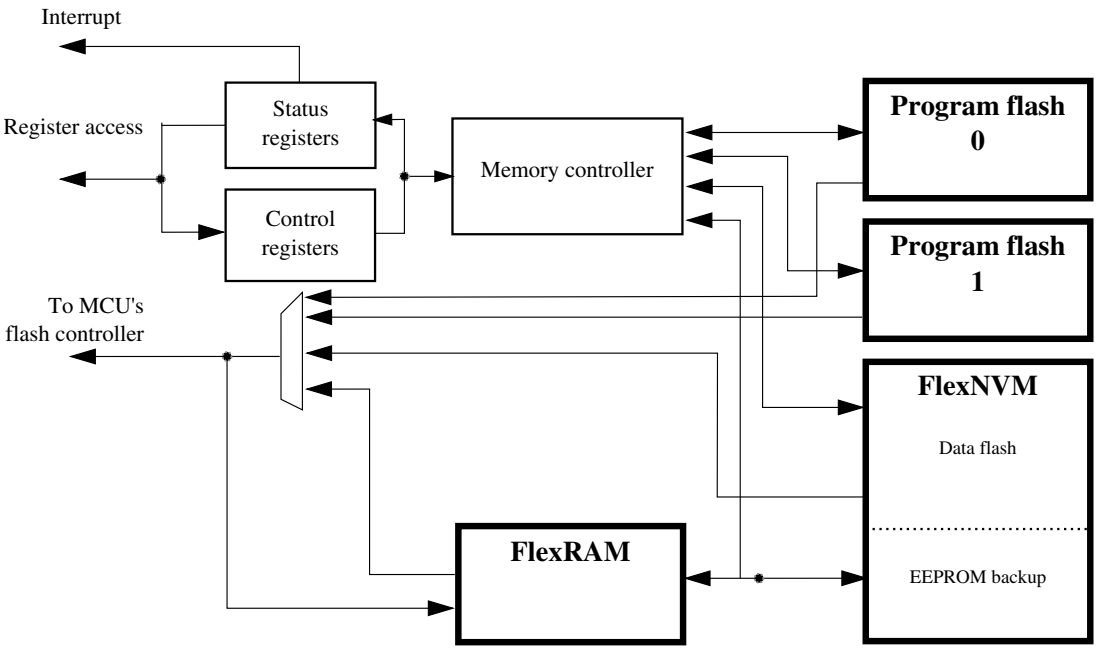


Figure 25-1. Flash Block Diagram

25.1.3 Glossary

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

Data flash memory — Partitioned from the FlexNVM block, the data flash memory provides nonvolatile storage for user data, boot code, and additional code store.

Data flash sector — The data flash sector is the smallest portion of the data flash memory that can be erased.

EEPROM — Using a built-in filing system, the flash memory module emulates the characteristics of an EEPROM by effectively providing a high-endurance, byte-writeable (program and erase) NVM.

EEPROM backup data header — The EEPROM backup data header is comprised of a 32-bit field found in EEPROM backup data memory which contains information used by the EEPROM filing system to determine the status of a specific EEPROM backup flash sector.

EEPROM backup data record — The EEPROM backup data record is comprised of a 2-bit status field, a 14-bit address field, and a 16-bit data field found in EEPROM backup data memory which is used by the EEPROM filing system. If the status field indicates a record is valid, the data field is mirrored in the FlexRAM at a location determined by the address field.

EEPROM backup data memory — Partitioned from the FlexNVM block, EEPROM backup data memory provides nonvolatile storage for the EEPROM filing system representing data written to the FlexRAM requiring highest endurance.

EEPROM backup data sector — The EEPROM backup data sector contains one EEPROM backup data header and up to 255 EEPROM backup data records, which are used by the EEPROM filing system.

Endurance — The number of times that a flash memory location can be erased and reprogrammed.

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

Flash block — A macro within the flash memory module which provides the nonvolatile memory storage.

FlexMemory — Flash configuration that supports data flash, EEPROM, and FlexRAM.

FlexNVM Block — The FlexNVM block can be configured to be used as data flash memory, EEPROM backup flash memory, or a combination of both.

FlexRAM — The FlexRAM refers to a RAM, dedicated to the flash memory module, that can be configured to store EEPROM data or as traditional RAM. When configured for EEPROM, valid writes to the FlexRAM generate new EEPROM backup data records stored in the EEPROM backup flash memory.

Flash Memory Module — All flash blocks plus a flash management unit providing high-level control and an interface to MCU buses.

IFR — Nonvolatile information register found in each flash block, separate from the main memory array.

NVM — Nonvolatile memory. A memory technology that maintains stored data during power-off. The flash array is an NVM using NOR-type flash memory technology.

NVM Normal Mode — An NVM mode that provides basic user access to 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.

Longword — 32 bits of data with an aligned longword having byte-address[1:0] = 00.

Word — 16 bits of data with an aligned word having byte-address[0] = 0.

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

Program flash Sector — The smallest portion of the program flash memory (consecutive addresses) that can be erased.

Retention — The length of time that data can be kept in the NVM without experiencing errors upon readout. Since erased (1) states are subject to degradation just like programmed (0) states, the data retention limit may be reached from the last erase operation (not from the programming time).

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

Section Program Buffer — Lower half of the FlexRAM allocated for storing large amounts of data for programming via the Program Section command.

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.

25.2 External Signal Description

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

25.3 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.

25.3.1 Flash Configuration Field Description

The program flash memory contains a 16-byte flash configuration field that stores default protection settings (loaded on reset) and security information that allows the MCU to restrict access to the flash memory module.

Flash Configuration Field Byte Address	Size (Bytes)	Field Description
0x0_0400 - 0x0_0407	8	Backdoor Comparison Key. Refer to Verify Backdoor Access Key Command and Unsecuring the 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_040C	1	Data flash protection byte. Refer to the description of the Data Flash Protection Register (FDPROT).
0x0_040D	1	EEPROM protection byte. Refer to the description of the EEPROM Protection Register (FEPROT).
0x0_040E	1	Flash nonvolatile option byte. Refer to the description of the Flash Option Register (FOPT).
0x0_040F	1	Flash security byte. Refer to the description of the Flash Security Register (FSEC).

25.3.2 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 following table and further described in the subsequent paragraphs.

Address Range	Size (Bytes)	Field Description
0x00 – 0xBF	192	Reserved
0xC0 – 0xFF	64	Program Once Field

25.3.2.1 Program Once Field

The Program Once Field in the program flash IFR provides 64 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 records using the Read Once and Program Once commands (see Read Once Command and Program Once Command).

25.3.3 Data Flash IFR Map

The data flash IFR is a 256 byte nonvolatile information memory that can be read and erased, but the user has limited program capabilities in the data flash IFR (see the Program Partition command in Program Partition Command, the Erase All Blocks command in Erase All Blocks Command, and the Read Resource command in Read Resource Command). The contents of the data flash IFR are summarized in the following table and further described in the subsequent paragraphs.

Address Range	Size (Bytes)	Field Description
0x00 – 0xFD	254	Reserved
0xFE	1	EEPROM data set size
0xFF	1	FlexNVM partition code

25.3.3.1 EEPROM Data Set Size

The EEPROM data set size byte in the data flash IFR supplies information which determines the amount of FlexRAM used in each of the available EEPROM subsystems. To program the EEESPLIT and EEESIZE values, see the Program Partition command described in Program Partition Command.

Table 25-1. EEPROM Data Set Size

Data flash IFR: 0x00FE							
7	6	5	4	3	2	1	0
1	1	EEESPLIT		EEESIZE			
	= Unimplemented or Reserved						

Table 25-2. EEPROM Data Set Size Field Description

Field	Description
7-6 Reserved	This read-only bitfield is reserved and must always be written as one.
5-4 EEESPLIT	**EEPROM Split Factor** — Determines the relative sizes of the two EEPROM subsystems. ‘00’ = Subsystem A: EEESIZE*1/8, subsystem B: EEESIZE*7/8 ‘01’ = Subsystem A: EEESIZE*1/4, subsystem B: EEESIZE*3/4 ‘10’ = Subsystem A: EEESIZE*1/2, subsystem B: EEESIZE*1/2 ‘11’ = Subsystem A: EEESIZE*1/2, subsystem B: EEESIZE*1/2
3-0 EEESIZE	**EEPROM Size** — Encoding of the total available FlexRAM for EEPROM use. **NOTE:** EEESIZE must be 0 bytes (1111b) when the FlexNVM partition code (FlexNVM Partition Code) is set to 'No EEPROM'. '0000' = Reserved '0001' = Reserved '0010' = Reserved '0011' = 2,048 Bytes '0100' = 1,024 Bytes '0101' = 512 Bytes '0110' = 256 Bytes '0111' = 128 Bytes '1000' = 64 Bytes '1001' = 32 Bytes '1010' = Reserved '1011' = Reserved '1100' = Reserved '1101' = Reserved '1110' = Reserved '1111' = 0 Bytes

25.3.3.2 FlexNVM Partition Code

The FlexNVM Partition Code byte in the data flash IFR supplies a code which specifies how to split the FlexNVM block between data flash memory and EEPROM backup memory supporting EEPROM functions. To program the DEPART value, see the Program Partition command described in Program Partition Command.

Table 25-3. FlexNVM Partition Code

Data Flash IFR: 0x00FF	

Table continues on the next page...

Table 25-3. FlexNVM Partition Code (continued)

7	6	5	4	3	2	1	0
1	1	1	1	DEPART			
	= Unimplemented or Reserved						

Table 25-4. FlexNVM Partition Code Field Description

Field	Description
7-4 Reserved	This read-only bitfield is reserved and must always be written as one.
3-0 DEPART	**FlexNVM Partition Code** — Encoding of the data flash / EEPROM backup split within the FlexNVM memory block. FlexNVM memory not partitioned for data flash will be used to store EEPROM records. 0000 = 32 KB of data flash, No EEPROM backup (No EEPROM) 0001 = 24 KB of data flash, 8 KB of EEPROM backup 0010 = 16 KB of data flash, 16 KB of EEPROM backup 0011 = No data flash, 32 KB of EEPROM backup 0100 = Reserved 0101 = Reserved 0110 = Reserved 0111 = Reserved 1000 = No data flash, 32 KB of EEPROM backup 1001 = 8 KB of data flash, 24 KB of EEPROM backup 1010 = 16 KB of data flash, 16 KB of EEPROM backup 1011 = 32 KB of data flash, No EEPROM backup (No EEPROM) 1100 = Reserved 1101 = Reserved 1110 = Reserved 1111 = Reserved (defaults to 32 KB of data flash, No EEPROM)

25.3.4 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).

FTFL memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_8380	Flash Option Register (FTFL_FOPT)	8	R	Undefined	25.34.1/450
FFFF_8381	Flash Security Register (FTFL_FSEC)	8	R	Undefined	25.34.2/450
FFFF_8382	Flash Configuration Register (FTFL_FCNFG)	8	R/W	See section	25.34.3/452
FFFF_8383	Flash Status Register (FTFL_FSTAT)	8	R/W	00h	25.34.4/454
FFFF_8384	Flash Common Command Object Registers (FTFL_FCCOB0)	8	R/W	00h	25.34.5/455
FFFF_8385	Flash Common Command Object Registers (FTFL_FCCOB1)	8	R/W	00h	25.34.5/455
FFFF_8386	Flash Common Command Object Registers (FTFL_FCCOB2)	8	R/W	00h	25.34.5/455
FFFF_8387	Flash Common Command Object Registers (FTFL_FCCOB3)	8	R/W	00h	25.34.5/455
FFFF_8388	Flash Common Command Object Registers (FTFL_FCCOB4)	8	R/W	00h	25.34.5/455
FFFF_8389	Flash Common Command Object Registers (FTFL_FCCOB5)	8	R/W	00h	25.34.5/455
FFFF_838A	Flash Common Command Object Registers (FTFL_FCCOB6)	8	R/W	00h	25.34.5/455
FFFF_838B	Flash Common Command Object Registers (FTFL_FCCOB7)	8	R/W	00h	25.34.5/455
FFFF_838C	Flash Common Command Object Registers (FTFL_FCCOB8)	8	R/W	00h	25.34.5/455
FFFF_838D	Flash Common Command Object Registers (FTFL_FCCOB9)	8	R/W	00h	25.34.5/455
FFFF_838E	Flash Common Command Object Registers (FTFL_FCCOBA)	8	R/W	00h	25.34.5/455
FFFF_838F	Flash Common Command Object Registers (FTFL_FCCOBB)	8	R/W	00h	25.34.5/455
FFFF_8390	Program Flash Protection Registers (FTFL_FPROT0)	8	R/W	Undefined	25.34.6/456
FFFF_8391	Program Flash Protection Registers (FTFL_FPROT1)	8	R/W	Undefined	25.34.6/456
FFFF_8392	Program Flash Protection Registers (FTFL_FPROT2)	8	R/W	Undefined	25.34.6/456
FFFF_8393	Program Flash Protection Registers (FTFL_FPROT3)	8	R/W	Undefined	25.34.6/456
FFFF_8394	Data Flash Protection Register (FTFL_FDPROT)	8	R/W	Undefined	25.34.7/458
FFFF_8395	EEPROM Protection Register (FTFL_FEPROT)	8	R/W	Undefined	25.34.8/459

25.34.1 Flash Option Register (FTFL_FOPT)

The flash option register allows the MCU to customize its operations by examining the state of these read-only bits, which are loaded from NVM at reset. The function of the bits is defined in the device's Chip Configuration details.

All bits in the register are read-only .

During the reset sequence, the register is loaded from the flash nonvolatile option byte in the Flash Configuration Field located in program flash memory. The flash basis for the values is signified by X in the reset value.

Address: FFFF_8380h base + 0h offset = FFFF_8380h

Bit	7	6	5	4	3	2	1	0
Read	OPT							
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FTFL_FOPT field descriptions

Field	Description
7–0 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.

25.34.2 Flash Security Register (FTFL_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: FFFF_8380h base + 1h offset = FFFF_8381h

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.

FTFL_FSEC field descriptions

Field	Description
7–6 KEYEN	Backdoor Key Security Enable These bits enable and disable 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 Bits Enables and disables mass erase capability of the flash memory module. The state of the MEEN bits is only relevant when the SEC bits are set to secure outside of NVM Normal Mode. When the SEC field is set to unsecure, the MEEN setting does not matter. 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 These bits enable or disable access to the flash memory contents during returned part failure analysis at Freescale. When SEC is secure and FSLACC is denied, access to the program flash contents is denied and any failure analysis performed by Freescale factory test must begin with a full erase to unsecure the part. When access is granted (SEC is unsecure, or SEC is secure and FSLACC is granted), Freescale factory testing has visibility of the current flash contents. The state of the FSLACC bits is only relevant when the SEC bits are set to secure. When the SEC field is set to unsecure, the FSLACC setting does not matter. 00 Freescale factory access granted 01 Freescale factory access denied 10 Freescale factory access denied 11 Freescale factory access granted
1–0 SEC	Flash Security These bits define 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, the SEC bits are 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

25.34.3 Flash Configuration Register (FTFL_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. SWAP,PFLSH, RAMRDY, and EEERDY are read-only status bits . The unassigned bits read as noted and are not writable. The reset values for the SWAP, PFLASH, RAMRDY , and EEERDY bits are determined during the reset sequence.

Address: FFFF_8380h base + 2h offset = FFFF_8382h

Bit	7	6	5	4	3	2	1	0
Read	CCIE	RDCOLLIE	ERSAREQ	ERSSUSP	SWAP	PFLSH	RAMRDY	EEERDY
Write								
Reset	0	0	0	0	*	*	*	*

* Notes:
- SWAP field: Device specific value
- PFLSH field: Device specific value
- RAMRDY field: Device specific value
- EEERDY field: Device specific value

FTFL_FCNFG field descriptions

Field	Description
7 CCIE	Command Complete Interrupt Enable The CCIE bit 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 The RDCOLLIE bit 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 This bit issues a request to the memory controller to execute the Erase All Blocks command and release security. ERSAREQ is not directly writable but is under indirect user control. Refer to the device's Chip Configuration details on how to request this command. The ERSAREQ bit 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.

Table continues on the next page...

FTFL_FCNFG field descriptions (continued)

Field	Description
	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 The ERSSUSP bit 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 SWAP	Swap For program flash only configurations, the SWAP flag indicates which physical program flash block is located at relative address 0x0000. The state of the SWAP flag is set by the flash memory module during the reset sequence. See the Swap Control command section for information on swap management. 0 Physical program flash 0 is located at relative address 0x0000 1 If the PFLSH flag is set, physical program flash 1 is located at relative address 0x0000. If the PFLSH flag is not set, physical program flash 0 is located at relative address 0x0000
2 PFLSH	Flash memory configuration 0 Flash memory module configured for FlexMemory that supports data flash and/or EEPROM. 1 Reserved.
1 RAMRDY	RAM Ready This flag indicates the current status of the FlexRAM . The state of the RAMRDY flag is normally controlled by the Set FlexRAM Function command. During the reset sequence, the RAMRDY flag is cleared if the FlexNVM block is partitioned for EEPROM and is set if the FlexNVM block is not partitioned for EEPROM. The RAMRDY flag is cleared if the Program Partition command is run to partition the FlexNVM block for EEPROM. The RAMRDY flag sets after completion of the Erase All Blocks command or execution of the erase-all operation triggered external to the flash memory module. 0 FlexRAM is not available for traditional RAM access. 1 FlexRAM is available as traditional RAM only; writes to the FlexRAM do not trigger EEPROM operations.
0 EEERDY	This flag indicates if the EEPROM backup data has been copied to the FlexRAM and is therefore available for read access. During the reset sequence, the EEERDY flag will remain cleared while CCIF is clear and will only set if the FlexNVM block is partitioned for EEPROM. 0 FlexRAM is not available for EEPROM operation. 1 FlexRAM is available for EEPROM operations where: • reads from the FlexRAM return data previously written to the FlexRAM in EEPROM mode and • writes to the FlexRAM clear EEERDY and launch an EEPROM operation to store the written data in the FlexRAM and EEPROM backup.

25.34.4 Flash Status Register (FTFL_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 or writes to the FlexRAM (when EEERDY is set) until the flag is cleared (by writing a one to it).

Address: FFFF_8380h base + 3h offset = FFFF_8383h

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

FTFL_FSTAT field descriptions

Field	Description
7 CCIF	Command Complete Interrupt Flag The CCIF flag indicates that a flash command or EEPROM file system operation has completed. The CCIF flag is cleared by writing a 1 to CCIF to launch a command, and CCIF stays low until command completion or command violation. The CCIF flag is also cleared by a successful write to FlexRAM while enabled for EEE, and CCIF stays low until the EEPROM file system has created the associated EEPROM data record. The CCIF bit is reset to 0 but is set to 1 by the memory controller at the end of the reset initialization sequence. Depending on how quickly the read occurs after reset release, the user may or may not see the 0 hardware reset value. 0 Flash command or EEPROM file system operation in progress 1 Flash command or EEPROM file system operation has completed
6 RDCOLERR	Flash Read Collision Error Flag The RDCOLERR error bit 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 The ACCERR error bit 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

Table continues on the next page...

FTFL_FSTAT field descriptions (continued)

Field	Description
	CCIF flag cannot be cleared to launch a command. The ACCERR bit is cleared by writing a 1 to it. Writing a 0 to the ACCERR bit has no effect. 0 No access error detected 1 Access error detected
4 FPVIOL	Flash Protection Violation Flag The FPVIOL error bit indicates an attempt was made to program or erase an address in a protected area of program flash or data flash memory during a command write sequence or a write was attempted to a protected area of the FlexRAM while enabled for EEPROM. While FPVIOL is set, the CCIF flag cannot be cleared to launch a command. The FPVIOL bit is cleared by writing a 1 to it. Writing a 0 to the FPVIOL bit has no effect. 0 No protection violation detected 1 Protection violation detected
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 bit 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.

25.34.5 Flash Common Command Object Registers (FTFL_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: FFFF_8380h base + 4h offset + (1d × i), where i=0d to 11d

Bit	7	6	5	4	3	2	1	0
Read Write	CCOBn							
Reset	0	0	0	0	0	0	0	0

FTFL_FCCOB*n* field descriptions

Field	Description
7–0 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

FTFL_FCCOB*n* field descriptions (continued)

Field	Description
	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. 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.

FCCOB Number	Typical Command Parameter Contents [7:0]
0	FCMD (a code that defines the flash command)
1	Flash address [23:16]
2	Flash address [15:8]
3	Flash address [7:0]
4	Data Byte 0
5	Data Byte 1
6	Data Byte 2
7	Data Byte 3
8	Data Byte 4
9	Data Byte 5
A	Data Byte 6
B	Data Byte 7

FCCOB Endianness and Multi-Byte Access :

The FCCOB register group uses a big endian addressing convention. For all command parameter fields larger than one 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).

25.34.6 Program Flash Protection Registers (FTFL_FPROT*n*)

The FPROT registers define which logical program flash regions are protected from program and erase operations. Protected flash regions cannot have their content changed; that is, these regions cannot be programmed and cannot be erased by any flash command. Unprotected regions can be changed by program and erase operations.

The four FPROT registers allow 32 protectable regions. Each bit protects a 1/32 region of the program flash memory . 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: FFFF_8380h 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.

FTFL_FPROT*n* field descriptions

Field	Description
7–0 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. **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.

FTFL_FPROT*n* field descriptions (continued)

Field	Description
	Each bit in the 32-bit protection register represents 1/32 of the total program flash. 0 Program flash region is protected. 1 Program flash region is not protected

25.34.7 Data Flash Protection Register (FTFL_FDPROT)

The FDPROT register defines which data flash regions are protected against program and erase operations. Protected Flash regions cannot have their content changed; that is, these regions cannot be programmed and cannot be erased by any flash command. Unprotected regions can be changed by both program and erase operations.

Address: FFFF_8380h base + 14h offset = FFFF_8394h

Bit	7	6	5	4	3	2	1	0
Read Write	DPROT							
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FTFL_FDPROT field descriptions

Field	Description
7–0 DPROT	Data Flash Region Protect Individual data flash regions can be protected from program and erase operations by setting the associated DPROT bit. Each DPROT bit protects one-eighth of the partitioned data flash memory space. The granularity of data flash protection cannot be less than the data flash sector size. If an unused DPROT bit is set, the Erase all Blocks command does not execute and the FSTAT[FPVIOL] flag is set. **In NVM Normal mode:** The protection can only be increased, meaning that currently unprotected memory can be protected but currently protected memory cannot be unprotected. Since unprotected regions are marked with a 1 and protected regions use a 0, only writes changing 1s to 0s are accepted. This 1-to-0 transition check is performed on a bit-by-bit basis. Those FDPROT bits with 1-to-0 transitions are accepted while all bits with 0-to-1 transitions are ignored. **In NVM Special mode:** All bits of the FDPROT register are writable without restriction. Unprotected areas can be protected and protected areas can be unprotected. **Restriction:** The user must never write to the FDPROT register while a command is running (CCIF=0). **Reset:** During the reset sequence, the FDPROT register is loaded with the contents of the data flash protection byte in the Flash Configuration Field located in program flash memory. The flash basis for the reset values is signified by X in the register diagram. To change the data flash protection that will be loaded during the reset sequence, unprotect the sector of program flash that contains the Flash Configuration Field. Then, erase and reprogram the data flash protection byte.

FTFL_FDPROT field descriptions (continued)

Field	Description
	Trying to alter data with the program and erase commands in any protected area in the data flash memory results in a protection violation error and sets the FSTAT[FPVIOL] bit. A full block erase of the data flash memory (see the Erase Flash Block command description) is not possible if the data flash memory contains any protected region or if the FlexNVM block has been partitioned for EEPROM. 0 Data Flash region is protected 1 Data Flash region is not protected

25.34.8 EEPROM Protection Register (FTFL_FEPROT)

The FEPROT register defines which EEPROM regions of the FlexRAM are protected against program and erase operations. Protected EEPROM regions cannot have their content changed by writing to it. Unprotected regions can be changed by writing to it. Unprotected regions can be changed by writing to the FlexRAM.

Address: FFFF_8380h base + 15h offset = FFFF_8395h

Bit	7	6	5	4	3	2	1	0
Read Write	EPROT							
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FTFL_FEPROT field descriptions

Field	Description
7–0 EPROT	EEPROM Region Protect Individual EEPROM regions can be protected from alteration by setting the associated EPROT bit. The EPROT bits are not used when the FlexNVM Partition Code is set to data flash only. When the FlexNVM Partition Code is set to data flash and EEPROM or EEPROM only, each EPROT bit covers one-eighth of the configured EEPROM data (see the EEPROM Data Set Size parameter description). **In NVM Normal mode:** The protection can only be increased. This means that currently-unprotected memory can be protected, but currently-protected memory cannot be unprotected. Since unprotected regions are marked with a 1 and protected regions use a 0, only writes changing 1s to 0s are accepted. This 1-to-0 transition check is performed on a bit-by-bit basis. Those FEPROT bits with 1-to-0 transitions are accepted while all bits with 0-to-1 transitions are ignored. **In NVM Special mode :** All bits of the FEPROT register are writable without restriction. Unprotected areas can be protected and protected areas can be unprotected. **Restriction:** Never write to the FEPROT register while a command is running (CCIF=0).

FTFL_FEPROT field descriptions (continued)

Field	Description
	Reset: During the reset sequence, the FEPROT register is loaded with the contents of the FlexRAM protection byte in the Flash Configuration Field located in program flash. The flash basis for the reset values is signified by X in the register diagram. To change the EEPROM protection that will be loaded during the reset sequence, the sector of program flash that contains the Flash Configuration Field must be unprotected; then the EEPROM protection byte must be erased and reprogrammed. Trying to alter data by writing to any protected area in the EEPROM results in a protection violation error and sets the FPVIOL bit in the FSTAT register. 0 EEPROM region is protected 1 EEPROM region is not protected

25.4 Functional Description

The following sections describe functional details of the flash memory module.

25.4.1 Program Flash Memory Swap

The user can configure the logical memory map of the program flash space such that either of the two physical program flash blocks can exist at relative address 0x0000. This swap feature enables the lower half of the logical program flash space to be operational while the upper half is being updated for future use.

The Swap Control command handles swapping the two logical P-Flash memory blocks within the memory map. See Swap Control Command for details.

25.4.2 Flash Protection

Individual regions within the flash memory can be protected from program and erase operations. Protection is controlled by the following registers:

- FPROT*n* — Four registers that protect 32 regions of the program flash memory as shown in the following figure

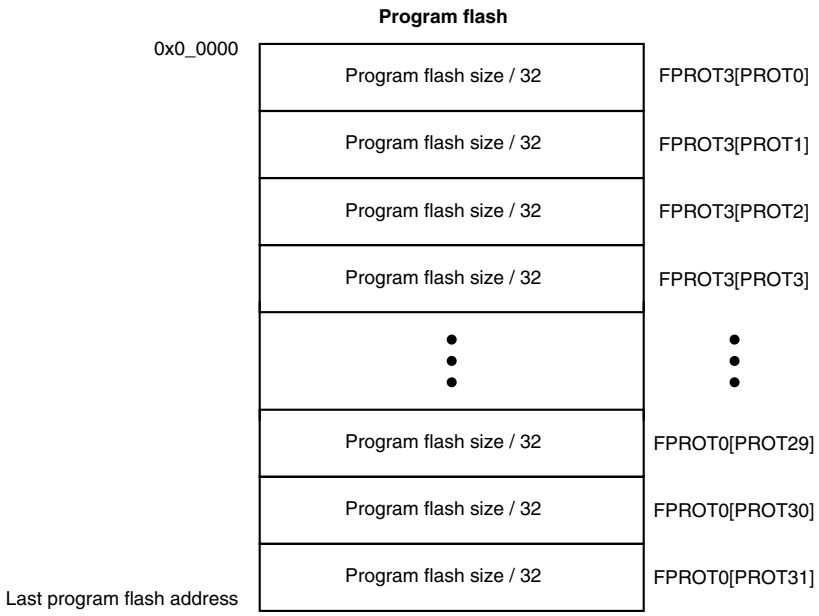


Figure 25-26. Program flash protection

- FDPROT —
  - protects eight regions of the data flash memory as shown in the following figure

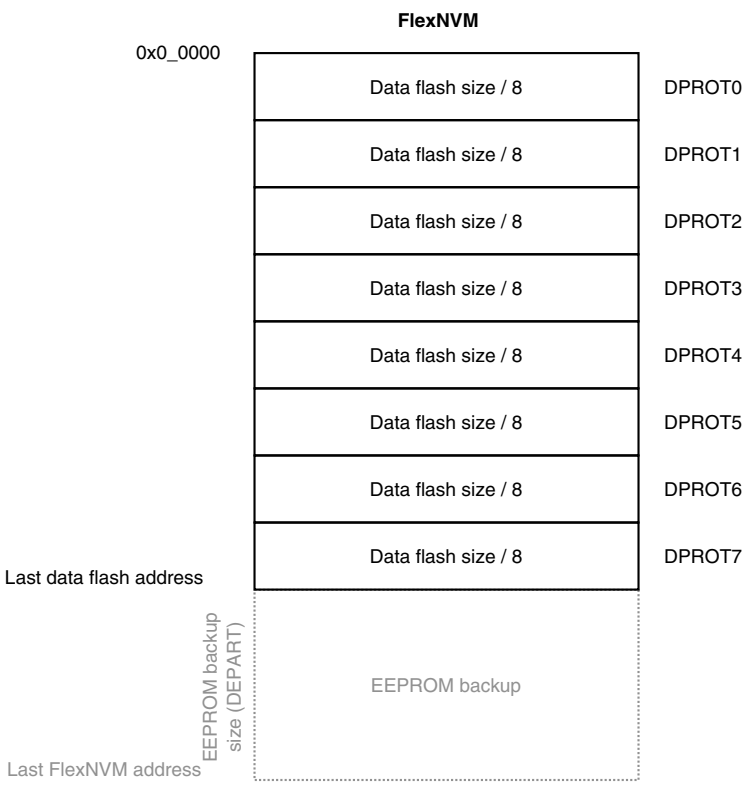


Figure 25-27. Data flash protection

- FEPROT — Protects eight regions of the EEPROM memory as shown in the following figure

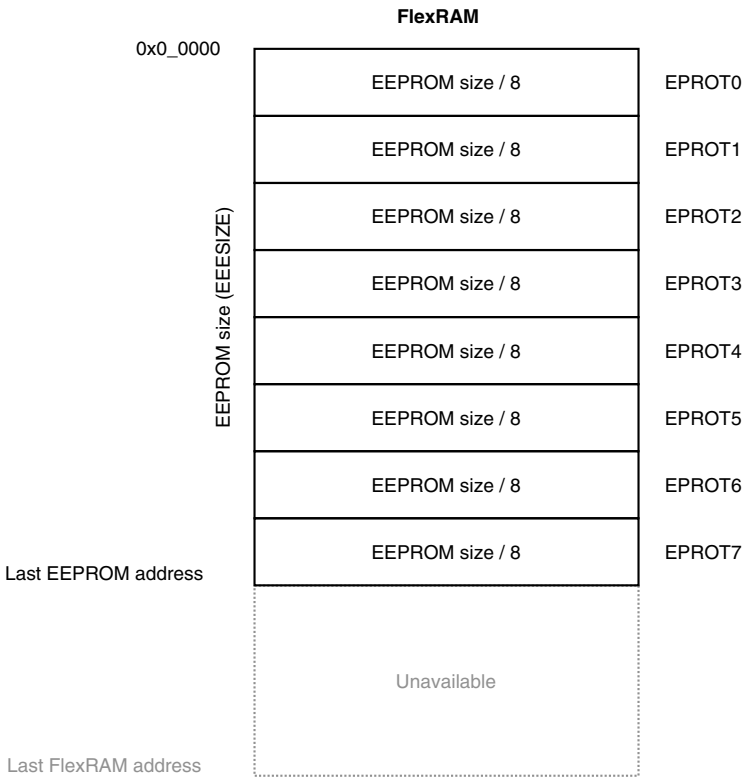


Figure 25-28. EEPROM protection

25.4.3 FlexNVM Description

This section describes the FlexNVM memory.

25.4.3.1 FlexNVM Block Partitioning for FlexRAM

The user can configure the FlexNVM block as either:

- Basic data flash,
- EEPROM flash records to support the built-in EEPROM feature, or
- A combination of both.

The user's FlexNVM configuration choice is specified using the Program Partition command described in Program Partition Command.

CAUTION

While different partitions of the FlexNVM block are available, the intention is that a single partition choice is used throughout the entire lifetime of a given application. The FlexNVM partition code choices affect the endurance and data retention characteristics of the device.

25.4.3.2 EEPROM User Perspective

The EEPROM system is shown in the following figure.

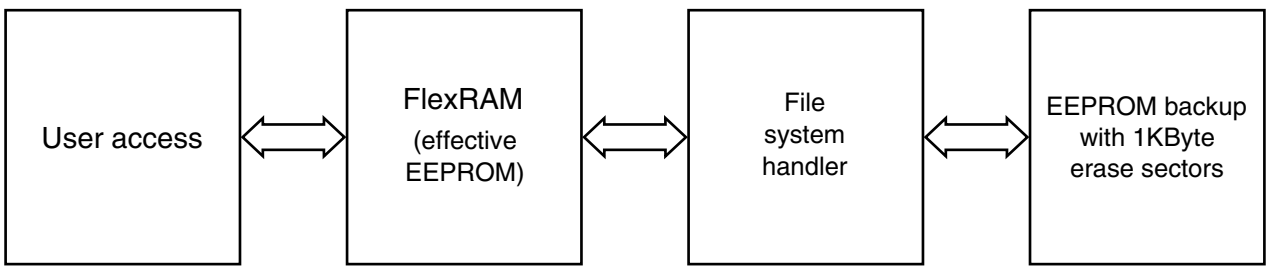


Figure 25-29. Top Level EEPROM Architecture

To handle varying customer requirements, the FlexRAM and FlexNVM blocks can be split into partitions as shown in the figure below.

1. **EEPROM partition** (EEESIZE) — The amount of FlexRAM used for EEPROM can be set from 0 Bytes (no EEPROM) to the maximum FlexRAM size (see Table 25-2). The remainder of the FlexRAM is not accessible while the FlexRAM is configured for EEPROM (see Set FlexRAM Function Command). The EEPROM partition grows upward from the bottom of the FlexRAM address space.
2. **Data flash partition** (DEPART) — The amount of FlexNVM memory used for data flash can be programmed from 0 bytes (all of the FlexNVM block is available for EEPROM backup) to the maximum size of the FlexNVM block (see Table 25-4).
3. **FlexNVM EEPROM partition** — The amount of FlexNVM memory used for EEPROM backup, which is equal to the FlexNVM block size minus the data flash memory partition size. The EEPROM backup size must be at least 16 times the EEPROM partition size in FlexRAM.
4. **EEPROM split factor** (EEESPLIT) — The FlexRAM partitioned for EEPROM can be divided into two subsystems, each backed by half of the partitioned EEPROM backup. One subsystem (A) is 1/8, 1/4, or 1/2 of the partitioned FlexRAM with the remainder belonging to the other subsystem (B).

The partition information (EEESIZE, DEPART, EEESPLIT) is stored in the data flash IFR and is programmed using the Program Partition command (see Program Partition Command). Typically, the Program Partition command is executed only once in the lifetime of the device.

Data flash memory is useful for applications that need to quickly store large amounts of data or store data that is static. The EEPROM partition in FlexRAM is useful for storing smaller amounts of data that will be changed often. The EEPROM partition in FlexRAM can be further sub-divided to provide subsystems, each backed by the same amount of EEPROM backup with subsystem A having higher endurance if the split factor is 1/8 or 1/4.

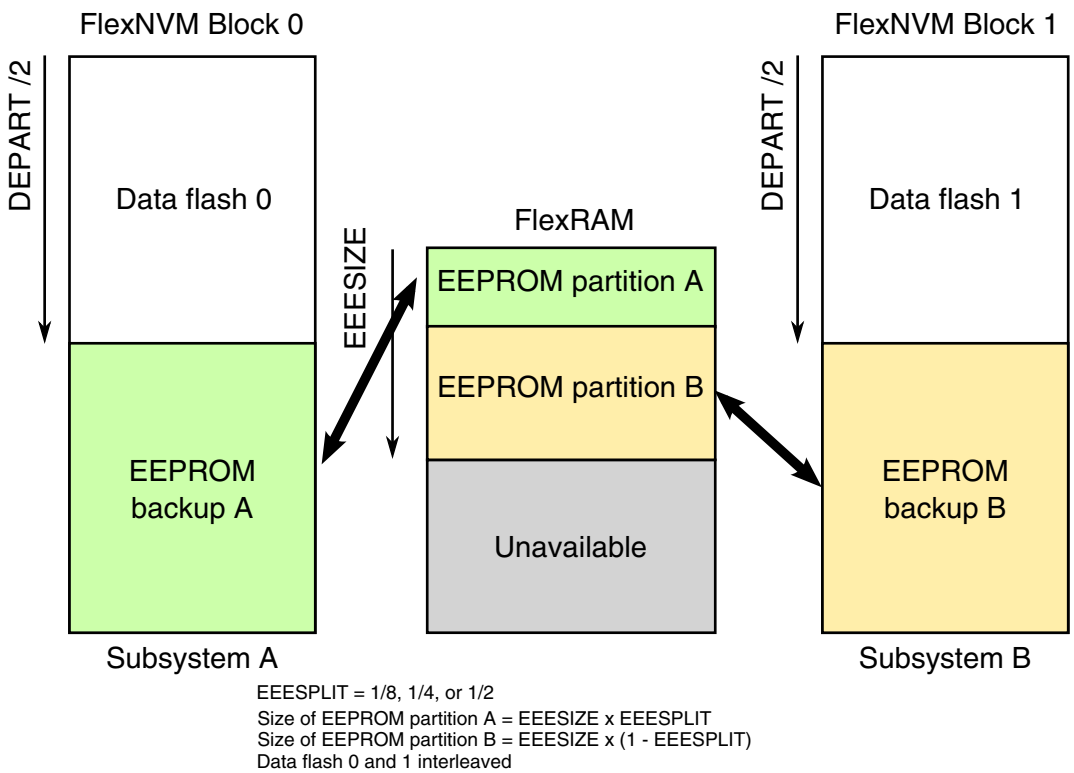


Figure 25-30. FlexRAM to FlexNVM Memory Mapping with 2 Sub-systems

25.4.3.3 EEPROM Implementation Overview

Out of reset with the FSTAT[CCIF] bit clear, the partition settings (EEESIZE, DEPART, EEESPLIT) are read from the data flash IFR and the EEPROM file system is initialized accordingly. The EEPROM file system locates all valid EEPROM data records in EEPROM backup and copies the newest data to FlexRAM. The FSTAT[CCIF] and FCNFG[EEERDY] bits are set after data from all valid EEPROM data records is copied to the FlexRAM. After the CCIF bit is set, the FlexRAM is available for read or write access.

When configured for EEPROM use, writes to an unprotected location in FlexRAM invokes the EEPROM file system to program a new EEPROM data record in the EEPROM backup memory in a round-robin fashion. As needed, the EEPROM file system identifies the EEPROM backup sector that is being erased for future use and partially erases that EEPROM backup sector. After a write to the FlexRAM, the FlexRAM is not accessible until the FSTAT[CCIF] bit is set. The FCNFG[EEERDY] bit will also be set. If enabled, the interrupt associated with the FSTAT[CCIF] bit can be used to determine when the FlexRAM is available for read or write access.

After a sector in EEPROM backup is full of EEPROM data records, EEPROM data records from the sector holding the oldest data are gradually copied over to a previously-erased EEPROM backup sector. When the sector copy completes, the EEPROM backup sector holding the oldest data is tagged for erase.

25.4.3.4 Write endurance to FlexRAM for EEPROM

When the FlexNVM partition code is not set to full data flash, the EEPROM data set size can be set to any of several non-zero values.

The bytes not assigned to data flash via the FlexNVM partition code are used by the flash memory module to obtain an effective endurance increase for the EEPROM data. The built-in EEPROM record management system raises the number of program/erase cycles that can be attained prior to device wear-out by cycling the EEPROM data through a larger EEPROM NVM storage space.

While different partitions of the FlexNVM are available, the intention is that a single choice for the FlexNVM partition code and EEPROM data set size is used throughout the entire lifetime of a given application. The EEPROM endurance equation and graph shown below assume that only one configuration is ever used.

$$\text{Writes\_subsystem} = \frac{\text{EEPROM} - 2 \times \text{EEESPLIT} \times \text{EEESIZE}}{\text{EEESPLIT} \times \text{EEESIZE}} \times \text{Write\_efficiency} \times n_{\text{nvmcycd}}$$

where

- Writes_subsystem — minimum number of writes to each FlexRAM location for subsystem (each subsystem can have different endurance)
- EEPROM — allocated FlexNVM for each EEPROM subsystem based on DEPART; entered with the Program Partition command
- EEESPLIT — FlexRAM split factor for subsystem; entered with the Program Partition command
- EEESIZE — allocated FlexRAM based on DEPART; entered with the Program Partition command

- Write_efficiency —
  - 0.25 for 8-bit writes to FlexRAM
  - 0.50 for 16-bit or 32-bit writes to FlexRAM
- $n_{nvmcycd}$ — data flash cycling endurance (the following graph assumes 10,000 cycles)

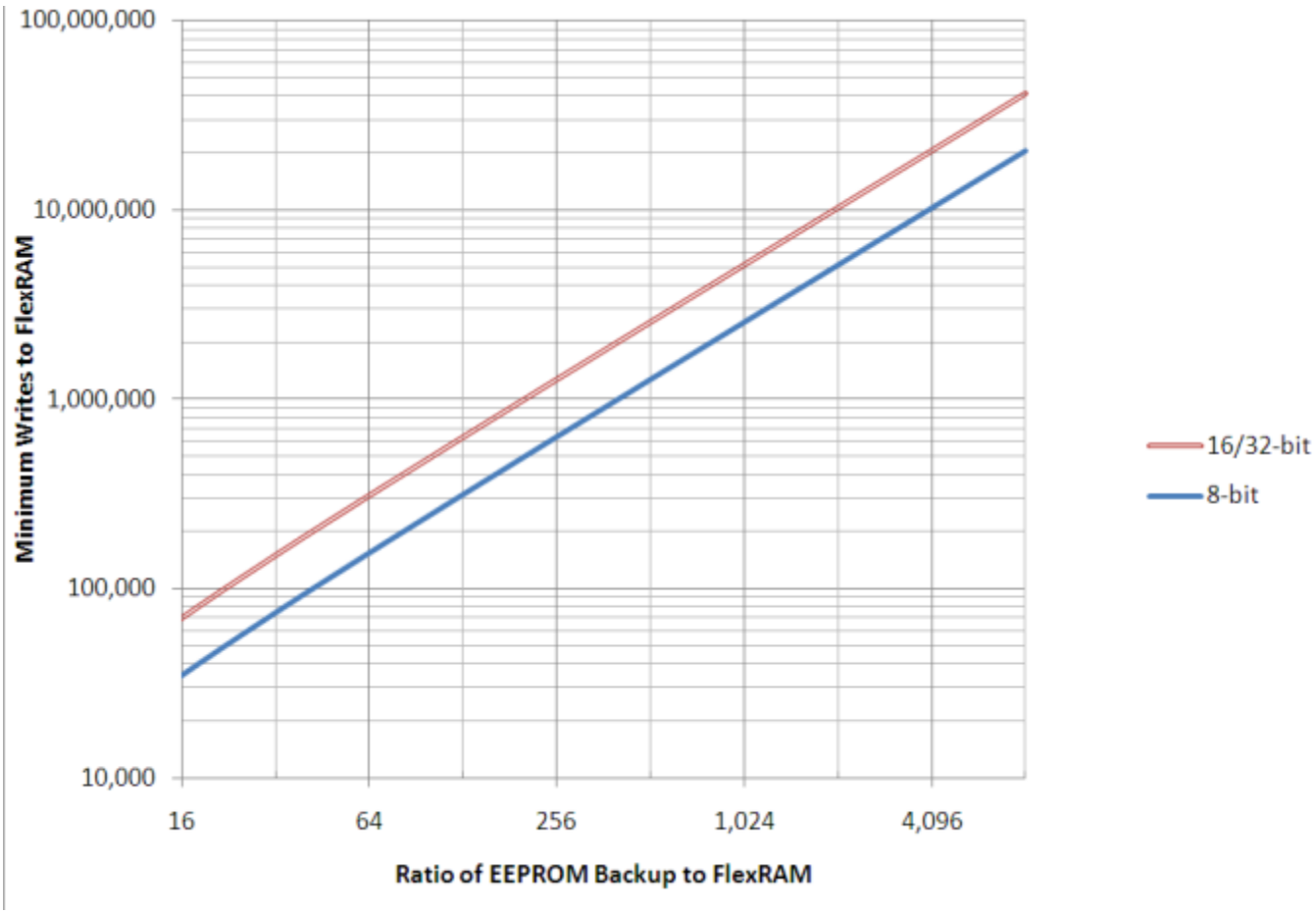


Figure 25-31. EEPROM backup writes to FlexRAM

25.4.4 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 25-30. 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.

25.4.5 Flash Operation in Low-Power Modes

25.4.5.1 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).

25.4.5.2 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.

25.4.6 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 25-31). Refer to the Chip Configuration details of this device for how to activate each mode.

25.4.7 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).

25.4.8 Read While Write (RWW)

The following simultaneous accesses are allowed:

- The user may read from the program flash memory while commands (typically program and erase operations) are active in the data flash and FlexRAM memory space.
- The MCU can fetch instructions from program flash during both data flash program and erase operations and while EEPROM backup data is maintained by the EEPROM commands.
- Conversely, the user may read from data flash and FlexRAM while program and erase commands are executing on the program flash.
- When configured as traditional RAM, writes to the FlexRAM are allowed during program and data flash operations.

Simultaneous data flash operations and FlexRAM writes, when FlexRAM is used for EEPROM, are not possible.

Simultaneous operations are further discussed in Allowed Simultaneous Flash Operations.

25.4.9 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.

25.4.10 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

25.4.10.1 Command Write Sequence

Flash commands are specified using a command write sequence illustrated in Figure 25-32. 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.

25.4.10.1.1 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.

25.4.10.1.2 Launch the Command by Clearing CCIF

Once all relevant command parameters have been loaded, the user launches the command by clearing the FSTAT[CCIF] bit by writing a '1' to it. The CCIF flag remains zero until the flash command completes.

The FSTAT register contains a blocking mechanism that prevents a new command from launching (can't clear CCIF) if the previous command resulted in an access error (FSTAT[ACCERR]=1) or a protection violation (FSTAT[FPVIOL]=1). In error scenarios, two writes to FSTAT are required to initiate the next command: the first write clears the error flags, the second write clears CCIF.

25.4.10.1.3 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. ACCERR reports invalid instruction codes and out-of bounds addresses. Usually, access errors suggest that the command was not set-up with valid parameters in the FCCOB register group.

   Program and erase commands also check the address to determine if the operation is requested to execute on protected areas. If the protection check fails, the FSTAT[FPVIOL] (protection error) flag is set.

   Command processing never proceeds to execution when the parameter or protection step fails. Instead, command processing is terminated after setting the FSTAT[CCIF] bit.

2. If the parameter and protection checks pass, the command proceeds to execution. Run-time errors, such as failure to erase verify, may occur during the execution phase. Run-time errors are reported in the FSTAT[MGSTAT0] bit. A command may have access errors, protection errors, and run-time errors, but the run-time errors are not seen until all access and protection errors have been corrected.
3. Command execution results, if applicable, are reported back to the user via the FCCOB and FSTAT registers.
4. The flash memory module sets the FSTAT[CCIF] bit signifying that the command has completed.

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

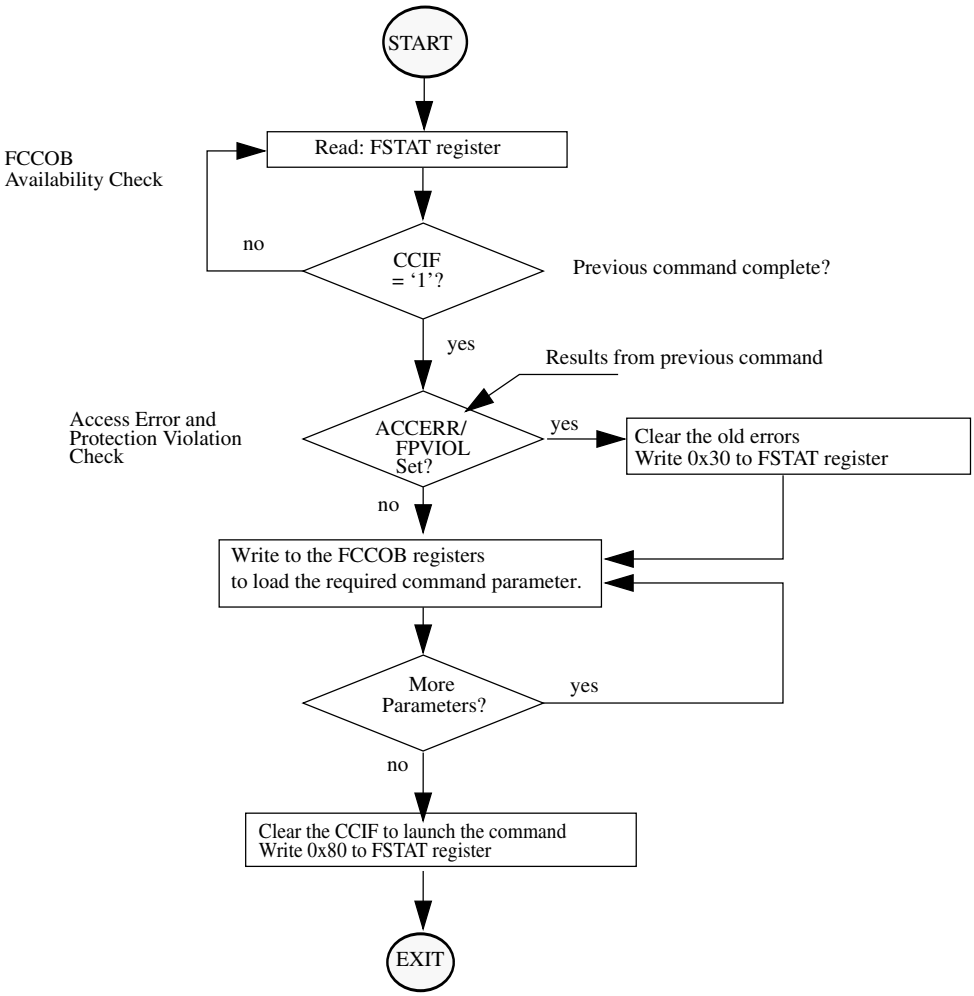


Figure 25-32. Generic Flash Command Write Sequence Flowchart

25.4.10.2 Flash Commands

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

FCMD	Command	Program flash	Data flash	FlexRAM	Function
0x00	Read 1s Block	×	×		Verify that a program flash or data flash block is erased. FlexNVM block must not be partitioned for EEPROM.

Table continues on the next page...

FCMD	Command	Program flash	Data flash	FlexRAM	Function
0x01	Read 1s Section	×	×		Verify that a given number of program flash or data flash locations from a starting address are erased.
0x02	Program Check	×	×		Tests previously-programmed locations at margin read levels.
0x03	Read Resource	IFR, ID	IFR		Read 4 bytes from program flash IFR, data flash IFR, or version ID.
0x06	Program Longword	×	×		Program 4 bytes in a program flash block or a data flash block.
0x08	Erase Flash Block	×	×		Erase a program flash block or data flash block. An erase of any flash block is only possible when unprotected. FlexNVM block must not be partitioned for EEPROM.
0x09	Erase Flash Sector	×	×		Erase all bytes in a program flash or data flash sector.
0x0B	Program Section	×	×	×	Program data from the Section Program Buffer to a program flash or data flash block.
0x40	Read 1s All Blocks	×	×		Verify that all program flash, data flash blocks, EEPROM backup data records, and data flash IFR are erased then release MCU security.
0x41	Read Once	IFR			Read 4 bytes of a dedicated 64 byte field in the program flash IFR.

Table continues on the next page...

FCMD	Command	Program flash	Data flash	FlexRAM	Function
0x43	Program Once	IFR			One-time program of 4 bytes of a dedicated 64-byte field in the program flash IFR.
0x44	Erase All Blocks	×	×	×	Erase all program flash blocks, program flash swap IFR, data flash blocks, FlexRAM, EEPROM backup data records, and data flash IFR. Then, verify-erase and release MCU security. **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.
0x46	Swap Control	×			Handles swap-related activities
0x80	Program Partition		IFR	×	Program the FlexNVM Partition Code and EEPROM Data Set Size into the data flash IFR. Format all EEPROM backup data sectors allocated for EEPROM. Initialize the FlexRAM.

Table continues on the next page...

FCMD	Command	Program flash	Data flash	FlexRAM	Function
0x81	Set FlexRAM Function		x	×	Switches FlexRAM function between RAM and EEPROM. When switching to EEPROM, FlexNVM is not available while valid data records are being copied from EEPROM backup to FlexRAM.

NOTE

FlexRAM, or Programming Acceleration RAM, is used during PGMSEC command.

25.4.10.3 Flash Commands by Mode

The following table shows the flash commands that can be executed in each flash operating mode.

Table 25-31. Flash Commands by Mode

FCMD	Command	NVM Normal			NVM Special		
		Unsecure	Secure	MEEN=10	Unsecure	Secure	MEEN=10
0x00	Read 1s Block	×	×	×	×	—	—
0x01	Read 1s Section	×	×	×	×	—	—
0x02	Program Check	×	×	×	×	—	—
0x03	Read Resource	×	×	×	×	—	—
0x06	Program Longword	×	×	×	×	—	—
0x08	Erase Flash Block	×	×	×	×	—	—
0x09	Erase Flash Sector	×	×	×	×	—	—
0x0B	Program Section	×	×	×	×	—	—
0x40	Read 1s All Blocks	×	×	×	×	×	—
0x41	Read Once	×	×	×	×	—	—
0x43	Program Once	×	×	×	×	—	—
0x44	Erase All Blocks	×	×	×	×	×	—
0x45	Verify Backdoor Access Key	×	×	×	×	—	—
0x46	Swap Control	×	×	×	×	—	—
0x80	Program Partition	×	×	×	×	—	—
0x81	Set FlexRAM Function	×	×	×	×	—	—

25.4.10.4 Allowed Simultaneous Flash Operations

Only the operations marked 'OK' in the following table are permitted to run simultaneously on the program flash, data flash, and FlexRAM memories. Some operations cannot be executed simultaneously because certain hardware resources are shared by the memories. The priority has been placed on permitting program flash reads while program and erase operations execute on the FlexNVM and FlexRAM. This provides read (program flash) while write (FlexNVM, FlexRAM) functionality.

Table 25-32. Allowed Simultaneous Memory Operations

		Program Flash			Data Flash			FlexRAM		
		Read	Program	Sector Erase	Read	Program	Sector Erase	Read	E-Write[1]	R-Write[2]
Program flash	Read	—				OK	OK		OK	
	Program		—		OK			OK		OK[3]
	Sector Erase			—	OK			OK		OK
Data flash	Read		OK	OK	—					
	Program	OK				—		OK		OK
	Sector Erase	OK					—	OK		OK
FlexRAM	Read		OK	OK		OK	OK	—		
	E-Write[1]	OK							—	
	R-Write[2]		OK	OK		OK	OK			—

1. When FlexRAM configured for EEPROM (writes are effectively multi-cycle operations).
2. When FlexRAM configured as traditional RAM (writes are single-cycle operations).
3. When FlexRAM configured as traditional RAM, writes to the RAM are ignored while the Program Section command is active (CCIF = 0).

25.4.11 Margin Read Commands

The Read-1s commands (Read 1s All Blocks, Read 1s Block, and Read 1s Section) and the Program Check command have a margin choice parameter that allows the user to apply non-standard read reference levels to the program flash and data flash array reads performed by these commands. Using the preset 'user' and 'factory' margin levels, these commands perform their associated read operations at tighter tolerances than a 'normal' read. These non-standard read levels are applied only during the command execution. All simple (uncommanded) flash array reads to the MCU always use the standard, un-margined, read reference level.

Only the 'normal' read level should be employed during normal flash usage. The non-standard, 'user' and 'factory' margin levels should be employed only in special cases. They can be used during special diagnostic routines to gain confidence that the device is not suffering from the end-of-life data loss customary of flash memory devices.

Erased ('1') and programmed ('0') bit states can degrade due to elapsed time and data cycling (number of times a bit is erased and re-programmed). The lifetime of the erased states is relative to the last erase operation. The lifetime of the programmed states is measured from the last program time.

The 'user' and 'factory' levels become, in effect, a minimum safety margin; i.e. if the reads pass at the tighter tolerances of the 'user' and 'factory' margins, then the 'normal' reads have at least this much safety margin before they experience data loss.

The 'user' margin is a small delta to the normal read reference level. 'User' margin levels can be employed to check that flash memory contents have adequate margin for normal level read operations. If unexpected read results are encountered when checking flash memory contents at the 'user' margin levels, loss of information might soon occur during 'normal' readout.

The 'factory' margin is a bigger deviation from the norm, a more stringent read criteria that should only be attempted immediately (or very soon) after completion of an erase or program command, early in the cycling life. 'Factory' margin levels can be used to check that flash memory contents have adequate margin for long-term data retention at the normal level setting. If unexpected results are encountered when checking flash memory contents at 'factory' margin levels, the flash memory contents should be erased and reprogrammed.

CAUTION

Factory margin levels must only be used during verify of the initial factory programming.

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

Do not attempt to read a flash block while the flash memory module is running a command (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.

When required by the command, address bit 23 selects between:

- program flash (=0)
- data flash (=1)

CAUTION

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

25.4.12.1 Read 1s Block Command

The Read 1s Block command checks to see if an entire program flash or data flash block has been erased to the specified margin level. The FCCOB flash address bits determine which logical block is erase-verified.

Table 25-33. Read 1s Block Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x00 (RD1BLK)
1	Flash address [23:16] in the flash block to be verified
2	Flash address [15:8] in the flash block to be verified
3	Flash address [7:0][1] in the flash block to be verified
4	Read-1 Margin Choice

1. Must be longword aligned (Flash address [1:0] = 00).

After clearing CCIF to launch the Read 1s Block command, the flash memory module sets the read margin for 1s according to Table 25-34 and then reads all locations within the selected program flash or data flash block.

When the data flash is targeted, DEPART must be set for no EEPROM, else the Read 1s Block command aborts setting the FSTAT[ACCERR] bit. If the flash memory module fails to read all 1s (i.e. the flash block is not fully erased), the FSTAT[MGSTAT0] bit is set. The CCIF flag sets after the Read 1s Block operation has completed.

Table 25-34. Margin Level Choices for Read 1s Block

Read Margin Choice	Margin Level Description
0x00	Use the 'normal' read level for 1s
0x01	Apply the 'User' margin to the normal read-1 level
0x02	Apply the 'Factory' margin to the normal read-1 level

Table 25-35. Read 1s Block Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid margin choice is specified	FSTAT[ACCERR]
Program flash is selected and the address is out of program flash range	FSTAT[ACCERR]
Data flash is selected and the address is out of data flash range	FSTAT[ACCERR]
Data flash is selected with EEPROM enabled	FSTAT[ACCERR]
Flash address is not longword aligned	FSTAT[ACCERR]
Read-1s fails	FSTAT[MGSTAT0]

25.4.12.2 Read 1s Section Command

The Read 1s Section command checks if a section of program flash or data flash memory is erased to the specified read margin level. The Read 1s Section command defines the starting address and the number of longwords to be verified.

Table 25-36. Read 1s Section Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x01 (RD1SEC)
1	Flash address [23:16] of the first longword to be verified
2	Flash address [15:8] of the first longword to be verified
3	Flash address [7:0][1] of the first longword to be verified
4	Number of longwords to be verified [15:8]
5	Number of longwords to be verified [7:0]
6	Read-1 Margin Choice

1. Must be longword aligned (Flash address [1:0] = 00).

Upon clearing CCIF to launch the Read 1s Section command, the flash memory module sets the read margin for 1s according to Table 25-37 and then reads all locations within the specified section of flash memory. If the flash memory module fails to read all 1s (i.e. the flash section is not erased), the FSTAT[MGSTAT0] bit is set. The CCIF flag sets after the Read 1s Section operation completes.

Table 25-37. 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 25-38. Read 1s Section Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid margin code is supplied	FSTAT[ACCERR]
An invalid flash address is supplied	FSTAT[ACCERR]
Flash address is not longword aligned	FSTAT[ACCERR]
The requested section crosses a Flash block boundary	FSTAT[ACCERR]
The requested number of longwords is zero	FSTAT[ACCERR]
Read-1s fails	FSTAT[MGSTAT0]

25.4.12.3 Program Check Command

The Program Check command tests a previously programmed program flash or data flash longword to see if it reads correctly at the specified margin level.

Table 25-39. 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 25-40, reads the specified longword, and compares the actual read data to the expected data provided by the FCCOB. If the comparison at margin-1 fails, the FSTAT[MGSTAT0] bit 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, the FSTAT[MGSTAT0] bit is set. The CCIF flag 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 0 data is written to the supplied byte address ('start'),
- Byte 1 data is programmed to byte address start+0b01,
- Byte 2 data is programmed to byte address start+0b10,
- Byte 3 data is programmed to byte address start+0b11.

NOTE

See the description of margin reads, Margin Read Commands

Table 25-40. 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 25-41. Program Check Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid flash address is supplied	FSTAT[ACCERR]
Flash address is not longword aligned	FSTAT[ACCERR]
An invalid margin choice is supplied	FSTAT[ACCERR]
Either of the margin reads does not match the expected data	FSTAT[MGSTAT0]

25.4.12.4 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, data flash IFR space, and the Version ID field. Each resource is assigned a select code as shown in Table 25-43.

Table 25-42. Read Resource Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x03 (RDRSRC)
1	Flash address [23:16]
2	Flash address [15:8]

Table continues on the next page...

Table 25-42. Read Resource Command FCCOB Requirements (continued)

FCCOB Number	FCCOB Contents [7:0]
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 25-43)

1. Must be longword aligned (Flash address [1:0] = 00).

Table 25-43. Read Resource Select Codes

Resource Select Code	Description	Resource Size	Local Address Range
0x00	Program Flash 0 IFR	256 Bytes	0x00_0000 - 0x00_00FF
0x00	Program Flash Swap IFR	256 Bytes	0x02_0000 - 0x02_00FF (512 KB of program flash) 0x01_0000 - 0x01_00FF (256 KB of program flash) 0x00_8000 - 0x00_80FF (128 KB of program flash)
0x00	Data Flash 0 IFR	256 Bytes	0x80_0000 - 0x80_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 25-44. 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]

25.4.12.5 Program Longword Command

The Program Longword command programs four previously-erased bytes in the program flash memory or in the data flash memory using an embedded algorithm.

CAUTION

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

Table 25-45. Program Longword Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x06 (PGM4)
1	Flash address [23:16]
2	Flash address [15:8]
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 swap indicator address in each program flash block is implicitly protected from programming. The targeted flash locations must be currently unprotected (see the description of the FPROT and FDPROT 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 0 data is written to the supplied byte address ('start'),
- Byte 1 data is programmed to byte address start+0b01,
- Byte 2 data is programmed to byte address start+0b10, and
- Byte 3 data is programmed to byte address start+0b11.

Table 25-46. 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]
Any errors have been encountered during the verify operation	FSTAT[MGSTAT0]

25.4.12.6 Erase Flash Block Command

The Erase Flash Block operation erases all addresses in a single program flash or data flash block.

Table 25-47. Erase Flash Block Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x08 (ERSBLK)
1	Flash address [23:16] in the flash block to be erased
2	Flash address [15:8] in the flash block to be erased
3	Flash address [7:0][1] in the flash block to be erased

1. Must be longword aligned (Flash address [1:0] = 00).

Upon clearing CCIF to launch the Erase Flash Block command, the flash memory module erases the main array of the selected flash block and verifies that it is erased. When the data flash is targeted, DEPART must be set for no EEPROM (see Table 25-4) else the Erase Flash Block command aborts setting the FSTAT[ACCERR] bit. The Erase Flash Block command aborts and sets the FSTAT[FPVIOL] bit if any region within the block is protected (see the description of the FPROT and FDPROT registers). The swap indicator address in each program flash block is implicitly protected from block erase unless the swap system is in the UPDATE or UPDATE-ERASED state and the program flash block being erased is the non-active block. If the erase verify fails, FSTAT[MGSTAT0] is set. The CCIF flag will set after the Erase Flash Block operation has completed.

Table 25-48. Erase Flash Block Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
Program flash is selected and the address is out of program flash range	FSTAT[ACCERR]
Data flash is selected and the address is out of data flash range	FSTAT[ACCERR]
Data flash is selected with EEPROM enabled	FSTAT[ACCERR]

Table continues on the next page...

Table 25-48. Erase Flash Block Command Error Handling (continued)

Error Condition	Error Bit
Flash address is not longword aligned	FSTAT[ACCERR]
Any area of the selected flash block is protected	FSTAT[FPVIOL]
Any errors have been encountered during the verify operation[1]	FSTAT[MGSTAT0]

1. User margin read may be run using the Read 1s Block command to verify all bits are erased.

25.4.12.7 Erase Flash Sector Command

The Erase Flash Sector operation erases all addresses in a flash sector.

Table 25-49. 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 longword aligned (flash address [1:0] = 00).

After clearing CCIF to launch the Erase Flash Sector command, the flash memory module erases the selected program flash or data flash sector and then verifies that it is erased. The Erase Flash Sector command aborts if the selected sector is protected (see the description of the FPROT and FDPROT registers). The swap indicator address in each program flash block is implicitly protected from sector erase unless the swap system is in the UPDATE or UPDATE-ERASED state and the program flash sector containing the swap indicator address being erased is the non-active block. If the erase-verify fails the FSTAT[MGSTAT0] bit is set. The CCIF flag is set after the Erase Flash Sector operation completes. The Erase Flash Sector command is suspendable (see the FCNFG[ERSSUSP] bit and Figure 25-33).

Table 25-50. 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 longword aligned	FSTAT[ACCERR]
The selected program flash or data flash sector is protected	FSTAT[FPVIOL]
Any errors have been encountered during the verify operation[1]	FSTAT[MGSTAT0]

1. User margin read may be run using the Read 1s Section command to verify all bits are erased.

25.4.12.7.1 Suspending an Erase Flash Sector Operation

To suspend an Erase Flash Sector operation set the FCNFG[ERSSUSP] bit (see Flash Configuration Field Description) when CCIF is clear and the CCOB command field holds the code for the Erase Flash Sector command. During the Erase Flash Sector operation (see Erase Flash Sector Command), the flash 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.

25.4.12.7.2 Resuming a Suspended Erase Flash Sector Operation

If the ERSSUSP bit is still set when CCIF is cleared to launch the next command, the previous Erase Flash Sector operation resumes. The 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 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.

25.4.12.7.3 Aborting a Suspended Erase Flash Sector Operation

The user may choose to abort a suspended Erase Flash Sector operation by clearing the ERSSUSP bit prior to clearing CCIF for the next command launch. When a suspended operation is aborted, the flash memory module starts the new command using the new FCCOB contents.

While FCNFG[ERSSUSP] is set, a write to the FlexRAM while FCNFG[EEERDY] is set clears ERSSUSP and aborts the suspended operation. The FlexRAM write operation is executed by the flash memory module.

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.

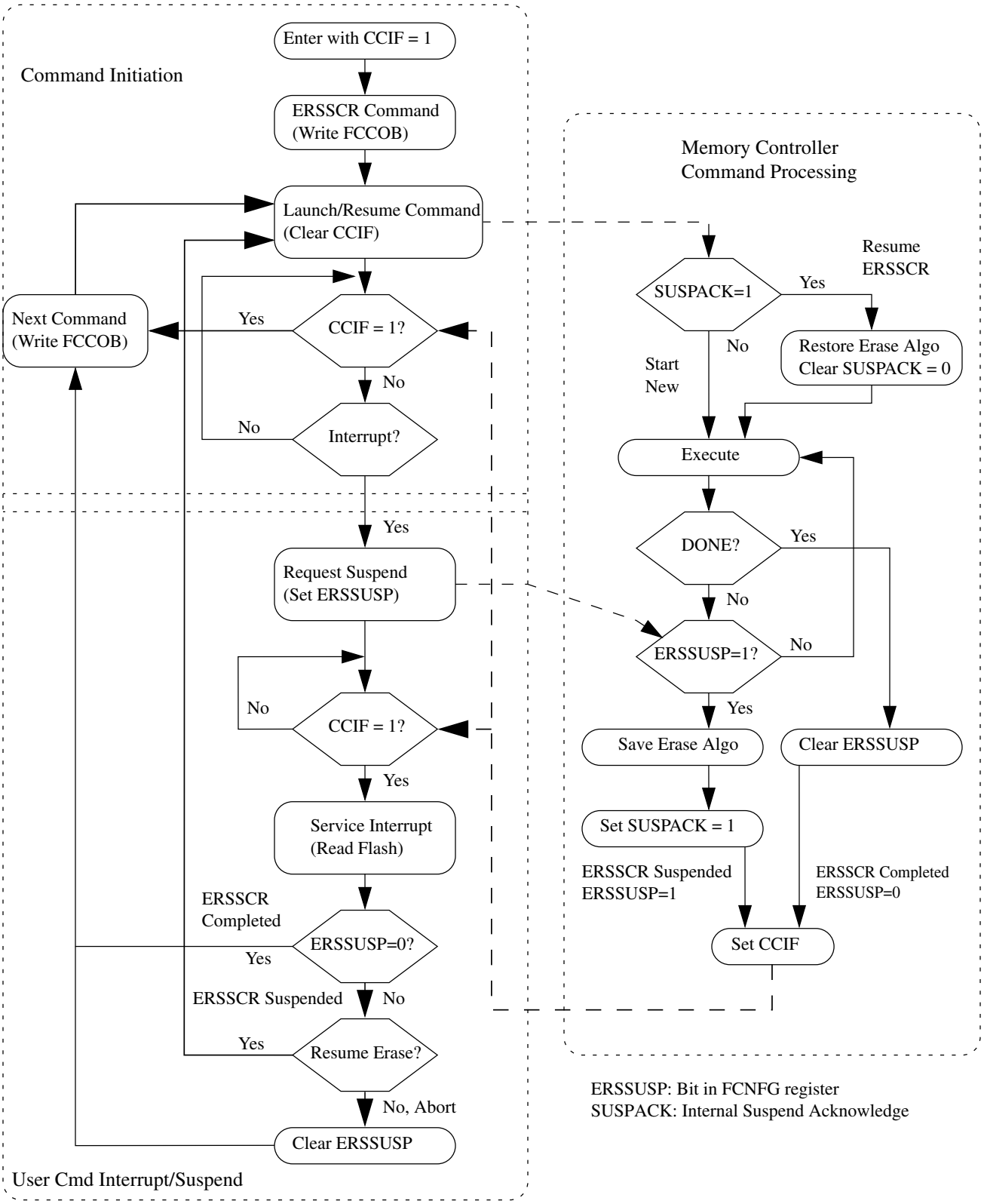


Figure 25-33. Suspend and Resume of Erase Flash Sector Operation

25.4.12.8 Program Section Command

The Program Section operation programs the data found in the section program buffer to previously erased locations in the flash memory using an embedded algorithm. Data is preloaded into the section program buffer by writing to the FlexRAM while it is set to function as traditional RAM or the programming acceleration RAM (see Flash Sector Programming).

The section program buffer is limited to the lower half of the RAM. Data written to the upper half of the RAM is ignored and may be overwritten during Program Section command execution.

CAUTION

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

Table 25-51. Program Section Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x0B (PGMSEC)
1	Flash address [23:16]
2	Flash address [15:8]
3	Flash address [7:0][1]
4	Number of longwords to program [15:8]
5	Number of longwords to program [7:0]

1. Must be longword aligned (Flash address [1:0] = 00).

After clearing CCIF to launch the Program Section command, the flash memory module blocks access to the FlexRAM and programs the data residing in the section program buffer into the flash memory starting at the flash address provided.

The starting address must be unprotected (see the description of the FPROT and FDPROT registers) to permit execution of the Program Section operation. The swap indicator address in each program flash block is implicitly protected from programming. If the swap indicator address is encountered during the Program Section operation, it is bypassed without setting FPVIOL and the contents are not programmed. Programming, which is not allowed to cross a flash sector boundary, continues until all requested longwords have been programmed. The Program Section command also verifies that after programming, all bits requested to be programmed are programmed.

After the Program Section operation completes, the CCIF flag is set and normal access to the RAM is restored. The contents of the section program buffer may be changed by the Program Section operation.

Table 25-52. Program Section Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid flash address is supplied	FSTAT[ACCERR]
Flash address is not longword aligned	FSTAT[ACCERR]
The requested section crosses a program flash sector boundary	FSTAT[ACCERR]
The requested number of longwords is zero	FSTAT[ACCERR]
The space required to store data for the requested number of longwords is more than half the size of the FlexRAM	FSTAT[ACCERR]
The FlexRAM is not set to function as a traditional RAM, i.e. set if RAMRDY=0	FSTAT[ACCERR]
The flash address falls in a protected area	FSTAT[FPVIOL]
Any errors have been encountered during the verify operation	FSTAT[MGSTAT0]

25.4.12.8.1 Flash Sector Programming

The process of programming an entire flash sector using the Program Section command is as follows:

1. If required, for FlexNVM devices, execute the Set FlexRAM Function command to make the FlexRAM available as traditional RAM and initialize the FlexRAM to all ones.
2. Launch the Erase Flash Sector command to erase the flash sector to be programmed.
3. Beginning with the starting address of the FlexRAM, sequentially write enough data to the RAM to fill an entire flash sector or half the FlexRAM, whichever is less. This area of the RAM serves as the section program buffer.

NOTE

In step 1, the section program buffer was initialized to all ones, the erased state of the flash memory.

The section program buffer can be written to while the operation launched in step 2 is executing, i.e. while CCIF = 0.

4. Execute the Program Section command to program the contents of the section program buffer into the selected flash sector.
5. If a flash sector is larger than half the RAM, repeat steps 3 and 4 until the sector is completely programmed.
6. To program additional flash sectors, repeat steps 2 through 4.
7. To restore EEPROM functionality for FlexNVM devices, execute the Set FlexRAM Function command to make the FlexRAM available as EEPROM.

25.4.12.9 Read 1s All Blocks Command

The Read 1s All Blocks command checks if the program flash blocks, data flash blocks, EEPROM backup records, and data flash IFR have been erased to the specified read margin level, if applicable, and releases security if the readout passes, i.e. all data reads as '1'.

Table 25-53. 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 25-54,
- checks the contents of the program flash, data flash, EEPROM backup records, and data flash IFR are in the erased state.

If the flash memory module confirms that these memory resources are erased, security is released by setting the FSEC[SEC] field to the unsecure state. The security byte in the flash configuration field (see Flash Configuration Field Description) remains unaffected by the Read 1s All Blocks command. If the read fails, i.e. all memory resources are not in the fully erased state, the FSTAT[MGSTAT0] bit is set.

The EEERDY and RAMRDY bits are clear during the Read 1s All Blocks operation and are restored at the end of the Read 1s All Blocks operation.

The CCIF flag sets after the Read 1s All Blocks operation has completed.

Table 25-54. 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 25-55. Read 1s All Blocks Command Error Handling

Error Condition	Error Bit
An invalid margin choice is specified	FSTAT[ACCERR]
Read-1s fails	FSTAT[MGSTAT0]

25.4.12.10 Read Once Command

The Read Once command provides read access to a reserved 64-byte field located in the program flash IFR (see Program Flash IFR Map and Program Once Field). Access to this field is via 16 records, each 4 bytes long. The Read Once field is programmed using the Program Once command described in Program Once Command.

Table 25-56. Read Once Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x41 (RDONCE)
1	Read Once record index (0x00 - 0x0F)
2	Not used
3	Not used
Returned Values	
4	Read Once byte 0 value
5	Read Once byte 1 value
6	Read Once byte 2 value
7	Read Once byte 3 value

After clearing CCIF to launch the Read Once command, a 4-byte Read Once record is read from the program flash IFR and stored in the FCCOB register. The CCIF flag is set after the Read Once operation completes. Valid record index values for the Read Once command range from 0x00 to 0x0F. During execution of the Read Once command, any attempt to read addresses within the program flash block containing this 64-byte field returns invalid data. The Read Once command can be executed any number of times.

Table 25-57. 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]

25.4.12.11 Program Once Command

The Program Once command enables programming to a reserved 64-byte field in the program flash IFR (see Program Flash IFR Map and Program Once Field). Access to the Program Once field is via 16 records, each 4 bytes long. The Program Once field can be

read using the Read Once command (see Read Once Command) or using the Read Resource command (see Read Resource Command). Each Program Once record can be programmed only once since the program flash IFR cannot be erased.

Table 25-58. Program Once Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x43 (PGMONCE)
1	Program Once record index (0x00 - 0x0F)
2	Not Used
3	Not Used
4	Program Once Byte 0 value
5	Program Once Byte 1 value
6	Program Once Byte 2 value
7	Program Once Byte 3 value

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.

The reserved program flash IFR location accessed by the Program Once command cannot be erased and any attempt to program one of these records when the existing value is not Fs (erased) is not allowed. Valid record index values for the Program Once command range from 0x00 to 0x0F. During execution of the Program Once command, any attempt to read addresses within the program flash block containing this 64-byte field returns invalid data.

Table 25-59. 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, the Program Once command is allowed to execute again on that same record.

25.4.12.12 Erase All Blocks Command

The Erase All Blocks operation erases all flash memory, initializes the FlexRAM, verifies all memory contents, and releases MCU security.

Table 25-60. 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, program flash swap IFR space, data flash memory, data flash IFR space, EEPROM backup memory, and FlexRAM, then verifies that all are erased.

If the flash memory module verifies that all flash memories and the FlexRAM were properly erased, security is released by setting the FSEC[SEC] field to the unsecure state and the FCNFG[RAMRDY] bit is set. The Erase All Blocks command aborts if any flash or FlexRAM region is protected. The swap indicator address in each program flash block is not implicitly protected from the Erase All Blocks operation. The security byte and all other contents of the flash configuration field (see Flash Configuration Field Description) are erased by the Erase All Blocks command. If the erase-verify fails, the FSTAT[MGSTAT0] bit is set. The CCIF flag is set after the Erase All Blocks operation completes.

Table 25-61. Erase All Blocks Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
Any region of the program flash memory, data flash memory, or FlexRAM is protected	FSTAT[FPVIOL]
Any errors have been encountered during the verify operation[1]	FSTAT[MGSTAT0]

1. User margin read may be run using the Read 1s All Blocks command to verify all bits are erased.

25.4.12.12.1 Triggering an Erase All External to the Flash Memory Module

The functionality of the Erase All Blocks command is also available in an uncommanded fashion outside of the flash memory. Refer to the device's Chip Configuration details for information on this functionality.

Before invoking the external erase all function, the 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, program flash swap IFR space, data flash memory, data flash IFR space, EEPROM backup, and FlexRAM regardless of the protection settings or if the swap system has been initialized. If the post-erase verify passes, the routine then releases security by setting the FSEC[SEC] field register to the unsecure state and the FCNFG[RAMRDY] bit sets. The security byte in the Flash Configuration Field is also programmed to the unsecure state. The status of the erase-all

request is reflected in the FCNFG[ERSAREQ] bit. The FCNFG[ERSAREQ] bit is cleared once the operation completes and the normal FSTAT error reporting is available as described in Erase All Blocks Command.

CAUTION

Since the IFR Swap Field in the program flash swap IFR containing the swap indicator address is erased during the Erase All Blocks command operation, the swap system becomes uninitialized. The Swap Control command must be run with the initialization code to set the swap indicator address and initialize the swap system.

25.4.12.13 Verify Backdoor Access Key Command

The Verify Backdoor Access Key command only executes if the mode and security conditions are satisfied (see Flash Commands by Mode). Execution of the Verify Backdoor Access Key command is further qualified by the FSEC[KEYEN] bits. The Verify Backdoor Access Key command releases security if user-supplied keys in the FCCOB match those stored in the Backdoor Comparison Key bytes of the Flash Configuration Field (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 25-62. Verify Backdoor Access Key Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]	Flash Configuration Field Offset Address
0	0x45 (VFYKEY)	
1-3	Not Used	
4	Key Byte 0	0x0_0000
5	Key Byte 1	0x0_0001
6	Key Byte 2	0x0_0002
7	Key Byte 3	0x0_0003
8	Key Byte 4	0x0_0004
9	Key Byte 5	0x0_0005
A	Key Byte 6	0x0_0006
B	Key Byte 7	0x0_0007

After clearing CCIF to launch the Verify Backdoor Access Key command, the 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 25-63. 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]

25.4.12.14 Swap Control Command

The Swap Control command handles specific activities associated with swapping the two logical program flash memory blocks within the memory map.

Table 25-64. Swap Control Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x46 (SWAP)
1	Flash address [23:16]
2	Flash address [15:8]
3	Flash address [7:0] [1]
4	Swap Control Code: 0x01 - Initialize Swap System 0x02 - Set Swap in Update State 0x04 - Set Swap in Complete State 0x08 - Report Swap Status
Returned values	

Table continues on the next page...

Table 25-64. Swap Control Command FCCOB Requirements (continued)

FCCOB Number	FCCOB Contents [7:0]
5	Current Swap State: 0x00 - Uninitialized 0x01 - Ready 0x02 - Update 0x03 - Update-Erased 0x04 - Complete
6	Current Swap Block Status: 0x00 - Program flash block 0 at 0x0_0000 0x01 - Program flash block 1 at 0x0_0000
7	Next Swap Block Status (after any reset): 0x00 - Program flash block 0 at 0x0_0000 0X01 - Program flash block 1 at 0x0_0000

1. Must be longword-aligned (Flash address [1:0] = 00).

Upon clearing CCIF to launch the Swap Control command, the flash memory module will handle swap-related activities based on the swap control code provided in FCCOB4 as follows:

- 0x01 (Initialize Swap System to UPDATE-ERASED State) - After verifying that the current swap state is UNINITIALIZED and that the flash address provided is in Program flash block 0 but not in the Flash Configuration Field, the flash address (shifted with bits[1:0] removed) will be programmed into the IFR Swap Field found in program flash swap IFR. After the swap indicator address has been programmed into the IFR Swap Field, the swap enable word will be programmed to 0x0000. After the swap enable word has been programmed, the swap indicator, located within the Program flash block 0 address provided, will be programmed to 0xFF00.
- 0x02 (Progress Swap to UPDATE State) - After verifying that the current swap state is READY and that the flash address provided matches the one stored in the IFR Swap Field, the swap indicator located within bits [15:0] of the flash address in the currently active program flash block will be programmed to 0xFF00.
- 0x04 (Progress Swap to COMPLETE State) - After verifying that the current swap state is UPDATE-ERASED and that the flash address provided matches the one stored in the IFR Swap Field, the swap indicator located within bits [15:0] of the flash address in the currently active program flash block will be programmed to 0x0000. Before executing with this swap control code, the user must erase the non-active swap indicator using the Erase Flash Block or Erase Flash Sector commands and update the application code or data as needed. The non-active swap indicator will

be checked at the erase verify level and if the check fails, the current swap state will be changed to UPDATE with FSTAT[ACCERR] set.

- 0x08 (Report Swap System Status) - After verifying that the flash address provided matches the one stored in the IFR Swap Field, the status of the swap system will be reported as follows:
  - FCCOB5 (Current Swap State) - indicates the current swap state based on the status of the swap enable word and the swap indicators. If the FSTAT[MGSTAT0] flag is set after command completion, the swap state returned was not successfully transitioned from and the appropriate swap command code must be attempted again. If the current swap state is UPDATE and the non-active swap indicator is 0xFFFF, the current swap state is changed to UPDATE-ERASED.
  - FCCOB6 (Current Swap Block Status) - indicates which program flash block is currently located at relative flash address 0x0_0000.
  - FCCOB7 (Next Swap Block Status) - indicates which program flash block will be located at relative flash address 0x0_0000 after the next reset of the flash memory module.

NOTE

It is recommended that the user execute the Swap Control command to report swap status (code 0x08) after any reset to determine if issues with the swap system were detected during the swap state determination procedure.

NOTE

It is recommended that the user write 0xFF to FCCOB5, FCCOB6, and FCCOB7 since the Swap Control command will not always return the swap state and status fields when an access error is detected.

The swap indicators are implicitly protected from being programmed during Program Longword or Program Section command operations and are implicitly unprotected during Swap Control command operations. The swap indicators are implicitly protected from being erased during Erase Flash Block and Erase Flash Sector command operations unless the swap indicator being erased is in the non-active program flash block and the swap system is in the UPDATE or UPDATE-ERASED state. Once the swap system has been initialized, the Erase All Blocks command can be used to uninitialize the swap system.

Table 25-65. Swap Control Command Error Handling

Error Condition	Swap Control Code	Error Bit
Command not available in current mode/security[1]	All	FSTAT[ACCERR]
Flash address is not in program flash block 0	All	FSTAT[ACCERR]
Flash address is in the Flash Configuration Field	All	FSTAT[ACCERR]
Flash address is not longword aligned	All	FSTAT[ACCERR]
Flash address does not match the swap indicator address in the IFR	2, 4	FSTAT[ACCERR]
Swap initialize requested when swap system is not in the uninitialized state	1	FSTAT[ACCERR]
Swap update requested when swap system is not in the ready state	2	FSTAT[ACCERR]
Swap complete requested when swap system is not in the update-erased state	4	FSTAT[ACCERR]
An undefined swap control code is provided	-	FSTAT[ACCERR]
Any errors have been encountered during the swap determination and program-verify operations	1, 2, 4	FSTAT[MGSTAT0]
Any brownouts were detected during the swap determination procedure	8	FSTAT[MGSTAT0]

1. Returned fields will not be updated, i.e. no swap state or status reporting

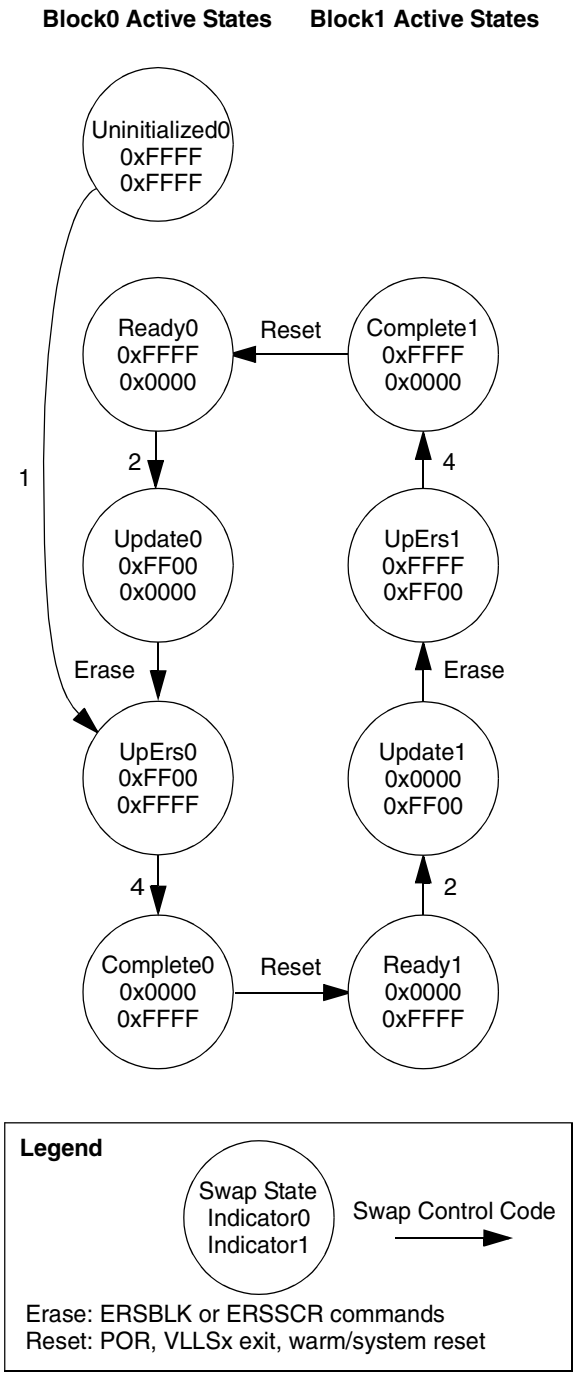


Figure 25-34. Valid Swap State Sequencing

Table 25-66. Swap State Report Mapping

Case	Swap Enable Field[1]	Swap Indicator 0[1]	Swap Indicator 1[1]	Swap State[2]	State Code	MGSTAT0	Active Block
1	0xFFFF	-	-	Uninitialized	0	0	0
2	0x0000	0xFF00	0x0000	Update	2	0	0
3	0x0000	0xFF00-	0xFFFF	Update-Erased	3	0	0
4	0x0000	0x0000	0xFFFF[3]	Complete[4]	4	0	0
5	0x0000	0x0000	0xFFFF	Ready[5]	1	0	1
6	0x0000	0x0000	0xFF00	Update	2	0	1
7	0x0000	0xFFFF	0xFF00	Update-Erased	3	0	1
8	0x0000	0xFFFF[3]	0x0000	Complete[4]	4	0	1
9	0x0000	0xFFFF	0x0000	Ready[5]	1	0	0
10	0xXXXX	-	-	Uninitialized	0	1	0
11	0x0000	0xFFFF	0xFFFF	Uninitialized	0	1	0
12	0x0000	0xFFXX	0xFFFF	Ready	1	1	0
13	0x0000	0xFFXX	0x0000	Ready	1	1	0
14[6]	0x0000	0xXXXX	0x0000	Ready	1	1	0
15[6]	0x0000	0xFFFF	0xFFXX	Ready	1	1	1
16	0x0000	0x0000	0xFFXX	Ready	1	1	1
17[6]	0x0000	0x0000	0xXXXX	Ready	1	1	1
18	0x0000	0xFF00	0xFFFF[7]	Update	2	1	0
19	0x0000	0xFF00	0xXXXX	Update	2	1	0
20	0x0000	0xFF(00)	0xFFXX	Update	2	1	0
21[6]	0x0000	0x0000	0x0000	Update	2	1	0
22[6]	0x0000	0xXXXX	0xXXXX	Update	2	1	0
23	0x0000	0xFFFF[7]	0xFF00	Update	2	1	1
24	0x0000	0xXXXX	0xFF00	Update	2	1	1
25	0x0000	0xFFXX	0xFF(00)	Update	2	1	1
26	0x0000	0xXX00	0xFFFF	Update-Erased	3	1	0
27	0x0000	0xXXXX	0xFFFF	Update-Erased	3	1	0
28	0x0000	0xFFFF	0xXX00	Update-Erased	3	1	1
29	0x0000	0xFFFF	0xXXXX	Update-Erased	3	1	1

1. 0xXXXX, 0xFFXX, 0xXX00 indicates a non-valid value was read; 0xFF(00) indicates more 0's than other indicator (if same number of 0's, then swap system defaults to block 0 active)
2. Cases 10-29 due to brownout (abort) detected during program or erase steps related to swap
3. Must read 0xFFFF with erase verify level before transition to Complete allowed
4. No reset since successful Swap Complete execution
5. Reset after successful Swap Complete execution
6. Not a valid case
7. Fails to read 0xFFFF at erase verify level

25.4.12.14.1 Swap State Determination

During the reset sequence, the state of the swap system is determined by evaluating the IFR Swap Field in the program flash swap IFR and the swap indicators located in each of the program flash blocks at the swap indicator address stored in the IFR Swap Field.

Table 25-67. Program Flash Swap IFR Fields

Address Range	Size (Bytes)	Field Description
0x00 – 0x01	2	Swap Enable Word
0x02 – 0x03	2	Swap Indicator Address
0x04 – 0xFF	252	Reserved

25.4.12.15 Program Partition Command

The Program Partition command prepares the FlexNVM block for use as data flash, EEPROM backup, or a combination of both and initializes the FlexRAM. The Program Partition command must not be launched from flash memory, since flash memory resources are not accessible during Program Partition command execution.

CAUTION

While different partitions of the FlexNVM are available, the intention is that a single partition choice is used throughout the entire lifetime of a given application. The FlexNVM Partition Code choices affect the endurance and data retention characteristics of the device.

Table 25-68. Program Partition Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x80 (PGMPART)
1	Not Used
2	Not Used
3	Not Used
4	EEPROM Data Size Code[1]
5	FlexNVM Partition Code[2]

1. See Table 25-69 and EEPROM Data Set Size
2. See Table 25-70

Table 25-69. Valid EEPROM Data Set Size Codes

EEPROM Data Size Code (FCCOB4)[1]		EEPROM Data Set Size (Bytes)
FCCOB4[5:4]	FCCOB4[EEESIZE]	
11	0xF	0[2]
11	0x9	32
11	0x8	64
11	0x7	128
11	0x6	256
11	0x5	512
11	0x4	1024
11	0x3	2048

1. FCCOB4[7:6] = 00
2. EEPROM Data Set Size must be set to 0 bytes when the FlexNVM Partition Code is set for no EEPROM.

Table 25-70. Valid FlexNVM Partition Codes

FlexNVM Partition Code (FCCOB5[DEPART])[1]	Data flash Size (KB)	EEPROM backup Size (KB)
0000	32	0
0001	24	8
0010	16	16
0011	0	32
1000	0	32
1001	8	24
1010	16	16
1011	32	0

1. FCCOB5[7:4] = 0000

After clearing CCIF to launch the Program Partition command, the flash memory module first verifies that the EEPROM Data Size Code and FlexNVM Partition Code in the data flash IFR are erased. If erased, the Program Partition command erases the contents of the FlexNVM memory. If the FlexNVM is to be partitioned for EEPROM backup, the allocated EEPROM backup sectors are formatted for EEPROM use. Finally, the partition codes are programmed into the data flash IFR using the values provided. The Program Partition command also verifies that the partition codes read back correctly after programming. If the FlexNVM is partitioned for EEPROM backup, the EEERDY flag will set with RAMRDY clear. If the FlexNVM is not partitioned for EEPROM backup, the RAMRDY flag will set with EEERDY clear. The CCIF flag is set after the Program Partition operation completes.

Prior to launching the Program Partition command, the data flash IFR must be in an erased state, which can be accomplished by executing the Erase All Blocks command or by an external request (see Erase All Blocks Command). The EEPROM Data Size Code and FlexNVM Partition Code are read using the Read Resource command (see Read Resource Command).

Table 25-71. Program Partition Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
The EEPROM data size and FlexNVM partition code bytes are not initially 0xFFFF	FSTAT[ACCERR]
Invalid EEPROM Data Size Code is entered (see Table 25-69 for valid codes)	FSTAT[ACCERR]
Invalid FlexNVM Partition Code is entered (see Table 25-70 for valid codes)	FSTAT[ACCERR]
FlexNVM Partition Code = full data flash (no EEPROM) and EEPROM Data Size Code allocates FlexRAM for EEPROM	FSTAT[ACCERR]
FlexNVM Partition Code allocates space for EEPROM backup, but EEPROM Data Size Code allocates no FlexRAM for EEPROM	FSTAT[ACCERR]
FCCOB4[7:6] != 00	FSTAT[ACCERR]
FCCOB5[7:4] != 0000	FSTAT[ACCERR]
Any errors have been encountered during the verify operation	FSTAT[MGSTAT0]

25.4.12.16 Set FlexRAM Function Command

The Set FlexRAM Function command changes the function of the FlexRAM:

- When not partitioned for EEPROM, the FlexRAM is typically used as traditional RAM.
- When partitioned for EEPROM, the FlexRAM is typically used to store EEPROM data.

Table 25-72. Set FlexRAM Function Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x81 (SETRAM)
1	FlexRAM Function Control Code (see Table 25-73)

Table 25-73. FlexRAM Function Control

FlexRAM Function Control Code	Action
0xFF	Make FlexRAM available as RAM: • Clear the FCNFG[EEERDY] and FCNFG[RAMRDY] flags • Write a background of ones to all FlexRAM locations • Set the FCNFG[RAMRDY] flag
0x00	Make FlexRAM available for EEPROM: • Clear the FCNFG[EEERDY] and FCNFG[RAMRDY] flags • Write a background of ones to all FlexRAM locations • Copy-down existing EEPROM data to FlexRAM • Set the FCNFG[EEERDY] flag

After clearing CCIF to launch the Set FlexRAM Function command, the flash memory module sets the function of the FlexRAM based on the FlexRAM Function Control Code.

When making the FlexRAM available as traditional RAM, the flash memory module clears the FCNFG[EEERDY] and FCNFG[RAMRDY] flags, overwrites the contents of the entire FlexRAM with a background pattern of all ones, and sets the FCNFG[RAMRDY] flag. The state of the FEPROT register does not prevent the FlexRAM from being overwritten. When the FlexRAM is set to function as a RAM, normal read and write accesses to the FlexRAM are available. When large sections of flash memory need to be programmed, e.g. during factory programming, the FlexRAM can be used as the Section Program Buffer for the Program Section command (see Program Section Command).

When making the FlexRAM available for EEPROM, the flash memory module clears the FCNFG[EEERDY] and FCNFG[RAMRDY] flags, overwrites the contents of the FlexRAM allocated for EEPROM with a background pattern of all ones, and copies the existing EEPROM data from the EEPROM backup record space to the FlexRAM. After completion of the EEPROM copy-down, the FCNFG[EEERDY] flag is set. When the FlexRAM is set to function as EEPROM, normal read and write access to the FlexRAM is available, but writes to the FlexRAM also invoke EEPROM activity. The CCIF flag is set after the Set FlexRAM Function operation completes.

Table 25-74. Set FlexRAM Function Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
FlexRAM Function Control Code is not defined	FSTAT[ACCERR]
FlexRAM Function Control Code is set to make the FlexRAM available for EEPROM, but FlexNVM is not partitioned for EEPROM	FSTAT[ACCERR]

25.4.13 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 (FTFL_FSEC) details.

Table 25-75. FSEC register fields

FSEC field	Description
KEYEN	Backdoor Key Access
MEEN	Mass Erase Capability
FSLACC	Freescale Factory Access
SEC	MCU security

25.4.13.1 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 25-76. 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 and Read 1s All Blocks commands.

25.4.13.2 Changing the Security State

The security state out of reset can be permanently changed by programming the security byte of the flash configuration field. This assumes that you are starting from a mode where the necessary program flash erase and program commands are available and that

the region of the program flash containing the flash configuration field is unprotected. If the flash security byte is successfully programmed, its new value takes affect after the next chip reset.

25.4.13.2.1 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.

25.4.14 Reset Sequence

On each system reset the flash memory module executes a sequence which establishes initial values for the flash block configuration parameters, FPROT, FDPROT, FEPROT, FOPT, and FSEC registers and the FCNFG[SWAP, PFLSH, RAMRDY, EEERDY] bits.

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 26
EzPort

26.1 Overview

The EzPort module is a serial flash programming interface that allows In-System Programming (ISP) of flash memory contents on a 32 bit general-purpose microcontroller. Memory contents can be read, erased, and programmed from an external source in a format that is compatible with many stand-alone flash memory chips, without necessitating the removal of the microcontroller from the system.

26.1.1 Introduction

The following figure is a high level block diagram of the EzPort.

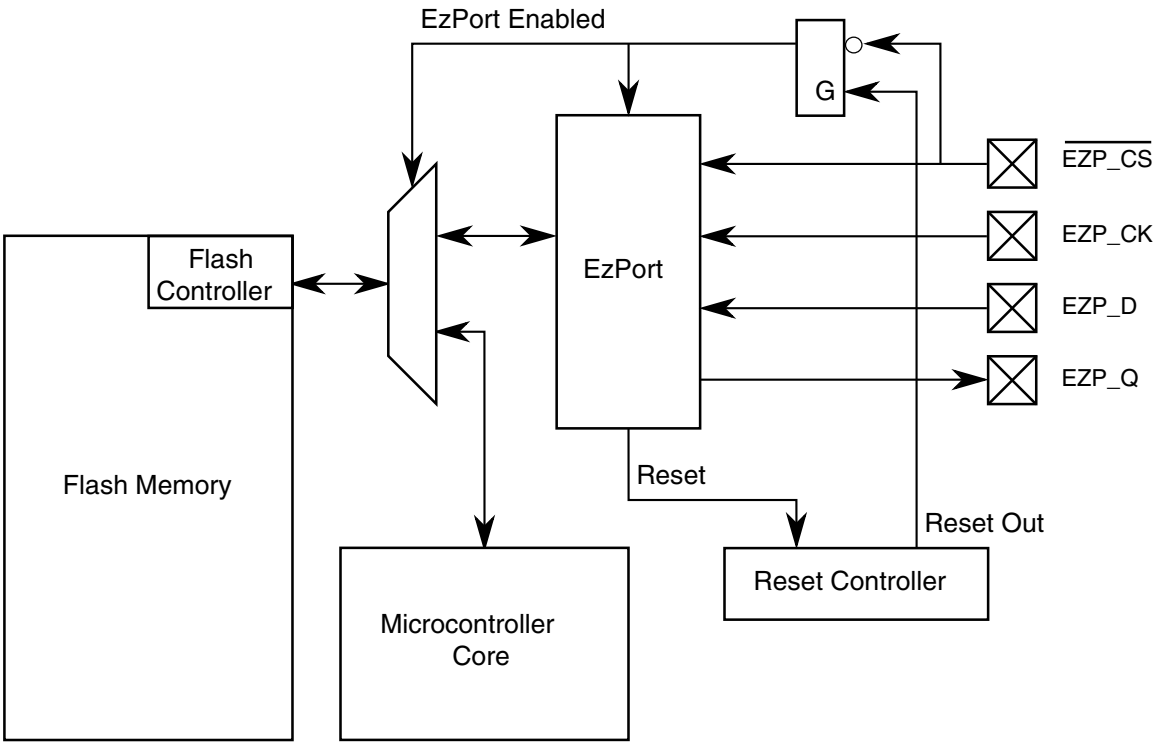


Figure 26-1. EzPort block diagram

26.1.2 Features

EzPort includes the following features:

- Serial interface that is compatible with a subset of the SPI format.
- Ability to read, erase, and program flash memory.
- Ability to reset the microcontroller, allowing it to boot from the flash memory after the memory has been configured.

26.1.3 Modes of operation

The EzPort can operate in one of two modes, enabled or disabled.

- Enabled — When enabled, the EzPort steals access to the flash memory, preventing access from other cores or peripherals. The rest of the microcontroller is disabled to avoid conflicts. The flash is configured for NVM Special mode.
- Disabled — When the EzPort is disabled, the rest of the microcontroller can access flash memory as normal.

The EzPort provides a simple interface to connect an external device to the flash memory on board a 32 bit microcontroller. The interface itself is compatible with the SPI interface, with the EzPort operating as a slave, running in either of the two following modes. The data is transmitted with the most significant bit first.

- CPOL = 0, CPHA = 0
- CPOL = 1, CPHA = 1

Commands are issued by the external device to erase, program, or read the contents of the flash memory. The serial data out from the EzPort is tri-stated unless data is being driven. This allows the signal to be shared among several different EzPort (or compatible) devices in parallel, as long as they have different chip-selects.

26.2 External signal description

The following table contains a list of EzPort external signals, and the following sections explain the signals in detail.

Table 26-1. EzPort external signal description

Name	Description	I/O
EZP_CK	EzPort Clock	Input
$\overline{\text{EZP\_CS}}$	EzPort Chip Select	Input
EZP_D	EzPort Serial Data In	Input
EZP_Q	EzPort Serial Data Out	Output

26.2.1 EzPort Clock (EZP_CK)

EZP_CK is the serial clock for data transfers. The serial data in (EZP_D) and chip select ($\overline{\text{EZP\_CS}}$) are registered on the rising edge of EZP_CK, while serial data out (EZP_Q) is driven on the falling edge of EZP_CK.

The maximum frequency of the EzPort clock is half the system clock frequency for all commands except when executing the Read Data or Read FlexRAM commands. When executing these commands, the EzPort clock has a maximum frequency of one-eighth the system clock frequency.

26.2.2 EzPort Chip Select ($\overline{\text{EZP\_CS}}$)

$\overline{\text{EZP\_CS}}$ is the chip select for signaling the start and end of serial transfers. If, while $\overline{\text{EZP\_CS}}$ is asserted, the microcontroller's reset out signal is negated, EzPort is enabled out of reset; otherwise it is disabled. After EzPort is enabled, asserting $\overline{\text{EZP\_CS}}$ commences a serial data transfer, which continues until $\overline{\text{EZP\_CS}}$ is negated again. The negation of $\overline{\text{EZP\_CS}}$ indicates the current command is finished and resets the EzPort state machine so that it is ready to receive the next command.

26.2.3 EzPort Serial Data In (EZP_D)

EZP_D is the serial data in for data transfers. EZP_D is registered on the rising edge of EZP_CK. All commands, addresses, and data are shifted in most significant bit first. When the EzPort is driving output data on EZP_Q, the data shifted in EZP_D is ignored.

26.2.4 EzPort Serial Data Out (EZP_Q)

EZP_Q is the serial data out for data transfers. EZP_Q is driven on the falling edge of EZP_CK. It is tri-stated unless $\overline{\text{EZP\_CS}}$ is asserted and the EzPort is driving data out. All data is shifted out most significant bit first.

26.3 Command definition

The EzPort receives commands from an external device and translates the commands into flash memory accesses. The following table lists the supported commands.

Table 26-2. EzPort commands

Command	Description	Code	Address Bytes	Data Bytes	Accepted when secure?
WREN	Write Enable	0x06	0	0	Yes
WRDI	Write Disable	0x04	0	0	Yes
RDSR	Read Status Register	0x05	0	1	Yes
READ	Flash Read Data	0x03	3[1]	1+	No
FAST_READ	Flash Read Data at High Speed	0x0B	3[1]	1+[2]	No
SP	Flash Section Program	0x02	3[3]	8 - SECTION[4]	No
SE	Flash Sector Erase	0xD8	3[3]	0	No
BE	Flash Bulk Erase	0xC7	0	0	Yes[5]
RESET	Reset Chip	0xB9	0	0	Yes
WRFCCOB	Write FCCOB Registers	0xBA	0	12	Yes[6]

Table continues on the next page...

Table 26-2. EzPort commands (continued)

Command	Description	Code	Address Bytes	Data Bytes	Accepted when secure?
FAST_RDFCCOB	Read FCCOB registers at high speed	0xBB	0	1 - 12[2]	No
WRFLEXRAM	Write FlexRAM	0xBC	3[1]	4	No
RDFLEXRAM	Read FlexRAM	0xBD	3[1]	1+	No
FAST_RDFLEXRAM	Read FlexRAM at high speed	0xBE	3[1]	1+[2]	No

1. Address must be 32-bit aligned (two LSBs must be zero).
2. One byte of dummy data must be shifted in before valid data is shifted out.
3. Address must be 64-bit aligned (three LSBs must be zero).
4. Please see the Flash Memory chapter for a definition of section size. Total number of data bytes programmed must be a multiple of 8.
5. Bulk Erase is accepted when security is set and only when the BEDIS status field is not set.
6. The flash will be in NVM Special mode, restricting the type of commands that can be executed through WRITE_FCCOB when security is enabled.

26.3.1 Command descriptions

This section describes the module commands.

26.3.1.1 Write Enable

The Write Enable (WREN) command sets the write enable register field in the EzPort status register. The write enable field must be set for a write command (SP, SE, BE, WRFCCOB, or WRFLEXRAM) to be accepted. The write enable register field clears on reset, on a Write Disable command, and at the completion of write command. This command must not be used if a write is already in progress.

26.3.1.2 Write Disable

The Write Disable (WRDI) command clears the write enable register field in the status register. This command must not be used if a write is already in progress.

26.3.1.3 Read Status Register

The Read Status Register (RDSR) command returns the contents of the EzPort status register.

Table 26-3. EzPort status register

	7	6	5	4	3	2	1	0
R	FS	WEF			FLEXRAM	BEDIS	WEN	WIP
W								
Reset:	0/1[1]	0	0	0	0/1[2]	0/1[3]	0	1[4]

1. Reset value reflects the status of flash security out of reset.
2. Reset value reflects FlexNVM flash partitioning. If FlexNVM flash has been paritioned for EEPROM, this field is set immediately after reset. Note that FLEXRAM is cleared after the EzPort initialization sequence completes, as indicated by clearing of WIP.
3. Reset value reflects whether bulk erase is enabled or disabled out of reset.
4. Initial value of WIP is 1, but the value clears to 0 after EzPort initialization is complete.

Table 26-4. EzPort status register field description

Field	Description
0 WIP	Write in progress. Sets after a write command (SP, SE, BE, WRFCCOB, or WRFLEXRAM) is accepted and clears after the flash memory has completed all operations associated with the write command, as indicated by the Command Complete Interrupt Flag (CCIF) inside the flash. This field is also asserted on reset and cleared when EzPort initialization is complete. Only the Read Status Register (RDSR) command is accepted while a write is in progress. 0 = Write is not in progress. Accept any command. 1 = Write is in progress. Only accept RDSR command.
1 WEN	Write enable Enables the write comman that follows. It is a control field that must be set before a write command (SP, SE, BE, WRFCCOB, or WRFLEXRAM) is accepted. Is set by the Write Enable (WREN) command and cleared by reset or a Write Disable (WRDI) command. This field also clears when the flash memory has completed all operations associated with the command. 0 = Disables the following write command. 1 = Enables the following write command.
2 BEDIS	Bulk erase disable Indicates whether bulk erase (BE) is disabled when flash is secure. 0 = BE is enabled. 1 = BE is disabled if FS is also set. Attempts to issue a BE command will result in the WEF flag being set.
3 FLEXRAM	FlexRAM mode Indicates the current mode of the FlexRAM. Valid only when WIP is cleared. 0 = FlexRAM is in RAM mode. RD/WRFLEXRAM command can be used to read/write data in FlexRAM. 1 = FlexRAM is in EEPROM mode. SP command is not accepted. RD/WRFLEXRAM command can be used to read/write data in the FlexRAM.

Table continues on the next page...

Table 26-4. EzPort status register field description (continued)

Field	Description
6 WEF	Write error flag Indicates whether there has been an error while executing a write command (SP, SE, BE, WRFCCOB, or WRFLEXRAM). The WEF flag will set if Flash Access Error Flag (ACCERR), Flash Protection Violation (FPVIOL), or Memory Controller Command Completion Status (MGSTAT0) inside the flash memory is set at the completion of the write command. See the flash memory chapter for further description of these flags and their sources. The WEF flag clears after a Read Status Register (RDSR) command. 0 = No error on previous write command. 1 = Error on previous write command.
7 FS	Flash security Indicates whether the flash is secure. See Table 26-2 for the list of commands that will be accepted when flash is secure. Flash security can be disabled by performing a BE command. 0 = Flash is not secure. 1 = Flash is secure.

26.3.1.4 Read Data

The Read Data (READ) command returns data from the flash memory or FlexNVM, depending on the initial address specified in the command word. The initial address must be 32-bit aligned with the two LSBs being zero.

Data continues being returned for as long as the EzPort chip select ($\overline{\text{EZP\_CS}}$) is asserted, with the address automatically incrementing. In this way, the entire contents of flash can be returned by one command. Attempts to read from an address which does not fall within the valid address range for the flash memory regions returns unknown data. See Flash memory map for EzPort access.

For this command to return the correct data, the EzPort clock (EZP_CK) must run at the internal system clock divided by eight or slower. This command is not accepted if the WEF, WIP, or FS field in the EzPort status register is set.

26.3.1.5 Read Data at High Speed

The Read Data at High Speed (FAST_READ) command is identical to the READ command, except for the inclusion of a dummy byte following the address bytes and before the first data byte is returned.

This command can be run with an EzPort clock (EZP_CK) frequency of half the internal system clock frequency of the microcontroller or slower. This command is not accepted if the WEF, WIP, or FS field in the EzPort status register is set.

26.3.1.6 Section Program

The Section Program (SP) command programs up to one section of flash memory that has previously been erased. Please see the Flash Memory chapter for a definition of section size. The starting address of the memory to program is sent after the command word and must be a 64-bit aligned address with the three LSBs being zero).

As data is shifted in, the EzPort buffers the data in FlexRAM before executing an SP command within the flash. For this reason, the number of bytes to be programmed must be a multiple of 8 and up to one flash section can be programmed at a time. For more details, see the Flash Block Guide.

Attempts to program more than one section, across a sector boundary or from an initial address which does not fall within the valid address range for the flash causes the WEF flag to set. See Flash memory map for EzPort access.

This command requires the FlexRAM to be configured for traditional RAM operation. By default, after entering EzPort mode, the FlexRAM is configured for traditional RAM operation. If the user reconfigures FlexRAM for EEPROM operation, then the user should use the WRFCCOB command to configure FlexRAM back to traditional RAM operation before issuing an SP command. See the Flash Memory chapter for details on how the FlexRAM function is modified.

This command is not accepted if the WEF, WIP, FLEXRAM, or FS field is set or if the WEN field is not set in the EzPort status register.

26.3.1.7 Sector Erase

The Sector Erase (SE) command erases the contents of one sector of flash memory. The three byte address sent after the command byte can be any address within the sector to erase, but must be a 64-bit aligned address (the three LSBs must be zero). Attempts to erase from an initial address which does not fall within the valid address range (see Flash memory map for EzPort access) for the flash results in the WEF flag being set.

This command is not accepted if the WEF, WIP or FS field is set or if the WEN field is not set in the EzPort status register.

26.3.1.8 Bulk Erase

The Bulk Erase (BE) command erases the entire contents of flash memory, ignoring any protected sectors or flash security. Flash security is disabled upon successful completion of the BE command.

Attempts to issue a BE command while the BEDIS and FS fields are set results in the WEF flag being set in the EzPort status register. Also, this command is not accepted if the WEF or WIP field is set or if the WEN field is not set in the EzPort status register.

26.3.1.9 EzPort Reset Chip

The Reset Chip (RESET) command forces the chip into the reset state. If the EzPort chip select ($\overline{\text{EZP\_CS}}$) pin is asserted at the end of the reset period, EzPort is enabled; otherwise, it is disabled. This command allows the chip to boot up from flash memory after being programmed by an external source.

This command is not accepted if the WIP field is set in the EzPort status register.

26.3.1.10 Write FCCOB Registers

The Write FCCOB Registers (WRFCCOB) command allows the user to write to the flash common command object registers and execute any command allowed by the flash.

NOTE

When security is enabled, the flash is configured in NVM Special mode, restricting the commands that can be executed by the flash.

After receiving 12 bytes of data, EzPort writes the data to the FCCOB 0-B registers in the flash and then automatically launches the command within the flash. If greater or less than 12 bytes of data is received, this command has unexpected results and may result in the WEF flag being set.

This command is not accepted if the WEF or WIP field is set or if the WEN field is not set in the EzPort status register.

26.3.1.11 Read FCCOB Registers at High Speed

The Read FCCOB Registers at High Speed (FAST_RDFCCOB) command allows the user to read the contents of the flash common command object registers. After receiving the command, EzPort waits for one dummy byte of data before returning FCCOB register data starting at FCCOB 0 and ending with FCCOB B.

This command can be run with an EzPort clock (EZP_CK) frequency half the internal system clock frequency of the microcontroller or slower. Attempts to read greater than 12 bytes of data returns unknown data. This command is not accepted if the WEF, WIP, or FS fields in the EzPort status register are 1.

26.3.1.12 Write FlexRAM

The Write FlexRAM (WRFLEXRAM) command allows the user to write four bytes of data to the FlexRAM. If the FlexRAM is configured for EEPROM operation, the WRFLEXRAM command can effectively be used to create data records in the EEPROM flash memory.

By default, after entering EzPort mode, the FlexRAM is configured for traditional RAM operation and functions as direct RAM. The user can alter the FlexRAM configuration by using WRFCCOB to execute a Set FlexRAM or Program Partition command within the flash.

The address of the FlexRAM location to be written is sent after the command word and must be a 32-bit aligned address (the two LSBs must be zero). Attempts to write an address which does not fall within the valid address range for the FlexRAM results in the value of the WEF flag being 1. See Flash memory map for EzPort access for more information.

After receiving four bytes of data, EzPort writes the data to the FlexRAM. If greater or less than four bytes of data is received, this command has unexpected results and may result in the value of the WEF flag being 1.

This command is not accepted if the WEF, WIP or FS fields are 1 or if the WEN field is 0 in the EzPort status register.

26.3.1.13 Read FlexRAM

The Read FlexRAM (RDFLEXRAM) command returns data from the FlexRAM. If the FlexRAM is configured for EEPROM operation, the RDFLEXRAM command can effectively be used to read data from EEPROM flash memory.

Data continues being returned for as long as the EzPort chip select ($\overline{\text{EZP\_CS}}$) is asserted, with the address automatically incrementing. In this way, the entire contents of FlexRAM can be returned by one command.

The initial address must be 32-bit aligned (the two LSBs must be zero). Attempts to read from an address which does not fall within the valid address range for the FlexRAM returns unknown data. See Flash memory map for EzPort access for more information.

For this command to return the correct data, the EzPort clock (EZP_CK) must run at the internal system clock divided by eight or slower. This command is not accepted if the WEF, WIP, or FS fields in the EzPort status register are set.

26.3.1.14 Read FlexRAM at High Speed

The Read FlexRAM at High Speed (FAST_RDFLEXRAM) command is identical to the RDFLEXRAM command, except for the inclusion of a dummy byte following the address bytes and before the first data byte is returned.

This command can be run with an EzPort clock (EZP_CK) frequency up to and including half the internal system clock frequency of the microcontroller. This command is not accepted if the WEF, WIP, or FS fields in the EzPort status register are set.

26.4 Flash memory map for EzPort access

The following table shows the flash memory map for access through EzPort.

NOTE

The flash block address map for access through EzPort may not conform to the system memory map. Changes are made to allow the EzPort address width to remain 24 bits.

Table 26-5. Flash Memory Map for EzPort Access

Valid start address	Size	Flash block	Valid commands
0x0000_0000	See device's chip configuration details	Flash	READ, FAST_READ, SP, SE, BE
0x0080_0000	See device's chip configuration details	FlexNVM	READ, FAST_READ, SP, SE, BE
0x0000_0000	See device's chip configuration details	FlexRAM	RDFLEXRAM, FAST_RDFLEXRAM, WRFLEXRAM, BE

Chapter 27
Cryptographic Acceleration Unit (CAU)

27.1 Introduction

The cryptographic acceleration unit (CAU) is a ColdFire coprocessor implementing a set of specialized operations in hardware to increase the throughput of software-based encryption and hashing functions.

27.2 Block Diagram

This simplified block diagram illustrates the CAU.

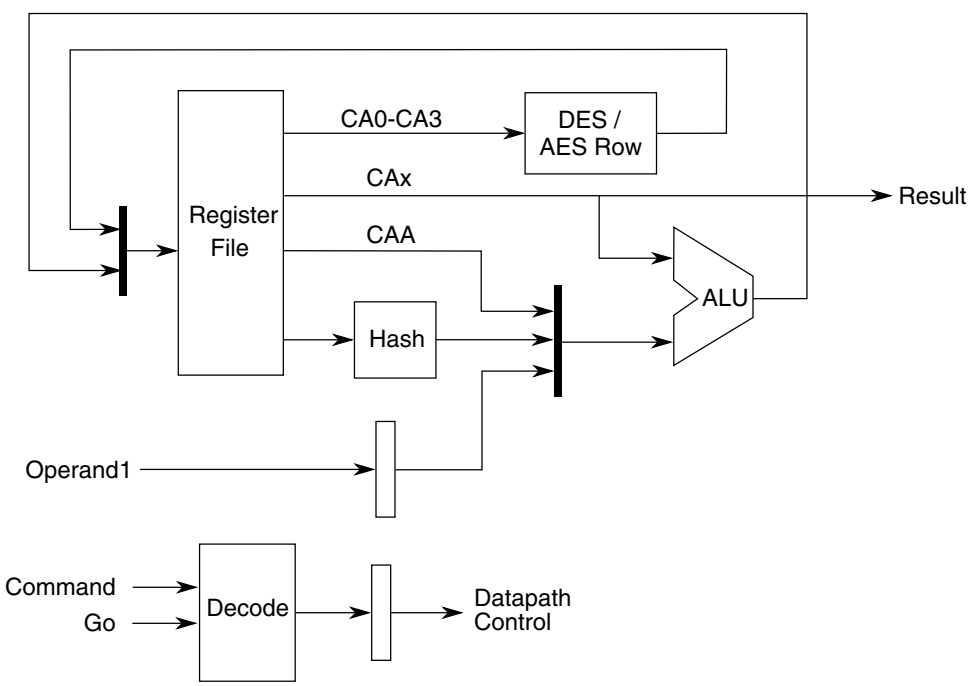


Figure 27-1. Top-level CAU block diagram

27.3 Overview

The set of implemented algorithms provides excellent support for network security standards, such as SSL and IPsec. Additionally, using the CAU efficiently permits the implementation of any higher level functions or modes of operation, such as HMAC, CBC, and so on based on the supported algorithms.

The cryptographic algorithms are implemented partially in software with only functions critical to increasing performance implemented in hardware. The CAU allows for efficient, fine-grained partitioning of functions between hardware and software:

- Implement the innermost security kernel functions using the coprocessor instructions.
- Implement higher level functions in software by using the standard processor instructions.

This partitioning of functions is key to minimizing size of the CAU while maintaining a high level of throughput. Using software for some functions also simplifies the CAU design. The CAU implements a set of coprocessor commands that operate on a register file of 32-bit registers. It is tightly coupled to the ColdFire core and there is no local memory or external interface.

27.4 Features

The CAU includes the following distinctive features:

- Supports DES, 3DES, AES, MD5, SHA-1, and SHA-256 algorithms
- Simple, flexible programming model

27.5 Register definition

The CAU contains multiple registers used by each of the supported algorithms. The following table shows registers that are applicable to each supported algorithm and indicates the corresponding letter designations for each algorithm. For more information on these letter designations, see the algorithm specifications.

Code	Register	DES	AES	MD5	SHA-1	SHA-256
0	CAU Status Register (CASR)	—	—	—	—	—
1	CAU Accumulator (CAA)	—	—	a	T	T
2	General-Purpose Register 0 (CA0)	C	W0	—	A	A
3	General-Purpose Register 1 (CA1)	D	W1	b	B	B
4	General-Purpose Register 2 (CA2)	L	W2	c	C	C
5	General-Purpose Register 3 (CA3)	R	W3	d	D	D
6	General-Purpose Register 4 (CA4)	—	—	—	E	E
7	General-Purpose Register 5 (CA5)	—	—	—	W	F
8	General-Purpose Register 6 (CA6)	—	—	—	—	G
9	General-Purpose Register 7 (CA7)	—	—	—	—	H
10	General-Purpose Register 8 (CA8)	—	—	—	—	W/T_1

The CAU supports only 32-bit operations and register accesses. All registers support read, write, and ALU operations. However, only bits 1–0 of the CASR are writeable. Bits 31–2 of the CASR must be written as 0 for compatibility with future versions of the CAU.

NOTE

In the following table, the "address" or "offset" refers to the command code value for the CAU registers.

CAU memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Status Register (CAU_CASR)	32	R/W	2000_0000h	27.5.1/524
1	Accumulator (CAU_CAA)	32	R/W	0000_0000h	27.5.2/525
2	General Purpose Register (CAU_CA0)	32	R/W	0000_0000h	27.5.3/525
3	General Purpose Register (CAU_CA1)	32	R/W	0000_0000h	27.5.3/525
4	General Purpose Register (CAU_CA2)	32	R/W	0000_0000h	27.5.3/525
5	General Purpose Register (CAU_CA3)	32	R/W	0000_0000h	27.5.3/525
6	General Purpose Register (CAU_CA4)	32	R/W	0000_0000h	27.5.3/525
7	General Purpose Register (CAU_CA5)	32	R/W	0000_0000h	27.5.3/525
8	General Purpose Register (CAU_CA6)	32	R/W	0000_0000h	27.5.3/525
9	General Purpose Register (CAU_CA7)	32	R/W	0000_0000h	27.5.3/525
A	General Purpose Register (CAU_CA8)	32	R/W	0000_0000h	27.5.3/525

27.5.1 Status Register (CAU_CASR)

CASR contains the status and configuration for the CAU.

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

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

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

CAU_CASR field descriptions

Field	Description
31–28 VER	CAU Version Indicates CAU version. 0x1 Initial CAU version. 0x2 Second version, added support for SHA-256 algorithm (This is the value on this device).

Table continues on the next page...

CAU_CASR field descriptions (continued)

Field	Description
27–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DPE	DES Parity Error Indicates whether the DES parity error is detected. 0 No error detected. 1 DES key parity error detected.
0 IC	Illegal Command Indicates an illegal instruction has been executed. 0 No illegal commands issued. 1 Illegal command issued.

27.5.2 Accumulator (CAU_CAA)

Commands use the CAU accumulator for storage of results and as an operand for the cryptographic algorithms.

Address: 0h base + 1h offset = 1h

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	ACC																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAU_CAA field descriptions

Field	Description
31–0 ACC	Accumulator Stores results of various CAU commands.

27.5.3 General Purpose Register (CAU_CA*n*)

The General Purpose Register is used in the CAU commands for storage of results and as operands for various cryptographic algorithms.

Address: 0h base + 2h offset + (1d × i), where i=0d to 8d

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

CAU_CA*n* field descriptions

Field	Description
31–0 CAn	General Purpose Registers Used by the CAU commands. Some cryptographic operations work with specific registers.

27.6 Functional description

This section discusses the programming model and operation of the CAU.

27.6.1 Programming Model

The CAU is an instruction-level coprocessor. It has a dedicated register file, a specialized ALU, and specialized units for performing cryptographic operations. The CAU design uses a simple, flexible accumulator-based architecture. Most commands, including load and store, can specify any register in the register file. Some cryptographic operations work with specific registers.

27.6.2 Coprocessor Instructions

Operation of the CAU is controlled via standard ColdFire coprocessor load (cp0ld) and store (cp0st) instructions. The CAU has a dedicated register file accessed using these instructions. The load instruction loads CAU registers and specifies CAU operations. The store instruction stores CAU registers. The example assembler syntax for the CAU is:

```
cp0ld.l         <ea>,<CMD>          ; coprocessor load
cp0st.l         <ea>,<CMD>          ; coprocessor store
```

The <ea> field specifies the source operand (operand1) for load instructions and destination (result) for store instructions. The basic ColdFire addressing modes {Rn, (An), -(An), (An)+, (d16,An)} are supported for this field. The <CMD> field is a 9-bit value that specifies the CAU command for an instruction. The CAU commands table shows how the CAU supports a single command (STR) for store instructions and 21 commands for the load instructions. The CAU only supports longword operations. A CAU command can be issued every clock cycle.

27.6.3 CAU commands

The CAU supports the commands shown in the following table. All other encodings are reserved. The CASR[IC] bit is set if an undefined command is issued. A specific illegal command (ILL) is defined to allow software self-checking. Reserved commands must not be issued so as to ensure compatibility with future implementations.

The CMD field specifies the 9-bit CAU opcode for the operation command.

See Assembler equate values for a set of assembly constants used in the command descriptions here. If supported by the assembler, macros can also be created for each instruction. The value CA*x* should be interpreted as any CAU register (CASR, CAA, and CA*n*) and the <ea> field as one of the supported ColdFire addressing modes {Rn, (An), -(An), (An)+, (d16,An)}. For example, the instruction to add the value from the core register D1 to the CAU register CA0 is:

```
cp0ld.l         %d1,#ADR+CA0        ; CA0=CA0+d1
```

Table 27-14. CAU commands

Type	Command name	Description	CMD									Operation
			8	7	6	5	4	3	2	1	0	
cp0ld	CNOP	No Operation	0x000									—
cp0ld	LDR	Load Reg	0x01					CAx				Op1 → CAx
cp0str	STR	Store Reg	0x02					CAx				CAx → Result
cp0ld	ADR	Add	0x03					CAx				CAx + Op1 → CAx
cp0ld	RADR	Reverse and Add	0x04					CAx				CAx + ByteRev(Op1) → CAx
cp0ld	ADRA	Add Reg to Acc	0x05					CAx				CAx + CAA → CAA
cp0ld	XOR	Exclusive Or	0x06					CAx				CAx ^ Op1 → CAx
cp0ld	ROTL	Rotate Left	0x07					CAx				(CAx <<< (Op1 % 32)) | (CAx >>> (32 - (Op1 % 32))) → CAx
cp0ld	MVRA	Move Reg to Acc	0x08					CAx				CAx → CAA
cp0ld	MVAR	Move Acc to Reg	0x09					CAx				CAA → CAx
cp0ld	AESS	AES Sub Bytes	0x0A					CAx				SubBytes(CAx) → CAx
cp0ld	AESIS	AES Inv Sub Bytes	0x0B					CAx				InvSubBytes(CAx) → CAx
cp0ld	AESC	AES Column Op	0x0C					CAx				MixColumns(CAx)^Op1→ CAx
cp0ld	AESIC	AES Inv Column Op	0x0D					CAx				InvMixColumns(CAx^Op1) → CAx
cp0ld	AESR	AES Shift Rows	0x0E0									ShiftRows(CA0-CA3) → CA0-CA3
cp0ld	AESIR	AES Inv Shift Rows	0x0F0									InvShiftRows(CA0-CA3)→ CA0-CA3
cp0ld	DESR	DES Round	0x10					IP	FP	KS[1:0]		DES Round(CA0-CA3) →CA0-CA3

Table continues on the next page...

Table 27-14. CAU commands (continued)

Type	Command name	Description	CMD									Operation
			8	7	6	5	4	3	2	1	0	
cp0ld	DESK	DES Key Setup	0x11					0	0	CP	DC	DES Key Op(CA0-CA1)→ CA0-CA1 Key Parity Error & CP → CASR[1]
cp0ld	HASH	Hash Function	0x12					0	HF[2:0]			Hash Func(CA1-CA3)+CAA→ CAA
cp0ld	SHS	Secure Hash Shift	0x130									CAA <<< 5→ CAA, CAA→CA0, CA0→CA1, CA1 <<< 30 → CA2, CA2→CA3, CA3→CA4
cp0ld	MDS	Message Digest Shift	0x140									CA3→CAA, CAA→CA1, CA1→CA2, CA2→CA3,
cp0ld	SHS2	Secure Hash Shift 2	0x150									CAA→CA0, CA0→CA1, CA1 → CA2, CA2→CA3, CA3 + CA8 →CA4, CA4 → CA5, CA5 → CA6, CA6 → CA7
cp0ld	ILL	Illegal Command	0x1F0									0x1→CASR[IC]

27.6.3.1 Coprocessor No Operation (CNOP)

```
cp0ld.l   #CNOP
```

The CNOP command is the coprocessor no-op defined by the ColdFire coprocessor definition for synchronization. It is not actually issued to the coprocessor from the core.

27.6.3.2 Load Register (LDR)

```
cp0ld.l   <ea>,#LDR+CAx
```

The LDR command loads CAx with the source data specified by <ea>.

27.6.3.3 Store Register (STR)

```
cp0st.l   <ea>,#STR+CAx
```

The STR command returns the value of CAx to the destination specified by <ea>.

27.6.3.4 Add to Register (ADR)

```
cp0ld.l   <ea>,#ADR+CAx
```

The ADR command adds the source operand specified by <ea> to CAx and stores the result in CAx.

27.6.3.5 Reverse and Add to Register (RADR)

```
cp0ld.l   <ea>,#RADR+CAx
```

The RADR command performs a byte reverse on the source operand specified by <ea>, adds that value to CAx, and stores the result in CAx. The table below shows an example.

Table 27-15. RADR command example

Operand	CAx before	CAx after
0x0102_0304	0xA0B0_C0D0	0xA4B3_C2D1

27.6.3.6 Add Register to Accumulator (ADRA)

```
cp0ld.l   #ADRA+CAx
```

The ADRA command adds CAx to CAA and stores the result in CAA.

27.6.3.7 Exclusive Or (XOR)

```
cp0ld.l   <ea>,#XOR+CAx
```

The XOR command performs an exclusive-or of the source operand specified by <ea> with CAx and stores the result in CAx.

27.6.3.8 Rotate Left (ROTL)

```
cp0ld.l   <ea>,#ROTL+CAx
```

ROTL rotates the CAx bits to the left with the result stored back to CAx. The number of bits to rotate is the value specified by <ea> modulo 32.

27.6.3.9 Move Register to Accumulator (MVRA)

```
cp0ld.l   #MVRA+CAx
```

The MVRA command moves the value from the source register CAx to the destination register CAA.

27.6.3.10 Move Accumulator to Register (MVAR)

```
cp0ld.l   #MVAR+CAx
```

The MVAR command moves the value from source register CAA to the destination register CAx.

27.6.3.11 AES Substitution (AESS)

```
cp0ld.l   #AESS+CAx
```

The AESS command performs the AES byte substitution operation on CAx and stores the result back to CAx.

27.6.3.12 AES Inverse Substitution (AESIS)

```
cp0ld.l   #AESIS+CAx
```

The AESIS command performs the AES inverse byte substitution operation on CAx and stores the result back to CAx.

27.6.3.13 AES Column Operation (AESC)

```
cp0ld.l   <ea>,#AESC+CAx
```

The AESC command performs the AES column operation on the contents of CAx. It then performs an exclusive-or of that result with the source operand specified by <ea> and stores the result in CAx.

27.6.3.14 AES Inverse Column Operation (AESIC)

```
cp0ld.l   <ea>,#AESIC+CAx
```

The AESIC command performs an exclusive-or operation of the source operand specified by <ea> on the contents of CAx followed by the AES inverse mix column operation on that result and stores the result back in CAx.

27.6.3.15 AES Shift Rows (AESR)

```
cp0ld.l   #AESR
```

The AESR command performs the AES shift rows operation on registers CA0, CA1, CA2, and CA3. The table below shows an example.

Table 27-16. AESR command example

Register	Before	After
CA0	0x0102_0304	0x0106_0B00
CA1	0x0506_0708	0x050A_0F04
CA2	0x090A_0B0C	0x090E_0308
CA3	0x0D0E_0F00	0x0D02_070C

27.6.3.16 AES Inverse Shift Rows (AESIR)

```
cp0ld.l   #AESIR
```

The AESIR command performs the AES inverse shift rows operation on registers CA0, CA1, CA2, and CA3. The table below shows an example.

Table 27-17. AESIR command example

Register	Before	After
CA0	0x0106_0B00	0x0102_0304
CA1	0x050A_0F04	0x0506_0708
CA2	0x090E_0308	0x090A_0B0C
CA3	0x0D02_070C	0x0D0E_0F00

27.6.3.17 DES Round (DESR)

```
cp0ld.l   #DESR+{IP}+{FP}+{KSx}
```

The DESR command performs a round of the DES algorithm and a key schedule update with the following source and destination designations: CA0=C, CA1=D, CA2=L, CA3=R. If the IP bit is set, DES initial permutation performs on CA2 and CA3 before the round operation. If the FP bit is set, DES final permutation, that is, inverse initial permutation, performs on CA2 and CA3 after the round operation. The round operation uses the source values from registers CA0 and CA1 for the key addition operation. The KSx field specifies the shift for the key schedule operation to update the values in CA0 and CA1. The following table defines the specific shift function performed based on the KSx field.

Table 27-18. Key shift function codes

KSx code	KSx define	Shift function
0	KSL1	Left 1
1	KSL2	Left 2
2	KSR1	Right 1
3	KSR2	Right 2

27.6.3.18 DES Key setup (DESK)

```
cp0ld.l   #DESK+{CP}+{DC}
```

The DESK command performs the initial key transformation, permuted choice 1, defined by the DES algorithm on CA0 and CA1 with CA0 containing bits 1–32 of the key and CA1 containing bits 33–64 of the key[1] . If the DC bit is set, no shift operation performs and the values C_0 and D_0 store back to CA0 and CA1, respectively. The DC bit must be set for decrypt operations. If the DC bit is not set, a left shift by one also occurs and the values C_1 and D_1 store back to CA0 and CA1, respectively. The DC bit should be cleared for encrypt operations. If the CP bit is set and a key parity error is detected, CASR[DPE] bit is set; otherwise, it is cleared.

27.6.3.19 Hash Function (HASH)

```
cp0ld.l   #HASH+HFx
```

The HASH command performs a hashing operation on a set of registers and adds that result to the value in CAA and stores the result in CAA. The specific hash function performed is based on the HFx field as defined in the table below.

1. The DES algorithm numbers the most significant bit of a block as bit 1 and the least significant as bit 64.

This table uses the following terms:

- ROTR^n(CAx): rotate CAx register right *n* times
- SHR^n(CAx): shift CAx register right *n* times

Table 27-19. Hash Function codes

HFx code	HFx define	Hash Function	Hash logic
0	HFF	MD5 F()	(CA1 & CA2) | ($\overline{CA1}$ & CA3)
1	HFG	MD5 G()	(CA1 & CA3) | (CA2 & $\overline{CA3}$)
2	HFH	MD5 H(), SHA Parity()	CA1 ^ CA2 ^ CA3
3	HFI	MD5 I()	CA2 ^ (CA1 | $\overline{CA3}$)
4	HFC	SHA Ch()	(CA1 & CA2) ^ ($\overline{CA1}$ & CA3)
5	HFM	SHA Maj()	(CA1 & CA2) ^ (CA1 & CA3) ^ (CA2 & CA3)
6	HF2C	SHA-256 Ch()	(CA4 & CA5) ^ ($\overline{CA1}$ & CA6)
7	HF2M	SHA-256 Maj()	(CA0 & CA1) ^ (CA0 & CA2) ^ (CA1 & CA2)
8	HF2S	SHA-256 Sigma 0	ROTR^2(CA0) ^ ROTR^{13}(CA0) ^ ROTR^{22}(CA0)
9	HF2T	SHA-256 Sigma 1	ROTR^6(CA4) ^ ROTR^{11}(CA4) ^ ROTR^{25}(CA4)
A	HF2U	SHA-256 Sigma 0	ROTR^7(CA8) ^ ROTR^{18}(CA8) ^ SHR^3(CA8)
B	HF2V	SHA-256 Sigma 1	ROTR^{17}(CA8) ^ ROTR^{19}(CA8) ^ SHR^{10}(CA8)

27.6.3.20 Secure Hash Shift (SHS)

```
cp0ld.l   #SHS
```

The SHS command does a set of parallel register-to-register move and shift operations for implementing SHA-1. The following source and destination assignments are made:

Register	Value prior to command	Value after command executes
CA4	CA4	CA3
CA3	CA3	CA2
CA2	CA2	CA1<<<30
CA1	CA1	CA0
CA0	CA0	CAA
CAA	CAA	CAA<<<5

27.6.3.21 Message Digest Shift (MDS)

```
cp0ld.l   #MDS
```

The MDS command does a set of parallel register-to-register move operations for implementing MD5. The following source and destination assignments are made:

Register	Value prior to command	Value after command executes
CA3	CA3	CA2
CA2	CA2	CA1
CA1	CA1	CAA
CAA	CAA	CA3

27.6.3.22 Secure Hash Shift 2 (SHS2)

```
cp0ld.l   #SHS2
```

The SHS2 command does an addition and a set of register to register moves in parallel for implementing SHA-256. The following source and destination assignments are made:

Register	Value prior to command	Value after command executes
CA7	CA7	CA6
CA6	CA6	CA5
CA5	CA5	CA4
CA4	CA4	CA3+CA8
CA3	CA3	CA2
CA2	CA2	CA1
CA1	CA1	CA0
CA0	CA0	CAA

27.6.3.23 Illegal command (ILL)

```
cp0ld.l   #ILL
```

The ILL command is a specific illegal command that sets CASR[IC]. All other illegal commands are reserved for use in future implementations.

27.7 Application/initialization information

This section discusses how to initialize and use the CAU.

27.7.1 Code example

A code fragment is shown below as an example of how the CAU is used. This example shows the round function of the AES algorithm. Core register A0 is pointing to the key schedule.

```
     cp0ld.l   #AESS+CA0          ; sub bytes w0
     cp0ld.l   #AESS+CA1          ; sub bytes w1
     cp0ld.l   #AESS+CA2          ; sub bytes w2
     cp0ld.l   #AESS+CA3          ; sub bytes w3
     cp0ld.l   #AESR              ; shift rows
     cp0ld.l   (%a0)+,#AESC+CA0   ; mix col, add key w0
     cp0ld.l   (%a0)+,#AESC+CA1   ; mix col, add key w1
     cp0ld.l   (%a0)+,#AESC+CA2   ; mix col, add key w2
     cp0ld.l   (%a0)+,#AESC+CA3   ; mix col, add key w3
```

27.7.2 Assembler equate values

The following equates ease programming of the CAU.

```
; CAU Registers (CAx)
     .set CASR,0x0
     .set CAA,0x1
     .set CA0,0x2
     .set CA1,0x3
     .set CA2,0x4
     .set CA3,0x5
     .set CA4,0x6
     .set CA5,0x7
     .set CA6,0x8
     .set CA7,0x9
     .set CA8,0xA
; CAU Commands
     .set CNOP,0x000
     .set LDR,0x010
     .set STR,0x020
     .set ADR,0x030
     .set RADR,0x040
     .set ADRA,0x050
     .set XOR,0x060
     .set ROTL,0x070
     .set MVRA,0x080
     .set MVAR,0x090
     .set AESS,0x0A0
     .set AESIS,0x0B0
     .set AESC,0x0C0
     .set AESIC,0x0D0
     .set AESR,0x0E0
     .set AESIR,0x0F0
     .set DESR,0x100
     .set DESK,0x110
     .set HASH,0x120
     .set SHS,0x130
     .set MDS,0x140
     .set SHS2,0x150
     .set ILL,0x1F0
; DESR  Fields
     .set IP,0x08         ; initial permutation
     .set FP,0x04         ; final permutation
     .set KSL1,0x00       ; key schedule left 1 bit
     .set KSL2,0x01       ; key schedule left 2 bits
```

```
     .set KSR1,0x02        ; key schedule right 1 bit
     .set KSR2,0x03        ; key schedule right 2 bits
; DESK Field
     .set DC,0x01          ; decrypt key schedule
     .set CP,0x02          ; check parity
; HASH Functions Codes
     .set HFF,0x0          ; MD5 F() CA1&CA2 | ~CA1&CA3
     .set HFG,0x1          ; MD5 G() CA1&CA3 | CA2&~CA3
     .set HFH,0x2          ; MD5 H(), SHA Parity() CA1^CA2^CA3
     .set HFI,0x3          ; MD5 I()  CA2^(CA1|~CA3)
     .set HFC,0x4          ; SHA Ch() CA1&CA2 ^ ~CA1&CA3
     .set HFM,0x5          ; SHA Maj() CA1&CA2 ^ CA1&CA3 ^ CA2&CA3
     .set HF2C,0x6         ; SHA-256 Ch() CA4&CA5 ^ ~CA4&CA6
     .set HF2M,0x7         ; SHA-256 Maj() CA0&CA1 ^ CA0&CA2 ^ CA1&CA2
     .set HF2S,0x8         ; SHA-256 Sigma 0 ROTR2(CA0)^ROTR13(CA0)^ROTR22(CA0)
     .set HF2T,0x9         ; SHA-256 Sigma 1 ROTR6(CA4)^ROTR11(CA4)^ROTR25(CA4)
     .set HF2U,0xA         ; SHA-256 sigma 0 ROTR7(CA8)^ROTR18(CA8)^SHR3(CA8)
     .set HF2V,0xB         ; SHA-256 sigma 1 ROTR17(CA8)^ROTR19(CA8)^SHR10(CA8)
```

Chapter 28
Random Number Generator Accelerator (RNGA)

28.1 Introduction

This chapter describes the Random Number Generator Accelerator (RNGA), including a programming model, functional description, and application information.

28.1.1 Overview

The RNGA module is a digital integrated circuit capable of generating 32-bit random numbers. The random bits are generated by clocking shift registers with clocks derived from ring oscillators. The configuration of the shift registers ensures statistically good data, that is, data that looks random. The oscillators with their unknown frequencies provide the required entropy needed to create random data.

It is important to note that there is no known cryptographic proof showing that this is a secure method of generating random data. In fact, there may be an attack against the random number generator described in this document if its output is used directly in a cryptographic application (the attack is based on the linearity of the internal shift registers). In light of this, it is highly recommended that the random data produced by this module be used as an input seed to a NIST approved (based on DES or SHA-1) Pseudo Random Number Generator as defined in *NIST FIPS PUB 186-2 Appendix 3* and *NIST FIPS PUB SP 800-90*. Other sources of entropy be used along with the RNGA to generate the seed to the pseudorandom algorithm. The more random sources combined to create the seed, the better. The following is a list of sources that could be easily combined with the output of this module:

- Current time using highest precision possible.
- Mouse and keyboard motions, or equivalent if being used on a cell phone or PDA.
- Other entropy supplied directly by the user.

28.2 Modes of operation

Although the RNGA has several modes, only one is intended for use during normal operation. The Sleep mode is entered by setting the appropriate bit in the RNGA Control Register.

- Normal mode

  In this mode, the RNGA generates random data. Because this is the default mode of operation, the user is not required to change the mode before requesting random data.

- Sleep mode

  In this mode the RNGA's oscillator clocks are shut off. The mode is entered by writing to the Sleep (SLP) bit in the RNGA Control Register. When in this mode, the RNGA Output Register will not be loaded.

28.3 Memory map and register definition

This section describes the RNGA registers.

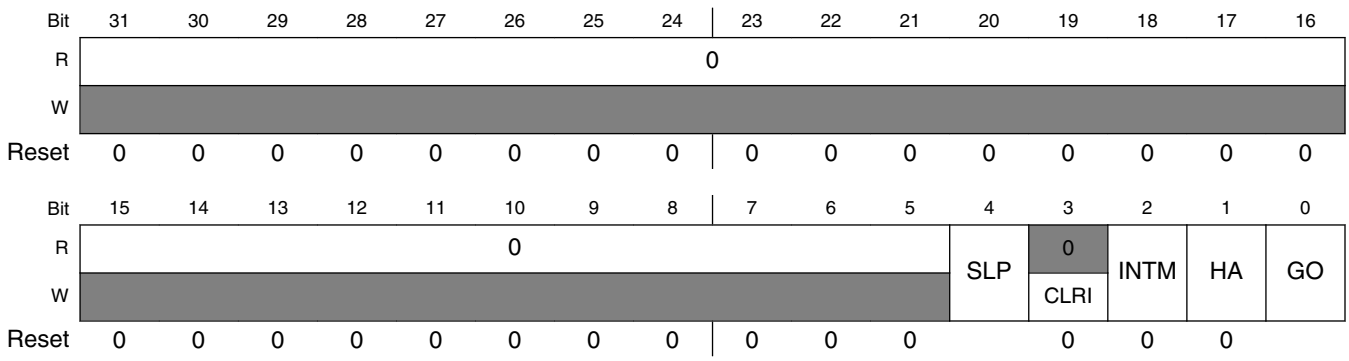
RNG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_8680	RNGA Control Register (RNG_CR)	32	R/W	0000_0000h	28.3.1/539
FFFF_8684	RNGA Status Register (RNG_SR)	32	R	0001_0000h	28.3.2/540
FFFF_8688	RNGA Entropy Register (RNG_ER)	32	W (always reads 0)	0000_0000h	28.3.3/542
FFFF_868C	RNGA Output Register (RNG_OR)	32	R	0000_0000h	28.3.4/543

28.3.1 RNGA Control Register (RNG_CR)

Immediately after reset, the RNGA begins generating entropy in its internal shift registers. Random data is not pushed to the RNGA Output Register until the GO bit in the RNGA Control register is set. After this, a random 32-bit word is pushed to the RNGA Output Register every 256 system clock cycles. If the RNGA Output Register is full, then no push will occur. In this way, the RNGA Output Register is kept as close to full as possible.

Address: FFFF_8680h base + 0h offset = FFFF_8680h



RNG_CR field descriptions

Field	Description
31–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 SLP	Sleep The RNGA can be placed in low-power mode by either asserting the module input, or by setting this bit. If either of these conditions are met, the oscillators are disabled. Clearing this bit causes the RNGA to exit Sleep mode. The RNGA Output Register is not pushed while the RNGA is in Sleep mode. 0 RNGA is not in Sleep mode. 1 RNGA is in Sleep mode.
3 CLRI	Clear Interrupt Writing a one to this bit clears the error interrupt as well as the error status bit (ERRI) in the RNGA Status Register. The bit is self clearing. 0 Do not clear the interrupt. 1 Clear the interrupt.
2 INTM	Interrupt Mask This bit masks the error interrupt. 0 Interrupt is enabled. 1 Interrupt is masked.

Table continues on the next page...

RNG_CR field descriptions (continued)

Field	Description
1 HA	High Assurance While this bit is high, a read of the RNGA Output Register while empty causes a security violation. This bit enables security violation bit (SECV) in the RNGA Status Register. This bit is sticky and can only be cleared through a hardware reset. 0 Notification of security violations is enabled. 1 Notification of security violations is masked.
0 GO	This bit must be set before the RNGA begins loading data into the RNGA Output Register. This bit is sticky and can only be cleared by a hardware reset . Setting this bit does not bring the RNGA out of Sleep mode. Furthermore, this bit does not need to be reset after exiting Sleep mode. 0 RNGA Output Register is not loaded with random data. 1 RNGA Output Register is loaded with random data.

28.3.2 RNGA Status Register (RNG_SR)

The RNGA Status Register is a read only register that reflects the internal status of the RNGA.

Address: FFFF_8680h base + 4h offset = FFFF_8684h

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OREG_LVL								0			SLP	ERRI	ORU	LRS	SECV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0

RNG_SR field descriptions

Field	Description
31–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–16 OREG_SIZE	Output Register Size This bit signals the actual size of the RNGA Output Register. In other words, this is the maximum possible RNGA Output Register Level. The bits should be interpreted as an integer. This value is set to 0x01.
15–8 OREG_LVL	Output Register Level This bit signals how many random words are currently resident in the RNGA Output Register. Only two values are possible. The bits should be interpreted as an integer (the value 0b00000001 indicates that 0x01 random word is in the RNGA Output Register, while the value 0b00000000 indicates that no random data is in the RNGA Output Register).
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 SLP	Sleep This bit reflects whether the RNGA is in Sleep mode that is, either the Sleep bit (SLP) in the RNGA Control Register is set or the “ipg_doze” input is asserted. When this bit is set, the RNGA is in Sleep mode and the oscillator clocks are inactive. While in this mode, the RNGA Output Register will not be loaded and the RNGA Output Register Level does not increase. 0 The RNGA is not in Sleep mode. 1 The RNGA is in Sleep mode.
3 ERRI	Error Interrupt This bit is always enabled and signals an RNGA Output Register underflow condition. This bit is different from the previous two bits in that it is only reset by a write to the clear interrupt bit (CLRI) in the RNGA Control Register. This bit is not masked by the Interrupt Mask bit of the RNGA Control Register. 0 The RNGA Output Register has not been read while empty. 1 The RNGA Output Register has been read while empty.
2 ORU	Output Register Underflow This bit is always enabled and signals an RNGA Output Register underflow condition. The bit is reset by reading the RNGA Status Register. 0 The RNGA Output Register has not been read while empty since last read of the RNGA Status Register. 1 The RNGA Output Register has been read while empty since last read of the RNGA Status Register.
1 LRS	Last Read Status This bit is always enabled and reflects the status of the most recent read of the RNGA Output Register. 0 Last read was performed while the RNGA Output Register was not empty. 1 Last read was performed while the RNGA Output Register was empty (underflow condition).
0 SECV	Security Violation When enabled by the High Assurance bit (HA) in the RNGA Control Register, this bit signals that a security violation has occurred. Currently, RNGA Output Register underflow is the only condition that is considered to be a security violation. The bit is sticky and can be cleared only by a hardware reset.

Table continues on the next page...

RNG_SR field descriptions (continued)

Field	Description
	0 No security violations have occured or the High Assurance bit (HA) in the RNGA Control Register is not set. 1 A security violation has occurred.

28.3.3 RNGA Entropy Register (RNG_ER)

The RNGA Entropy Register is a write-only register that allows the user to insert entropy into the RNGA. This register allows an external user to continually seed the RNGA with externally generated random data. Although the use of this register is recommended, it is also optional. The RNGA Entropy Register can be written at any time during operation.

Address: FFFF_8680h base + 8h offset = FFFF_8688h

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

RNG_ER field descriptions

Field	Description
31–0 EXT_ENT	External Entropy A write to this register allows the user to introduce 32 bits of entropy to the internal state of RNGA. Eight writes are required to introduce 256 bits of external entropy.

28.3.4 RNGA Output Register (RNG_OR)

The RNGA Output Register provides temporary storage for random data generated by the RNGA. As long as the RNGA Output Register is not empty, a read of this address will return 32 bits of random data. If the RNGA Output Register is read when it is empty, Error Interrupt (ERRI), Output Register Underflow (ORU), and Last Read Staus (LRS) bits in the RNGA Status Register are set. If the interrupt is enabled in the RNGA Control Register, an interrupt is triggered to the interrupt controller The RNGA Output Register Level field in the RNGA Status Register can be polled to monitor how many 32-bit words currently resides in the RNGA Output Register. When in Normal mode, a new random word is pushed into the RNGA Output Register every 256 system clock cycles (as long as the RNGA Output Register is not full). Polling the RNGA Status Register is very important to make sure random values are present before reading from the RNGA Output Register.

Address: FFFF_8680h base + Ch offset = FFFF_868Ch

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

RNG_OR field descriptions

Field	Description
31–0 RANDOUT	Random Output 32-bits of Random Data

28.4 Functional description

The following figure shows the RNGA has three functional blocks: output FIFO, internal bus interface, and the RNGA core/control logic blocks. The following sections describe these blocks in more detail.

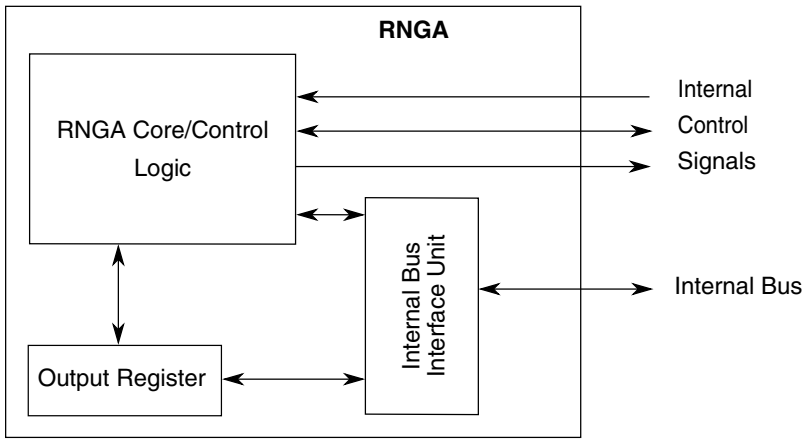


Figure 28-5. RNGA block diagram

28.4.1 RNGA Output Register

The RNGA Output Register provides temporary storage for random data generated by the RNGA Core/Control Logic block. The RNGA status register allows the user to monitor the number of random words in the RNGA Output Register through the RNGA Output Register Level field. If the user reads from the RNGA Output Register when it is empty and the interrupt is enabled, the RNGA drives an interrupt request to the interrupt controller . Polling the RNGA Status Register is very important to make sure random values are present before reading from the RNGA Output Register.

28.4.2 RNGA Core/Control Logic Block

This block contains the RNGA's Control Logic as well as its Core Engine used to generate random data.

28.4.2.1 RNGA Control Block

The Control Block contains the address decoder, all addressable registers, and control state machines for the RNGA. This block is responsible for communication with both the peripheral interface and the RNGA Output Register interface. The block also controls the Core Engine to generate random data. The general functionality of the block is as follows. After reset, entropy is generated and stored in the RNGA's shift registers. After the GO bit is set in the RNGA Control Register, the RNGA Output Register is loaded with a random word every 256 system clock cycles. The process of loading the RNGA Output Register continues as long as the RNGA Output Register is not full.

28.4.2.2 Core Engine

The Core Engine Block contains the logic used to generate random data. The logic within the Core Engine contains the internal shift registers as well as the logic used to generate the two oscillator-based clocks. The Control Block controls how the shift registers are configured as well as when the oscillator clocks are turned on.

28.5 Initialization/application information

The intended general operation of the RNGA is as follows:

1. Reset/initialize
2. Write to the RNGA Control Register and set the Interrupt Mask (INTM), High Assurance (HA), and GO bits.
3. Poll the RNGA Status Register for RNGA Output Register Level
4. Read available random data from the RNGA Output Register
5. Repeat steps 3 and 4 as needed

Chapter 29
Cyclic Redundancy Check (CRC)

29.1 Introduction

The cyclic redundancy check (CRC) module generates 16/32-bit CRC code for error detection.

The CRC module provides a programmable polynomial, WAS, and other parameters required to implement a 16-bit or 32-bit CRC standard.

The 16/32-bit code is calculated for 32 bits of data at a time.

29.1.1 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

29.1.2 Block diagram

The following is a block diagram of the CRC.

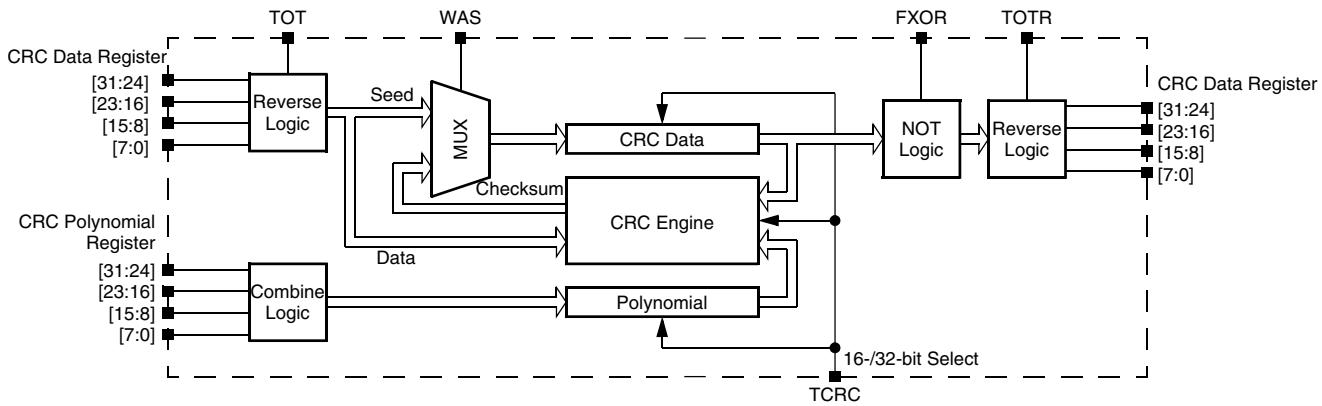


Figure 29-1. Programmable cyclic redundancy check (CRC) block diagram

29.1.3 Modes of operation

Various MCU modes affect the CRC module's functionality.

29.1.3.1 Run mode

This is the basic mode of operation.

29.1.3.2 Low-power modes (Wait or Stop)

Any CRC calculation in progress stops when the MCU enters a low-power mode that disables the module clock. It resumes after the clock is enabled or via the system reset for exiting the low-power mode. Clock gating for this module is MCU dependent.

29.2 Memory map and register descriptions

CRC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_83C0	CRC Data register (CRC_DATA)	32	R/W	FFFF_FFFFh	29.2.1/549
FFFF_83C4	CRC Polynomial register (CRC_GPOLY)	32	R/W	0000_1021h	29.2.2/550
FFFF_83C8	CRC Control register (CRC_CTRL)	32	R/W	0000_0000h	29.2.3/550

29.2.1 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: FFFF_83C0h base + 0h offset = FFFF_83C0h

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

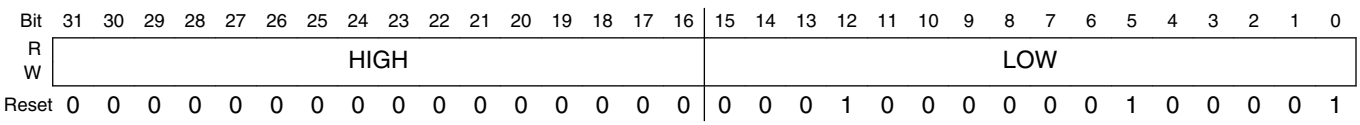
CRC_DATA field descriptions

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.
7–0 LL	CRC Low Lower Byte When CTRL[WAS] is 1, values written to this field are part of the seed value. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation.

29.2.2 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: FFFF_83C0h base + 4h offset = FFFF_83C4h



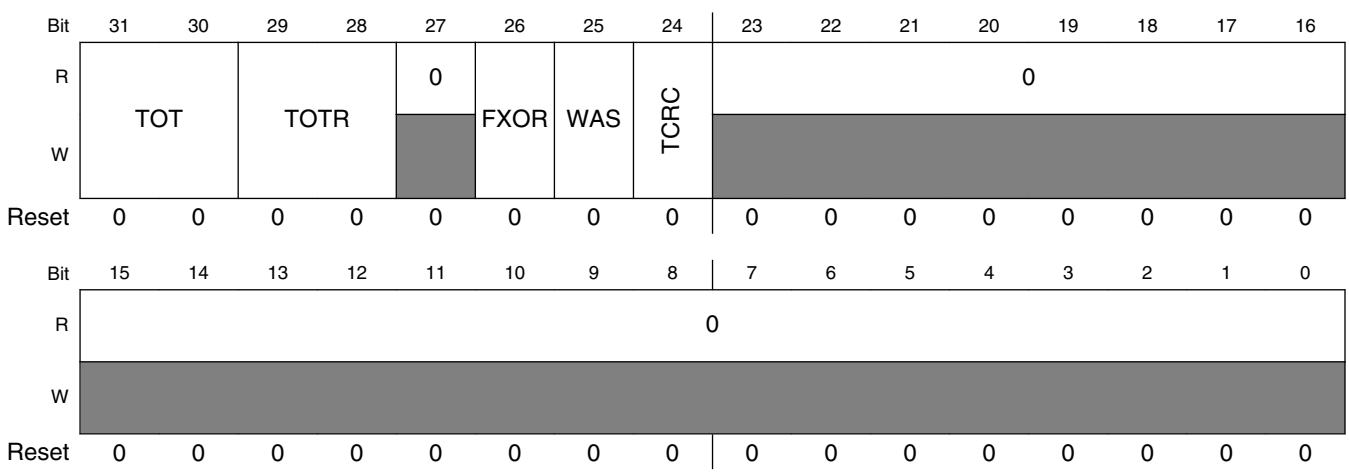
CRC_GPOLY field descriptions

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).
15–0 LOW	Low Polynominal Half-word Writable and readable in both 32-bit and 16-bit CRC modes.

29.2.3 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: FFFF_83C0h base + 8h offset = FFFF_83C8h



CRC_CTRL field descriptions

Field	Description
31–30 TOT	Type Of Transpose For Writes Define the transpose configuration of the data written to the CRC data register. See the description of the transpose feature for the available transpose options. 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 Identify the transpose configuration of the value read from the CRC Data register. See the description of the transpose feature for the available transpose options. 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.
23–0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3 Functional description

29.3.1 CRC initialization/reinitialization

To enable the CRC calculation, the user must program the WAS, POLYNOMIAL, and necessary parameters for transpose and CRC result inversion in the applicable registers. Asserting CTRL[WAS] enables the programming of the seed value into the CRC data register.

After a completed CRC calculation, reasserting CTRL[WAS] and programming a seed, whether the value is new or a previously used seed value, reinitialize the CRC module for a new CRC computation. All other parameters must be set before programming the seed value and subsequent data values.

29.3.2 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.

29.3.2.1 16-bit CRC

To compute a 16-bit CRC:

1. Clear 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 GPOLY[LOW] field. The GPOLY[HIGH] field is not usable in 16-bit CRC mode.
4. Set CTRL[WAS] to program the seed value.
5. Write a 16-bit seed to CRC[LU:LL]. CRC[HU:HL] are not used.
6. Clear CTRL[WAS] to start writing data values.
7. Write data values into CRC[HU:HL:LU:LL]. A CRC is computed on every data value write, and the intermediate CRC result is stored back into CRC[LU:LL].
8. When all values have been written, read the final CRC result from CRC[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.

29.3.2.2 32-bit CRC

To compute a 32-bit CRC:

1. Set 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 GPOLY[HIGH:LOW].
4. Set CTRL[WAS] to program the seed value.
5. Write a 32-bit seed to CRC[HU:HL:LU:LL].
6. Clear CTRL[WAS] to start writing data values.
7. Write data values into CRC[HU:HL:LU:LL]. A CRC is computed on every data value write, and the intermediate CRC result is stored back into CRC[HU:HL:LU:LL].
8. When all values have been written, read the final CRC result from CRC[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.

29.3.3 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.

29.3.3.1 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]}

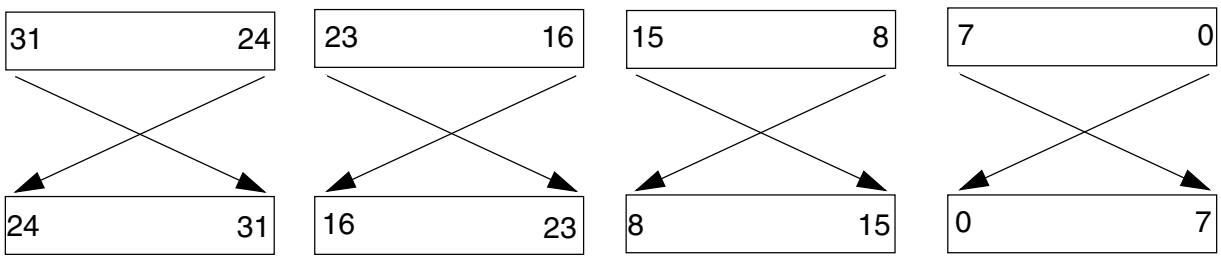


Figure 29-5. Transpose type 01

3. CTRL[TOT] or CTRL[TOTR] is 10

   Both bits in bytes and bytes are transposed.

   reg[31:0] becomes = {reg[0:7], reg[8:15],reg[16:23], reg[24:31]}

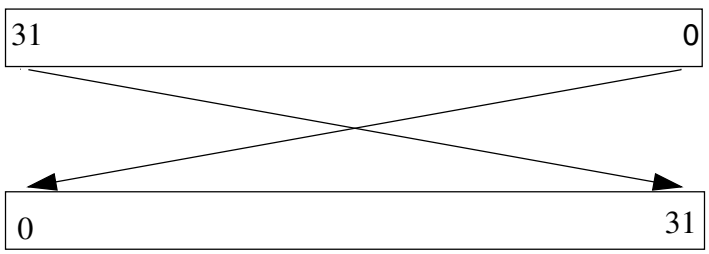


Figure 29-6. Transpose type 10

4. CTRL[TOT] or CTRL[TOTR] is 11

   Bytes are transposed, but bits are not transposed.

   reg[31:0] becomes {reg[7:0], reg[15:8], reg[23:16], reg[31:24]}

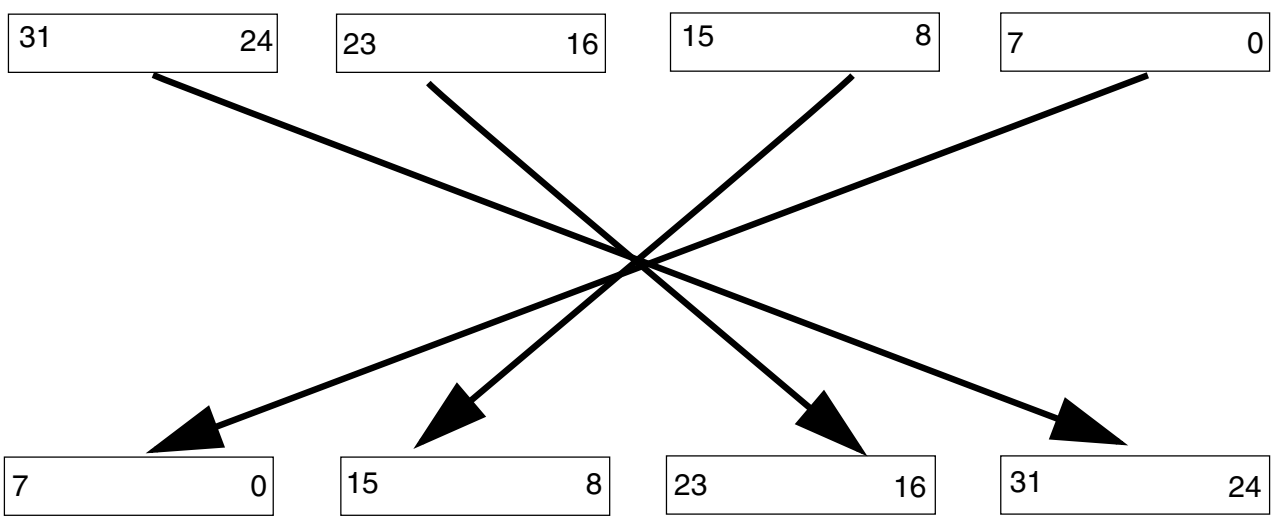


Figure 29-7. Transpose type 11

NOTE

For 8-bit and 16-bit write accesses to the CRC data register, the data is transposed with zeros on the unused byte or bytes (taking 32 bits as a whole), but the CRC is calculated on the valid byte(s) only. When reading the CRC data register for a 16-bit CRC result and using transpose options 10 and 11, the resulting value after transposition resides in the CRC[HU:HL] fields. The user software must account for this situation when reading the 16-bit CRC result, so reading 32 bits is preferred.

29.3.4 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 30
Miscellaneous and Error Status Module (MESM)

30.1 Introduction

The error detecting code (EDC) improves the fault tolerance and reliability of the internal RAM. The miscellaneous and error status module (MESM) provides the logic and programming model for system software to configure and collect information on memory errors reported by EDC as well as basic platform configuration information.

30.1.1 Overview

This device implements a parity check code in the on-chip RAM to detect single bit errors. The MESM provides a set of registers that configure the operation of the parity check logic and report any EDC errors. System software may configure the device for regions of RAM to be protected with the EDC and then query a set of read-only status and information registers for details on any errors that have been detected.

The RAM controller implements a *nibble-wide parity check code*, that is, there is a single parity bit for every 4 bits of data. This type of parity check provides protection against failures in an odd number of bits (1,3,...) but even numbers (2,4,...) in error are undetected. It is the simplest and lowest cost of any error detection code. An alternating scheme of even and odd parity bits is implemented to provide improved detection capabilities against certain classes of RAM failures.

The terms even and odd parity are defined such that the total number of "1" bits in the concatenation of the data field and associated parity bit are even or odd. For this device, the parity check is defined as follows, where PCHECK[7:0] is the parity check bits associated with the RAM:

```
PCHECK[7] is the even parity check bit associated with RAM[31:28]
PCHECK[6] is the odd parity check bit associated with RAM[27:24]
PCHECK[5] is the even parity check bit associated with RAM[23:20]
PCHECK[4] is the odd parity check bit associated with RAM[19:16]
PCHECK[3] is the even parity check bit associated with RAM[15:12]
```

```
PCHECK[2] is the odd parity check bit associated with RAM[11:8]
PCHECK[1] is the even parity check bit associated with RAM[7:4]
PCHECK[0] is the odd parity check bit associated with RAM[3:0]
```

When properly initialized and enabled, the parity check code is generated on every RAM write operation and verified on every read operation. In the event a single bit error is detected, multiple reporting mechanisms are available and defined by programmable configuration registers. There are two basic reporting mechanisms: the RAM read access can be terminated with an error (a "bus error") and/or a non-maskable EDC alert interrupt can be generated.

The EDC is implemented with 8 check bits (PCHECK[7:0]) for every 32-bit RAM longword. When enabled, the parity check logic performs the following:

1. On every RAM write, two parity check bits are generated for each 8-bit byte of write data and stored in the 32+8 (32 data bits + 8 check bits) memory. Only the bytes being written are affected.
2. If the read check is enabled, each RAM read performs the required calculation of the data and associated parity check bits to determine if a single bit error is present. Only the bytes being read are checked. If an error is detected, the RAM controller can error terminate the bus cycle and send information about the faulting access to MESM, where the address, attributes and read data are captured in fault status registers. Additionally, an EDC alert interrupt can also be asserted.

Note

For proper operation, the *RAM must be written* to store the correct parity bits before checking of reads is enabled.

The relative location of the MESM module within a simplified Version 1 ColdFire core + DMA reduced product platform is shown in the following figure. The crossbar switch (AXBS) for this platform supports 2 bus masters (core, DMA) and 3 slaves (flash, RAM/RGPIO, IPS), while the DMA includes a bus master port plus an IPS slave connection for its programming model. The MESM connections with the IPS bus and the platform RAM controller are clearly shown.

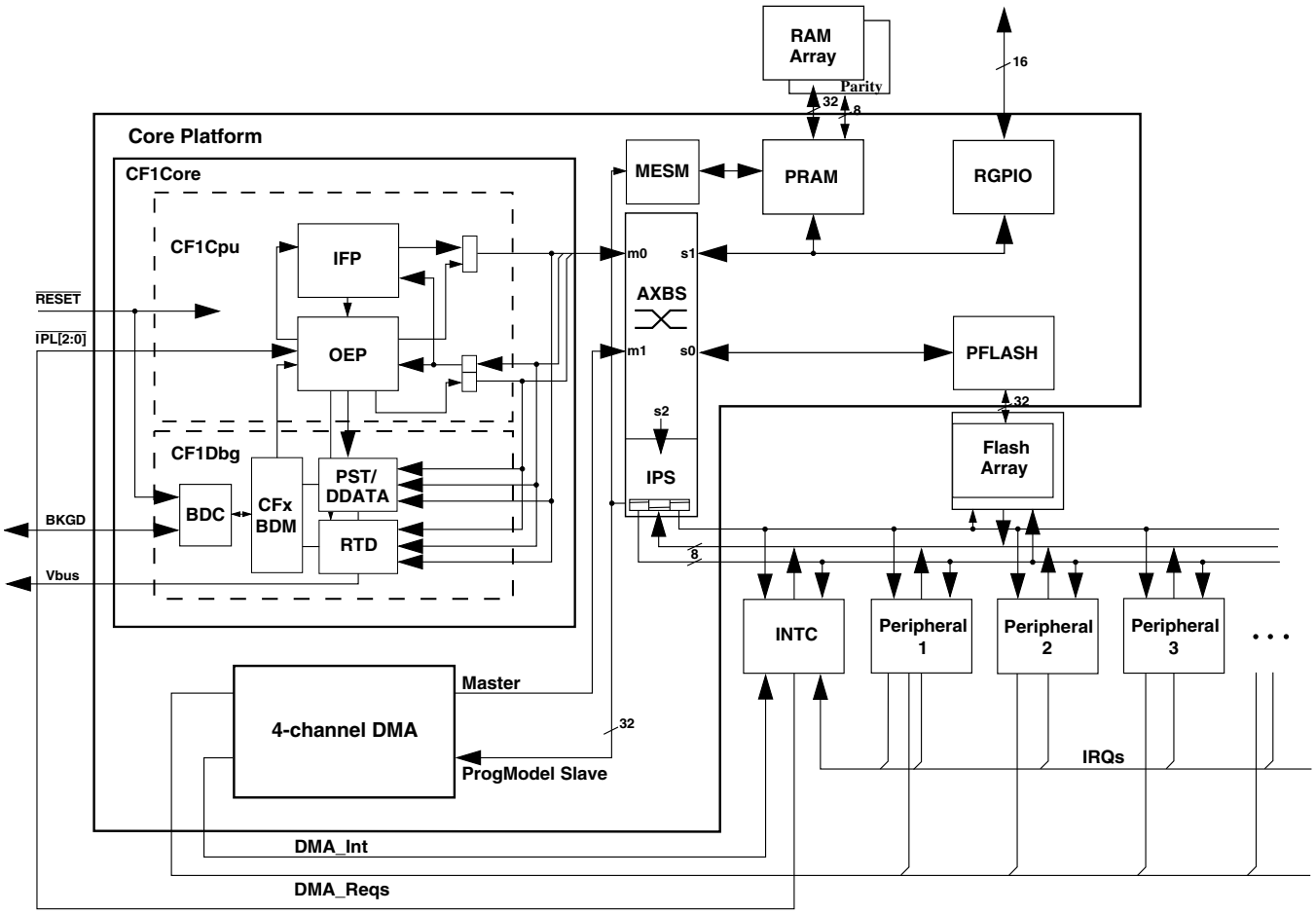


Figure 30-1. V1 ColdFire Core Platform with DMA + MESM Block Diagram

A simplified block diagram of the platform RAM controller with parity support is shown in the following figure.

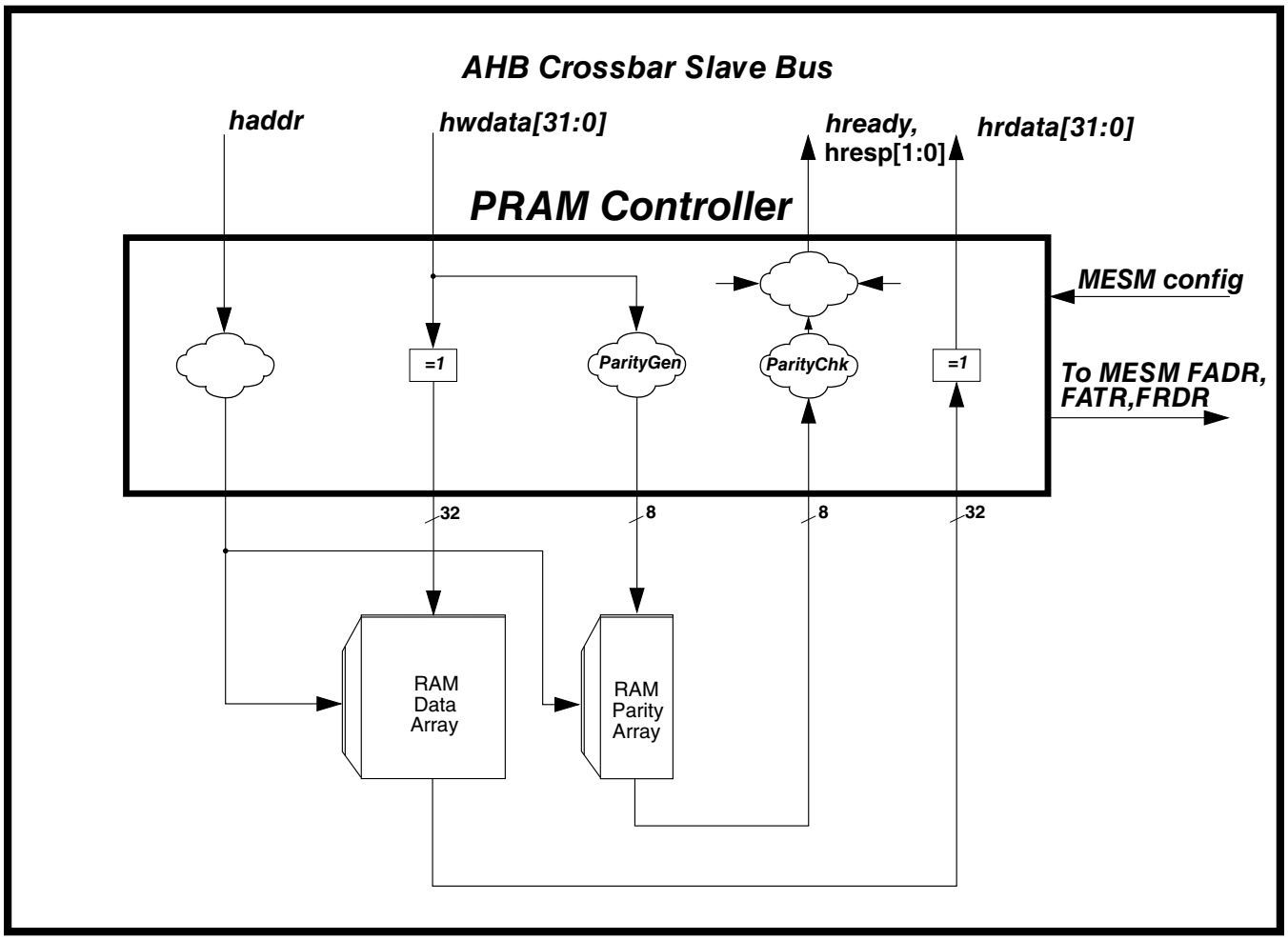


Figure 30-2. Simplified Platform RAM Controller with Parity Block Diagram

30.1.2 Features

The key features of the MESM include:

- Read-only static core platform configuration information
- Programmable configuration for enabling RAM single bit error detection
- Programmable support for error reporting: bus error terminations and/or alert interrupt
- Read-only fault status registers containing address, attributes and data from faulting RAM read
- Programmable support for software insertion of single bit errors to "check the checkers"
- Memory-mapped device connected to a 32-bit slave peripheral (IPS) bus

- All programming model references must be supervisor mode

30.1.3 Modes of Operation

The MESM module does not support any special modes of operation. Its operation is primarily defined by EDC events detected by the RAM controller which are configured by MESM register bits. Additionally, as a memory-mapped device located on the platform's slave peripheral bus, it responds based strictly on memory addresses for accesses to its programming model and only allows supervisor mode references.

The MESM module resides in the *platform's clock domain;* this includes its interconnection with the RAM controller and the slave peripheral bus.

30.2 External Signal Description

The MESM module does not directly support any external interfaces.

30.3 Memory Map and Register Definition

The MESM module supports a compact 32-byte software programming model, split into three sections: read-only platform configuration, the EDC control and status registers, and the captured fault registers.

The programming model can only be referenced with supervisor mode accesses; attempted references in user mode are terminated with an error as are attempted writes to read-only registers. Additionally, the supervisor mode references can be made with any operand size (byte, word, longword), although performance is typically maximized by referencing the registers using their natural size.

MESM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_E800	MESM Platform Configuration Register 0 (MESM_PLTCFG0)	32	R	CF10_0000h	30.3.1/562
FFFF_E804	MESM Platform Configuration Register 1 (MESM_PLTCFG1)	32	R	F000_D000h	30.3.2/563
FFFF_E808	MESM Error Detecting Code Control Register (MESM_EDCCR)	32	R/W	0000_0000h	30.3.3/564

Table continues on the next page...

MESM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_E80C	MESM Error Detecting Code Status Register (MESM_EDCSR)	32	R/W	0000_0000h	30.3.4/566
FFFF_E810	MESM Parity Error Generation Register (MESM_PEGR)	32	R/W	Undefined	30.3.5/567
FFFF_E814	MESM Fault Address Register (MESM_FADR)	32	R	Undefined	30.3.6/569
FFFF_E818	MESM Fault Attribute Register (MESM_FATR)	32	R	Undefined	30.3.7/570
FFFF_E81C	MESM Fault Read Data Register (MESM_FRDR)	32	R	Undefined	30.3.8/572

30.3.1 MESM Platform Configuration Register 0 (MESM_PLTCFG0)

This register contains two 16-bit read-only status fields providing information on the processor core type and the platform hardware revision. Attempted writes to this location are terminated with an error.

Address: FFFF_E800h base + 0h offset = FFFF_E800h

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	PCT																PLREV															
W																																
Reset	1	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MESM_PLTCFG0 field descriptions

Field	Description
31–16 PCT	Processor Core Type. This field provides a read-only value defining the processor as a Version 1 ColdFire core: 0xCF10.
15–0 PLREV	Platform Revision. This read-only field is specified by a platform input signal to define a software-visible revision number.

30.3.2 MESM Platform Configuration Register 1 (MESM_PLTCFG1)

This register contains two 16-bit read-only status fields providing information on the AHB crossbar switch (AXBS) configuration. The AMC is a 16-bit read-only field identifying the presence/absence of crossbar bus master connectionls while the ASC is a 16-bit read-only value identifying the presence/absence of crossbar slave connections. Attempted writes to this location are terminated with an error.

Address: FFFF_E800h base + 4h offset = FFFF_E804h

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	AMC[0:7]								0								ASC[0:7]								0							
W																																
Reset	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0

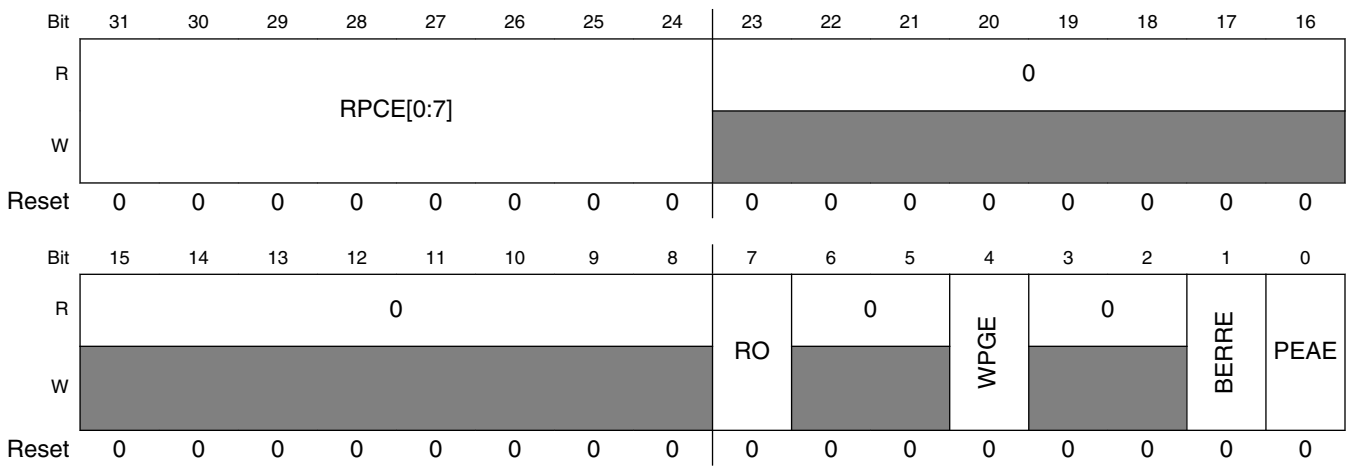
MESM_PLTCFG1 field descriptions

Field	Description
31–24 AMC[0:7]	AXBS Master Configuration These bits represent the bit map defining the presence of bus masters connected to the crossbar switch. For this device, the AMC field is 0xFO. AMC[n] = 0 If a bus master connection to AXBS input port 'n' is absent. AMC[n] = 1 If a bus master connection to AXBS input port 'n' is present.
23–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–8 ASC[0:7]	AXBS Slave Configuration These bits represent the bit map defining the presence of bus slaves connected to the crossbar switch. For this device, the ASC field is 0xD0. ASC[n] = 0 If a bus slave connection to AXBS output port 'n' is absent. ASC[n] = 1 If a bus slave connection to AXBS output port 'n' is present.
7–0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

30.3.3 MESM Error Detecting Code Control Register (MESM_EDCCR)

This register defines the configuration of the device's RAM error detecting code. It includes configuration fields enabling parity checking on reads, parity generation on writes, the response to the detection of a properly-enabled read parity error, and so on. If EDCCR[RO] = 1, this register can only be read and attempted writes are terminated with an error.

Address: FFFF_E800h base + 8h offset = FFFF_E808h



MESM_EDCCR field descriptions

Field	Description
31–24 RPCE[0:7]	Read Parity Check Enable n. This field provides enable bits for the RAM read parity check on each 8 KB address space. If the bit is cleared, parity checking on reads to the selected RAM address space is disabled, else it is enabled. RPCEn = 0 Disables the parity checking on reads in the selected RAM address space RPCE0 = 1 Enables parity checking on reads in RAM space 0x0080_0000 - 0x0080_1FFF RPCE1 = 1 Enables parity checking on reads in RAM space 0x0080_2000 - 0x0080_3FFF RPCE2 = 1 Enables parity checking on reads in RAM space 0x0080_4000 - 0x0080_5FFF RPCE3 = 1 Enables parity checking on reads in RAM space 0x0080_6000 - 0x0080_7FFF RPCE4 = 1 Enables parity checking on reads in RAM space 0x0080_8000 - 0x0080_9FFF RPCE5 = 1 Enables parity checking on reads in RAM space 0x0080_A000 - 0x0080_BFFF RPCE6 = 1 Enables parity checking on reads in RAM space 0x0080_C000 - 0x0080_5DFF RPCE7 = 1 Enables parity checking on reads in RAM space 0x0080_E000 - 0x0080_FFFF
23–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 RO	Read Only. This bit is intended to prevent accidental writes of the EDCCR from changing the defined RAM parity configuration.

Table continues on the next page...

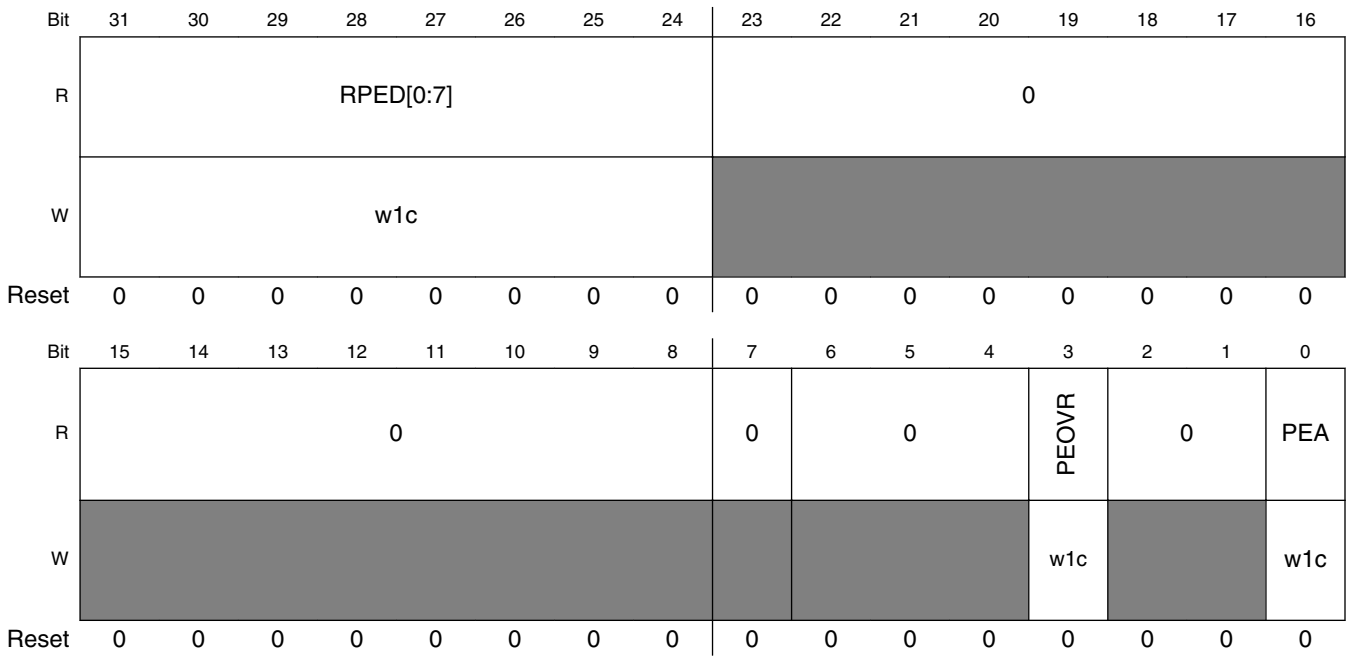
MESM_EDCCR field descriptions (continued)

Field	Description
	0 EDCCR can be read or written 1 EDCCR can only be read. A system reset is required before this register can again be written.
6–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 WPGE	Write Parity Generation Enable. This bit provides a global enable for parity to be generated on all writes to the RAM. Each 8 KB section of the RAM must be initialized by data writes while WPGE = 1 before the corresponding read parity check enable (RPCEn) can be safely set. 0 Parity is not generated and stored in memory on RAM write. 1 Parity is generated and stored in memory on every RAM write.
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 BERRE	Bus Error Enable. This bit partially determines the response in the event of a properly-enabled read parity error is detected. The EDCCR[BERRE] and EDCCR[PEAE] bits allow system software to configure the appropriate response when a properly-enabled read parity error is detected. It is important to note that the default response of the CF1Core to a bus error response is a system reset. If this MESM bus error response is enabled, it is recommended that the core's CPUCR[ARD] = 1 so that the processor generates an access error exception instead of the reset on any reported read parity errors. 0 If a properly-enabled read parity error is detected, the RAM read is not terminated with a bus error response. 1 If a properly-enabled read parity error is detected, the RAM read is terminated with a bus error response.
0 PEAE	Parity Error Alert Enable. This bit partially determines the response in the event of a properly-enabled read parity error is detected. The EDCCR[BERRE] and EDCCR[PEAE] bits allow system software to configure the appropriate response when a properly-enabled read parity error is detected. 0 If a properly-enabled read parity error is detected, an MESM parity error alert interrupt request is not asserted. 1 If a properly-enabled read parity error is detected, an MESM parity error alert interrupt request is asserted.

30.3.4 MESM Error Detecting Code Status Register (MESM_EDCSR)

This register contains the status of the device's RAM error detecting code. It includes bit fields reporting on detected read parity errors, the EDC alert interrupt, and so on.

Address: FFFF_E800h base + Ch offset = FFFF_E80Ch



MESM_EDCSR field descriptions

Field	Description
31–24 RPED[0:7]	Read Parity Error Detected n. This field provides status bits signaling a properly-enabled RAM read parity error has been detected in the corresponding 8 KB address space. The bits in this field are set by the hardware and the entire field cleared whenever system software writes 1 to the PEA bit. RPEDn = 0 indicates a read parity error has not been detected in the corresponding RAM address space. RPED0 = 1 indicates a parity error in the RAM space 0x0080_0000 - 0x0080_1FFF RPED1 = 1 indicates a parity error in the RAM space 0x0080_2000 - 0x0080_3FFF RPED2 = 1 indicates a parity error in the RAM space 0x0080_4000 - 0x0080_5FFF RPED3 = 1 indicates a parity error in the RAM space 0x0080_6000 - 0x0080_7FFF RPED4 = 1 indicates a parity error in the RAM space 0x0080_8000 - 0x0080_9FFF RPED5 = 1 indicates a parity error in the RAM space 0x0080_A000 - 0x0080_BFFF RPED6 = 1 indicates a parity error in the RAM space 0x0080_C000 - 0x0080_5DFF RPED7 = 1 indicates a parity error in the RAM space 0x0080_E000 - 0x0080_FFFF
23–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MESM_EDCSR field descriptions (continued)

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 PEOVR	Parity Error Overrun. The MESM logic captures a properly-enabled EDC event in the RAM. If another parity error is detected before system software has retrieved all the error information from the original event, this overrun flag is set. The window of time is defined from the detection of the original EDC event until the EDCSR[PEA] is written with 1 to clear it and rearm the capture logic. This bit is set by the hardware and cleared whenever system software writes 1 to the PEA bit. 0 No detection of a subsequent read parity error has occurred. 1 A subsequent read parity error has been detected before the contents of the original error were read from the MESM.
2–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 PEA	Parity Error Alert. This bit signals a properly-enabled RAM read parity error has been detected. The location of the parity error is signaled in the EDCSR[RPEDn] field. The assertion of this bit may coincide with the reporting of a bus error on the RAM read (if EDCCR[BERRE] = 1) and/or the assertion of an interrupt (if EDCCR[PEAE] = 1). The alert is negated by writing 1 to this bit. Writing 0 has no effect. Additionally, writing 1 to this bit clears all the other fields within the EDCSR register and rearms the error capture logic. Accordingly, the contents of the capture registers should be read before this flag is cleared. 0 RAM read parity error has not been detected. 1 RAM read parity error has been detected. The contents of the FADR, FATR, and FRDR registers are valid.

30.3.5 MESM Parity Error Generation Register (MESM_PEGR)

The Parity Error Generation Register is a 32-bit control register used to force the generation of single bit data inversions in the RAM with nibble parity. This capability is provided for two purposes:

- It provides a software-controlled mechanism for 'injecting' errors into the RAM during data writes to verify the integrity of the parity check logic.
- It provides a mechanism to allow testing of the software service routines associated with memory error logging.

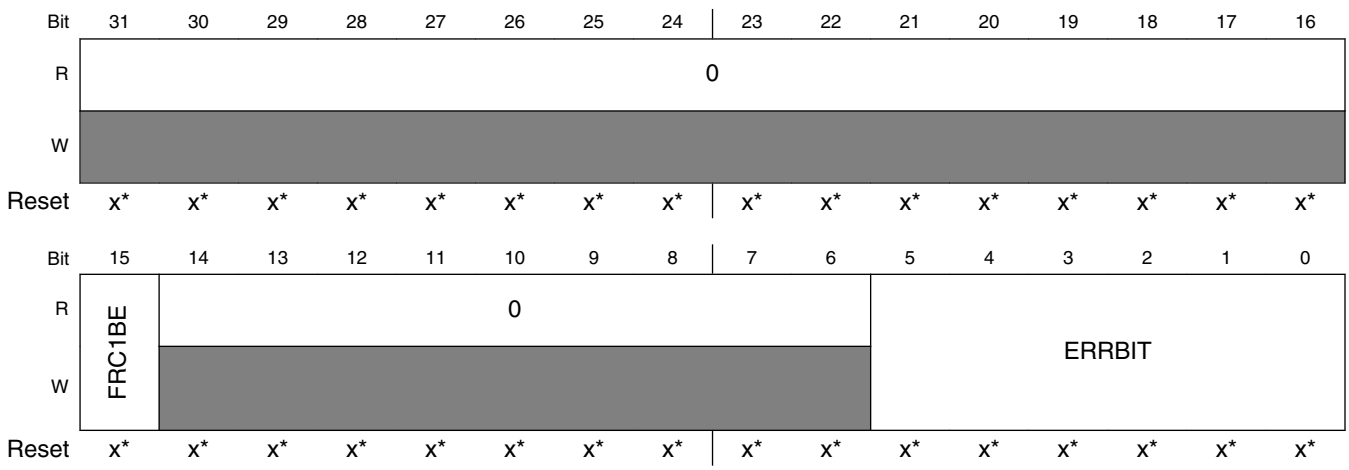
The intent is to generate parity errors during RAM data write cycles such that subsequent reads of the corrupted address locations generate parity events that are terminated with an error response and/or assert an alert interrupt request.

This register can only be written as a 32-bit quantity with a data value of 0x0000_{0,8}0{ERRBIT less than 40}. This provides some protection against accidental updates to this register. Any attempted write with a smaller reference size (byte, word) or with an “illegal” ERRBIT value is terminated with an error and the register unchanged.

To minimize any unintended system implications, the FRC1BE control bit can be cleared by software or in response to system events. Specifically, the FRC1BE bit can be cleared through the following methods:

1. System software writes the PEGR with a logical zero for data bit 15;
2. The processor begins any type of exception processing - this includes any interrupt exception, bus error, or any type of internal fault;
3. MESM detects a read parity error.

Address: FFFF_E800h base + 10h offset = FFFF_E810h



* Notes:
- x = Undefined at reset.

MESM_PEGR field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 FRC1BE	Force 1-bit Error. When this flag is set and EDCCR[WPGE] = 1, the write data value has one bit inverted to introduce a parity error. The data bit to be inverted is specified by ERRBIT[5:0]. If the RAM write size is byte or word, then the error is only introduced if the ERRBIT is contained within the memory byte(s) being updated. The data bit inversion is performed on every RAM write while FRC1BE = 1 and EDCCR[WPGE] = 1. This bit can be cleared by 3 mechanisms: a software write to the bit with a logical zero data value, processor initiation of any type of exception processing, and MESM detection of a read parity error.
14–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MESM_PEGR field descriptions (continued)

Field	Description
5–0 ERRBIT	Error Bit. This vector defines the bit position which is complemented to create the data inversion on the RAM write operation. The RAM controller follows a vector bit ordering scheme where LSB=0. Errors in the parity check bits can be generated by setting this field to a value greater than the RAM width. The association between the ERRBIT field and the corrupted memory bit is shown below. ERRBIT values greater than 39 are illegal and attempted writes with these values are terminated with an error and the register unchanged. **NOTE:** PCHECK[7:0] are the parity check bits. if ERRBIT = 0, then RAM[0] is inverted if ERRBIT = 1, then RAM[1] is inverted .... if ERRBIT = 31, then RAM[31] is inverted if ERRBIT = 32, then PCHECK[0] covering RAM[3: 0] is inverted if ERRBIT = 33, then PCHECK[1] covering RAM[7: 4] is inverted if ERRBIT = 34, then PCHECK[2] covering RAM[11: 8] is inverted if ERRBIT = 35, then PCHECK[3] covering RAM[15:12] is inverted if ERRBIT = 36, then PCHECK[4] covering RAM[19:16] is inverted if ERRBIT = 37, then PCHECK[5] covering RAM[23:20] is inverted if ERRBIT = 38, then PCHECK[6] covering RAM[27:24] is inverted if ERRBIT = 39, then PCHECK[7] covering RAM[31:28] is inverted

30.3.6 MESM Fault Address Register (MESM_FADR)

When a properly-enabled RAM read parity error is detected, the faulting address is captured in the read-only FADR register. The MESM logic supports capturing a single parity error event; if a subsequent error is detected before the captured error information has been read from the corresponding registers, the EDCSR[PEOVR] flag is set but no additional information is captured. The bits in this register are set by the hardware and signaled by the assertion of EDCSR[PEA].

Address: FFFF_E800h base + 14h offset = FFFF_E814h

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	FLTADDR																															
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.

MESM_FADR field descriptions

Field	Description
31–0 FLTADDR	Fault Address. This field defines the access address when a properly-enabled RAM read parity error is detected. Since parity checking is only supported on the device's RAM, the field is always in the form 0x(00)80_xxxx.

30.3.7 MESM Fault Attribute Register (MESM_FATR)

When a properly-enabled RAM read parity error is detected, the faulting attributes are captured in the read-only FATR register. The MESM logic supports capturing a single parity error event; if a subsequent error is detected before the captured error information has been read from the corresponding registers, the EDCSR[PEOVR] flag is set but no additional information is captured.

This register includes multiple fields: the bus master number and access attributes from the faulting RAM reference, the error syndrome defining the data nibble(s) in error and the parity check bits read from memory. The bits in this register are set by the hardware and signaled by the assertion of EDCSR[PEA].

Address: FFFF_E800h base + 18h offset = FFFF_E818h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PESYND								PCHECK							
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	0				PEMN				PEWT	0	PESZ		0		PESM	PEDA
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- x = Undefined at reset.

MESM_FATR field descriptions

Field	Description
31–24 PESYND	Parity Error Syndrome. This field signals the detection of a properly-enabled RAM read parity error(s). There is a flag for each of the 8 nibble parity checks. If multiple parity errors in a single RAM read are detected, multiple bits in this field are set. PESYNDn = 0 signals a read parity error has not been detected in the RAM nibble PESYND7 = 1 signals a read parity error was detected in RAM[31:28] PESYND6 = 1 signals a read parity error was detected in RAM[27:24] PESYND5 = 1 signals a read parity error was detected in RAM[23:20] PESYND4 = 1 signals a read parity error was detected in RAM[19:16] PESYND3 = 1 signals a read parity error was detected in RAM[15:12] PESYND2 = 1 signals a read parity error was detected in RAM[11:8] PESYND1 = 1 signals a read parity error was detected in RAM[7:4] PESYND0 = 1 signals a read parity error was detected in RAM[3:0]
23–16 PCHECK	Parity Check Bits. These are the “raw” nibble parity check bits read from the RAM on the faulting access. PCHECK7 is the even parity check bit associated with RAM[31:28] PCHECK6 is the odd parity check bit associated with RAM[27:24] PCHECK5 is the even parity check bit associated with RAM[23:20] PCHECK4 is the odd parity check bit associated with RAM[19:16] PCHECK3 is the even parity check bit associated with RAM[15:12] PCHECK2 is the odd parity check bit associated with RAM[11: 8] PCHECK1 is the even parity check bit associated with RAM[7: 4] PCHECK0 is the odd parity check bit associated with RAM[3: 0]
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–8 PEMN	Parity Error Master Number This 4-bit field contains the crossbar bus master number of the captured RAM read parity error.
7 PEWT	Parity Error Write. This bit signals the type of RAM access when the parity error was detected. Since the parity check is only performed on RAM reads, this bit is always a logical zero.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–4 PESZ	Parity Error Size. This field signals the size of the RAM access when the parity error was detected. 00 8-bit access 01 16-bit access 10 32-bit access 11 Reserved

Table continues on the next page...

MESM_FATR field descriptions (continued)

Field	Description
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 PESM	Parity Error Supervisor Mode. This flag signals the mode of the RAM access when the parity error was detected. 0 User mode 1 Supervisor mode
0 PEDA	Parity Error Data Access. This flag signals the type of RAM access when the parity error was detected. 0 Instruction Fetch 1 Data

30.3.8 MESM Fault Read Data Register (MESM_FRDR)

When a properly-enabled RAM read parity error is detected, the faulting data is captured in the read-only FRDR register. The MESM logic supports capturing a single parity error event; if a subsequent error is detected before the captured error information has been read from the corresponding registers, the EDCSR[PEOVR] flag is set but no additional information is captured. The bits in this register are set by the hardware and signaled by the assertion of EDCSR[PEA].

Address: FFFF_E800h base + 1Ch offset = FFFF_E81Ch

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	FLTDATA																															
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.

MESM_FRDR field descriptions

Field	Description
31–0 FLTDATA	Fault Data. This field defines the access data when a properly-enabled RAM read parity error is detected. The “raw” parity check bits associated with the read parity error are captured in FATR[PCHECK].

30.4 Functional Description

This section provides details on basic functionality associated with the RAM parity logic and the relative timing of RAM reads which detect single bit errors.

30.4.1 Nibble Parity Generation and Checking

The RAM controller implements a nibble-wide parity check code, that is, there is a single parity bit for every 4 bits of data. An alternating scheme of even and odd parity bits is implemented to provide improved detection capabilities against certain classes of RAM failures. The terms even and odd parity are defined such that the total number of "1" bits in the concatenation of the data field and associated parity bit are even or odd. For this device, the parity check is defined as follows, where PCHECK[7:0] is the parity check bits associated with the 32-bit RAM:

```
PCHECK[7] is the even parity check bit associated with RAM[31:28]
PCHECK[6] is the  odd parity check bit associated with RAM[27:24]
PCHECK[5] is the even parity check bit associated with RAM[23:20]
PCHECK[4] is the  odd parity check bit associated with RAM[19:16]
PCHECK[3] is the even parity check bit associated with RAM[15:12]
PCHECK[2] is the  odd parity check bit associated with RAM[11: 8]
PCHECK[1] is the even parity check bit associated with RAM[ 7: 4]
PCHECK[0] is the  odd parity check bit associated with RAM[ 3: 0]
```

Consider the following RTL description of the Boolean equations related to the nibble parity generation and check functions. In the following description, the signal names represent:

```
wdata[31:0]                // write data to ram
      wpcheck[7:0]              // write data parity check bits to ram
      rdata[31:0]               // data read from ram
      rpcheck[7:0]              // parity check bits read from ram
      parity_err_syndrome[7:0]  // nibble parity check results
      read_parity_error         // logical summation of nibble parity check results
//*************************************************************************
// nibble parity generation on ram write data
// the  odd pcheck[*] bit numbers implement even parity
// the even pcheck[*] bit numbers implement  odd parity
wpcheck[7]      = (wdata[31]    ^  wdata[30]    ^  wdata[29]    ^  wdata[28]),
wpcheck[6]      =~(wdata[27]    ^  wdata[26]    ^  wdata[25]    ^  wdata[24]),
wpcheck[5]      = (wdata[23]    ^  wdata[22]    ^  wdata[21]    ^  wdata[20]),
wpcheck[4]      =~(wdata[19]    ^  wdata[18]    ^  wdata[17]    ^  wdata[16]),
wpcheck[3]      = (wdata[15]    ^  wdata[14]    ^  wdata[13]    ^  wdata[12]),
wpcheck[2]      =~(wdata[11]    ^  wdata[10]    ^  wdata[9]     ^  wdata[8]),
wpcheck[1]      = (wdata[7]     ^  wdata[6]     ^  wdata[5]     ^  wdata[4]),
wpcheck[0]      =~(wdata[3]     ^  wdata[2]     ^  wdata[1]     ^  wdata[0]),
//*************************************************************************
// nibble parity check on ram read data
parity_err_syndrome[7]
    = (rdata[31]  ^  rdata[30]  ^  rdata[29]    ^  rdata[28]    ^  rpcheck[7]),
parity_err_syndrome[6]
    =~(rdata[27]  ^  rdata[26]  ^  rdata[25]    ^  rdata[24]    ^  rpcheck[6])
parity_err_syndrome[5]
```

```
    = (rdata[23]  ^  rdata[22]  ^  rdata[21]    ^  rdata[20]    ^  rpcheck[5]),
parity_err_syndrome[4]
    =~(rdata[19]  ^  rdata[18]  ^  rdata[17]    ^  rdata[16]    ^  rpcheck[4]),
parity_err_syndrome[3]
    = (rdata[15]  ^  rdata[14]  ^  rdata[13]    ^  rdata[12]    ^  rpcheck[3]),
parity_err_syndrome[2]
    =~(rdata[11]  ^  rdata[10]  ^  rdata[9]     ^  rdata[8]     ^  rpcheck[2])
parity_err_syndrome[1]
    = (rdata[7]   ^  rdata[6]   ^  rdata[5]     ^  rdata[4]     ^  rpcheck[1]),
parity_err_syndrome[0]
    =~(rdata[3]   ^  rdata[2]   ^  rdata[1]     ^  rdata[0]     ^  rpcheck[0]),
read_parity_error
    = | parity_err_syndrome[7:0];              // logically sum all bits
```

30.4.2 RAM Parity Initialization

As previously noted, the contents of the RAM must be written while EDCCR[WPGE] = 1 to load the memory with correct parity before reads are checked.

This memory initialization function is typically performed by the processor or the DMA.

If the processor performs the initialization, the use of the MOVEM store instruction is recommended for maximum performance and minimum execution time. The typical construct may include a simple loop with multiple MOVEM instructions, each storing 4 or 8 registers to the RAM.

An alternative approach can make use of a DMA channel to copy a "constant" memory value into the RAM region needing initialization. This approach offers the advantage where the initialization time may be hidden based on simultaneous execution of the processor and the DMA.

In both cases, it is important that the RAM region be first written before parity checking on reads is enabled to prevent "spurious" fault detection.

30.4.3 RAM Read Bus Timings

This section presents two timing diagrams showing a series of three RAM read accesses (x, y, z) where the middle read (y) detects a parity error. In particular, one diagram shows a bus error response (enabled with EDCCR[BERRE] = 1) while the other shows the assertion of the parity error alert interrupt (enabled with EDCCR[PEAE] = 1).

Recall the *default* response of the CF1Core to a bus cycle terminated with an error is a system reset. If the MESM bus error response is enabled (EDCCR[BERRE] = 1), it is recommended that the core's CPUCR[ARD] = 1 so that the processor generates an access error exception instead of the reset on any reported read parity errors.

In both diagrams, the 2-stage pipelined AHB bus protocol is shown (address_phase, data_phase) along with the termination control signals (hready, hresp). Additionally, the behavior of the MESM parity error signal and fault capture registers is also shown.

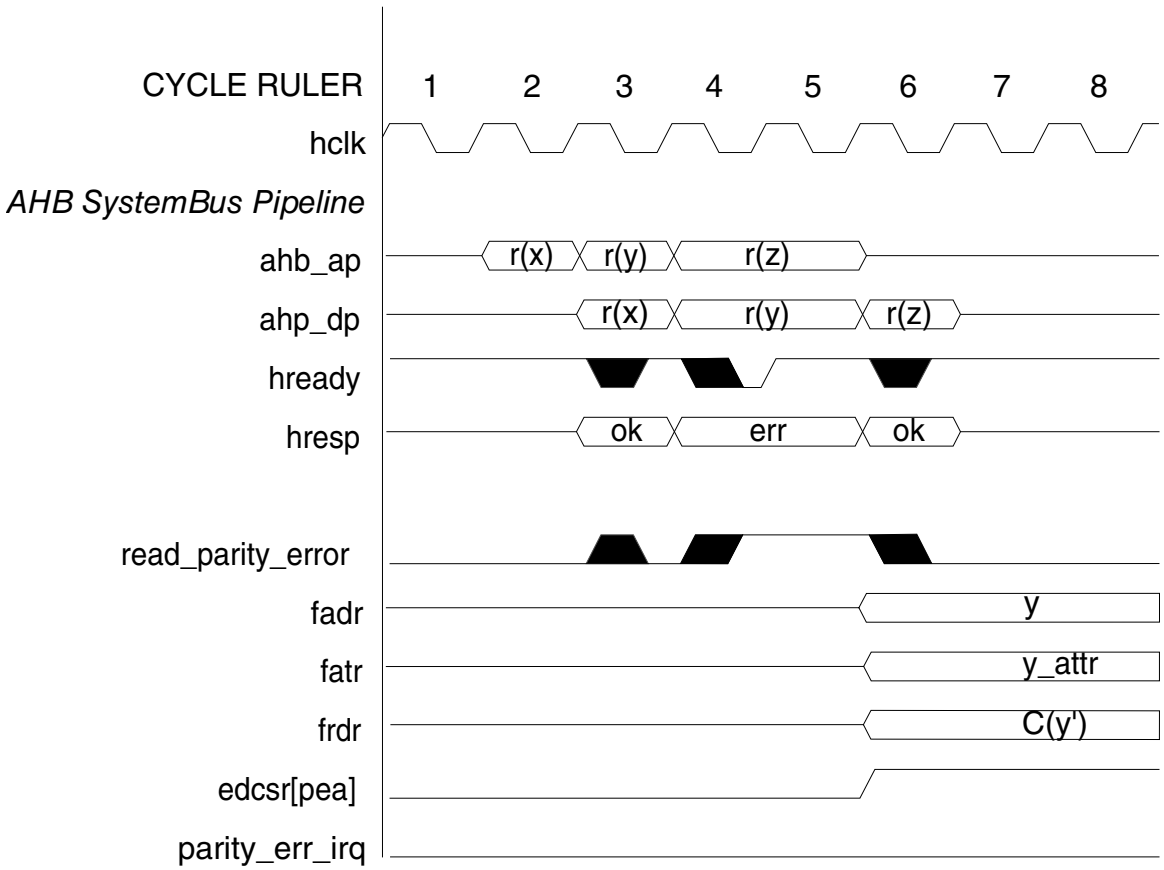


Figure 30-11. RAM Read Parity Error with Bus Error Termination

The preceding figure shows three consecutive RAM read accesses where the middle reference r(y) detects a parity error. During the AHB data phase (ahb_dp), the RAM reads the data and parity check bits which are then examined for single bit errors. This example shows operation when parity errors are configured to generate an error termination on the faulted bus cycle. The AHB data phase for the read of "y" occurs in cycles 4 and 5. During cycle 4, the RAM read parity error is initially detected - this in turn begins the standard 2-cycle AHB error termination as shown by hresp = ERR during these cycles. Once the data phase for the faulted r(y) completes, the address, attributes and read data are captured in the appropriate MESM registers (FADR, FATR, FRDR) and the parity error alert bit in the EDCSR register asserted. Note the FRDR and FATR[PCHECK] capture the faulted read operand {C(y')} and associated parity check bits.

Next, consider the behavior when a parity error is configured to generate the alert interrupt.

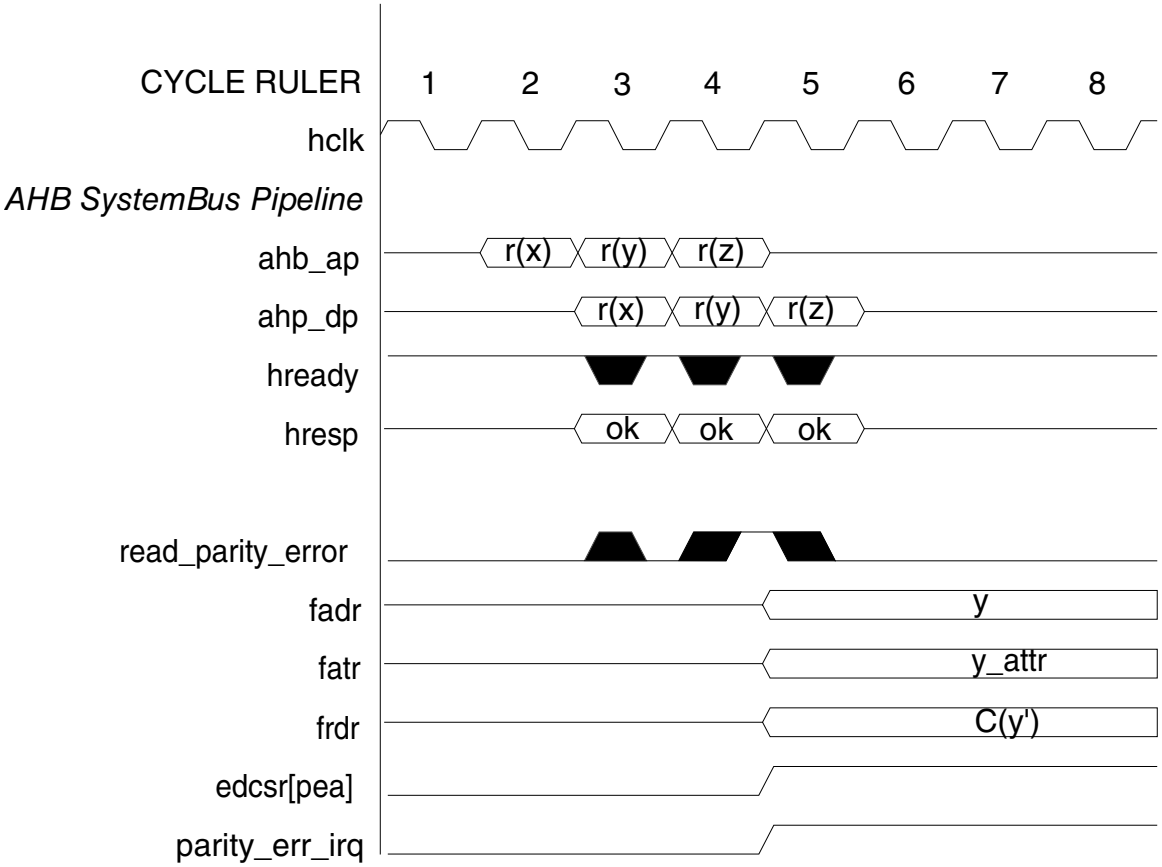


Figure 30-12. RAM Read Parity Error with Alert Interrupt Assertion

Again, a sequence of 3 RAM reads is shown with r(y) returning corrupted data. The AHB data phase for this read occurs in cycle 4. During this time, the read parity error is detected, but the system bus cycle completes without an error since EDCCR[BERRE] = 0. Instead, the alert interrupt request (parity_err_irq) is asserted to signal the detection of the RAM read error. As before, once the data phase for the faulted r(y) completes in cycle 4, the address, attributes and read data are captured in the appropriate MESM registers (FADR, FATR, FRDR) and the parity error alert bit in the EDCSR register asserted.

30.4.4 Error Handler Routines

MESM supports two mechanisms for reporting of RAM read parity errors: the offending bus cycle may be terminated with an error, that is, a "bus error" and/or a maskable parity error alert interrupt request can be generated.

As previously noted, the default response of the CF1Core to a bus cycle terminated with an error is a system reset. If the MESM bus error response is enabled (EDCCR[BERRE] = 1), it is recommended that the core's CPUCR[ARD] = 1 so that the processor generates an access error exception instead of the reset on any reported read parity errors. Additionally, it is important to note that the use of the bus error termination is dependent

on the response for the appropriate bus master. If the processor generated the RAM read that detected the parity error, a bus error exception (also known as an access error exception) is generated. For the V1 ColdFire core, these types of bus errors on memory reads are generally recoverable, although the exception handler may be required to perform address register adjustments if the fault occurred on an operand read using post-increment (Ay)+ or predecrement -(Ay) addressing modes. Additionally, if the error occurred on a MOVEM load instruction, recovery is possible only if the register list to be loaded does ***not*** include the base address register used in the effective address calculation. Bus errors on instruction fetches are always recoverable.

The exact response to bus errors by other masters, for example, the DMA, are generally module-specific. See the appropriate module documentation for details.

If the parity error alert interrupt response is enabled, the asserted IRQ is typically mapped as a level-6 maskable request. Accordingly, the interrupt is recognized at the next instruction boundary (ColdFire cores sample for interrupts once per instruction) and exception processing initiated.

The corresponding software handler (either the bus error handler or the interrupt service routine) would typically access the FADR, FATR and FRDR registers and save this information as part of the error logging process. Once the details on the fault are recorded, the EDCSR[PEA] bit is cleared (by writing a logical one to it) - this action clears all the bits in the EDCSR and rearms the registers which capture the fault information.

Chapter 31
Modulo Timer (MTIM)

31.1 Introduction

The MTIM (or MTIM16) module is a simple 16-bit timer with several software selectable clock sources and a programmable interrupt.

31.2 Features

Timer system features include:

- 16-bit up-counter
  - Free-running or 16-bit modulo limit
  - Software controllable interrupt on overflow
  - Counter reset bit (TRST)
  - Counter stop bit (TSTP)
- Four software selectable clock sources for input to prescaler:
  - System bus clock — rising edge
  - Fixed frequency clock (XCLK) — rising edge
  - External clock source on the TCLK pin — rising edge
  - External clock source on the TCLK pin — falling edge
- Nine selectable clock prescale values:
  - Clock source divide by 1, 2, 4, 8, 16, 32, 64, 128, or 256
- Modulo compare matched can be an output

31.2.1 Block Diagram

The following figure is a block diagram of the modulo timer module.

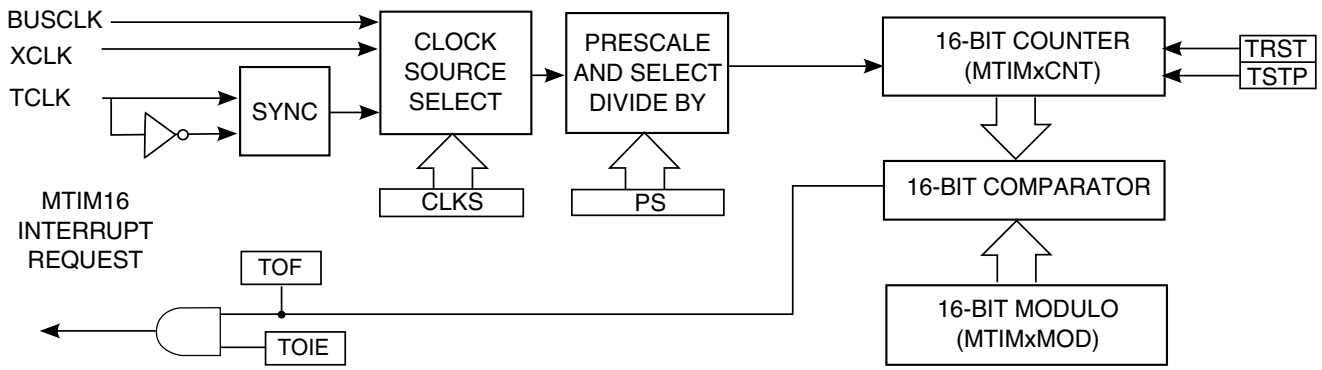


Figure 31-1. Modulo Timer (MTIM16) Block Diagram

31.2.2 Modes of Operation

This section defines MTIM16 operation in stop, wait, and background debug modes.

31.2.2.1 MTIM16 in Wait Mode

The MTIM16 continues to run in wait mode if enabled prior to the execution of the WAIT instruction. The timer overflow interrupt brings the MCU out of wait mode if it is enabled. For lowest possible current consumption, the MTIM16 should be stopped by software if it is not needed as an interrupt source during wait mode.

31.2.2.2 MTIM16 in Stop Modes

MTIM operation in stop modes is chip-specific. See details about MCU power modes and clocking.

- If the MTIM is unclocked in any MCU stop mode, it is disabled in that mode, regardless of the module settings before the STOP instruction was executed. It cannot be used as a wakeup source in that mode.
- If the MTIM is clocked and enabled in an MCU stop mode, it can be used as a wakeup source in that mode.

Upon waking from very low-power stop modes, the MTIM enters its reset state.

For low-power stop modes:

- If the device exits any of these modes with a reset, the MTIM module enters its reset state.
- If the device exits any of these modes with an interrupt, the MTIM module continues from its state in the low-power stop mode.
- If the counter was active upon entering any of these modes, the count resumes from the current value.

31.2.2.3 MTIM16 in Active Background Mode

The MTIM16 stops all counting until the microcontroller returns to normal user operating mode. Counting resumes from the suspended value as long as an MTIM16 reset did not occur (TRST written to a 1).

31.3 External Signal Description

This section describes the module external signals.

31.3.1 TCLK — External Clock Source Input into MTIM16

The MTIM16 includes one external signal, TCLK, used to input an external clock when selected as the MTIM16 clock source. The signal properties of TCLK are shown in the following table.

Table 31-1. Signal Properties

Signal	Function	I/O
TCLK	External clock source input into MTIM16	I

The TCLK input must be synchronized by the bus clock. Also, variations in duty cycle and clock jitter must be accommodated. As a result, the TCLK signal must be limited to one-fourth of the bus frequency.

The TCLK pin can be muxed with a general-purpose port pin. Refer to the chip-level signal multiplexing and pin assignment details for more information.

31.4 Memory Map and Register Descriptions

Each MTIM16 module includes six registers.

If a chip has more than one MTIM16 module, register names include placeholder characters.

MTIM memory map

Address offset (hex)	Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	FFFF_81F0	MTIM16 status and control register (MTIM0_SC)	8	R/W	10h	31.4.1/582
1	FFFF_81F1	MTIM16 clock configuration register (MTIM0_CLK)	8	R/W	00h	31.4.2/583
2	FFFF_81F2	MTIM16 counter register high (MTIM0_CNTH)	8	R	00h	31.4.3/584
3	FFFF_81F3	MTIM16 counter register low (MTIM0_CNTL)	8	R	00h	31.4.4/585
4	FFFF_81F4	MTIM16 modulo register high (MTIM0_MODH)	8	R/W	00h	31.4.5/586
5	FFFF_81F5	MTIM16 modulo register low (MTIM0_MODL)	8	R/W	00h	31.4.6/587

31.4.1 MTIM16 status and control register (MTIMx_SC)

This register contains the overflow status flag and control bits. Use them to configure the interrupt enable, reset the counter, and stop the counter.

Address: FFFF_81F0h base + 0h offset = FFFF_81F0h

Bit	7	6	5	4	3	2	1	0
Read	TOF	TOIE	0	TSTP	0			
Write	0		TRST					
Reset	0	0	0	1	0	0	0	0

MTIM0_SC field descriptions

Field	Description
7 TOF	MTIM16 overflow flag This bit is set when the MTIM16 counter register overflows to 0x0000 after reaching the value in the MTIM16 modulo register. Clear TOF by reading the SC register while TOF is set and then by writing 0 to TOF. Writing 1 has no effect. TOF is also cleared when 1 is written to TRST.

Table continues on the next page...

MTIM0_SC field descriptions (continued)

Field	Description
	0 MTIM16 counter has not reached the overflow value in the MTIM16 modulo register. 1 MTIM16 counter has reached the overflow value in the MTIM16 modulo register.
6 TOIE	MTIM16 overflow interrupt enable This read/write bit enables MTIM16 overflow interrupts. If TOIE is set, then an interrupt is generated when TOF = 1. Reset clears TOIE. Do not set TOIE if TOF = 1; instead, clear TOF first, and then set TOIE. 0 TOF interrupts are disabled. Use software polling. 1 TOF interrupts are enabled.
5 TRST	MTIM16 counter reset When 1 is written to this write-only bit, the MTIM16 counter register resets to 0x0000 and TOF is cleared. Writing 1 to this bit also causes the modulo value to take effect at once. Reading this bit always returns 0. 0 No effect. MTIM16 counter remains in its current state. 1 MTIM16 counter is reset to 0x0000.
4 TSTP	MTIM16 counter stop When set, this read/write bit stops the MTIM16 counter at its current value. Counting resumes from the current value when TSTP is cleared. Reset sets TSTP to prevent the MTIM16 from counting. 0 MTIM16 counter is active. 1 MTIM16 counter is stopped.
3–0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.4.2 MTIM16 clock configuration register (MTIMx_CLK)

This register contains the clock select bits (CLKS) and the prescaler select bits (PS).

Address: FFFF_81F0h base + 1h offset = FFFF_81F1h

Bit	7	6	5	4	3	2	1	0
Read	0		CLKS		PS			
Write								
Reset	0	0	0	0	0	0	0	0

MTIM0_CLK field descriptions

Field	Description
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–4 CLKS	Clock source select These two read/write bits select one of four different clock sources as the input to the MTIM16 prescaler. Changing the clock source while the counter is active does not clear the counter. The count continues with the new clock source. Reset clears CLKS to 00.

Table continues on the next page...

MTIM0_CLK field descriptions (continued)

Field	Description
	00 Encoding 0. Bus clock (BUSCLK) 01 Encoding 1. Fixed-frequency clock (XCLK) 10 Encoding 3. External source (TCLK pin), falling edge 11 Encoding 4. External source (TCLK pin), rising edge
3–0 PS	Clock source prescaler These four read/write bits select one of nine outputs from the 8-bit prescaler. Changing the prescaler value while the counter is active does not clear the counter. The count continues with the new prescaler value. Reset clears PS to 0000. 0000 Encoding 0. MTIM16 clock source / 1 0001 Encoding 1. MTIM16 clock source / 2 0010 Encoding 2. MTIM16 clock source / 4 0011 Encoding 3. MTIM16 clock source / 8 0100 Encoding 4. MTIM16 clock source / 16 0101 Encoding 5. MTIM16 clock source / 32 0110 Encoding 6. MTIM16 clock source / 64 0111 Encoding 7. MTIM16 clock source / 128 1xxx Encoding 8+. MTIM16 clock source / 256

31.4.3 MTIM16 counter register high (MTIMx_CNTH)

This register is the read-only value of the high byte of the current MTIM16 16-bit counter.

When either the CNTH or CNTL register is read, the content of the two registers is latched into a buffer where they remain latched until the other register is read. This allows the coherent 16-bit value to be read in both big-endian and little-endian compile environments and ensures the 16-bit counter is unaffected by the read operation. The coherency mechanism is automatically restarted by an MCU reset or by setting the TRST bit of the SC register (whether BDM mode is active or not).

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when BDM became active, even if one or both halves of the counter register are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, the appropriate value from the other half of the 16-bit value is read after returning to normal execution. The value read from the CNTH and CNTL registers in BDM mode is the value of these registers and not the value of their read buffer.

Address: FFFF_81F0h base + 2h offset = FFFF_81F2h

Bit	7	6	5	4	3	2	1	0
Read	CNTH							
Write								
Reset	0	0	0	0	0	0	0	0

MTIM0_CNTH field descriptions

Field	Description
7–0 CNTH	MTIM16 count (high byte) These 8 read-only bits contain the current high byte value of the 16-bit counter. Writing has no effect on this register. Reset clears the register to 0x00.

31.4.4 MTIM16 counter register low (MTIMx_CNTL)

This register is the read-only value of the low byte of the current MTIM16 16-bit counter.

When either the CNTH or CNTL register is read, the content of the two registers is latched into a buffer where they remain latched until the other register is read. This allows the coherent 16-bit value to be read in both big-endian and little-endian compile environments and ensures the 16-bit counter is unaffected by the read operation. The coherency mechanism is automatically restarted by an MCU reset or by setting the TRST bit of the SC register (whether BDM mode is active or not).

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when BDM became active, even if one or both halves of the counter register are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, the appropriate value from the other half of the 16-bit value is read after returning to normal execution. The value read from the CNTH and CNTL registers in BDM mode is the value of these registers and not the value of their read buffer.

Address: FFFF_81F0h base + 3h offset = FFFF_81F3h

Bit	7	6	5	4	3	2	1	0
Read	CNTL							
Write								
Reset	0	0	0	0	0	0	0	0

MTIM0_CNTL field descriptions

Field	Description
7–0 CNTL	MTIM16 count (low byte)

MTIM0_CNTL field descriptions (continued)

Field	Description
	These 8 read-only bits contain the current low byte value of the 16-bit counter. Writing has no effect on this register. Reset clears the register to 0x00.

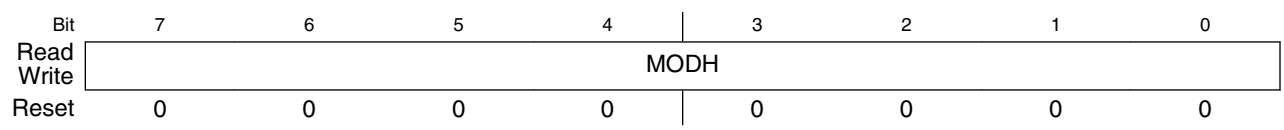
31.4.5 MTIM16 modulo register high (MTIMx_MODH)

A value of 0x0000 in MODH:MODL puts the MTIM16 in free-running mode. Writing to either MODH or MODL latches the value into a buffer and the latched buffers are updated after the second byte writing. The updated values take effect and reload to the MODH:MODL registers in the next MTIM16 counter cycle, except for the first writing of modulo after a chip reset or in BDM mode. However, after a software reset, the MODH:MODL takes effect at once even if it did not take effect before the reset. On the first writing of MODH:MODL after chip reset, the counter is reset and the modulo takes effect immediately. The latching mechanism may be manually reset by setting the TRST bit of the SC register (whether BDM is active or not).

When BDM is active, the coherency mechanism is frozen so that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the modulo register are written while BDM is active. Any writing to the modulo registers bypasses the buffer latches and writes directly to the modulo register while BDM is active, and also the counter is cleared at the same time.

Reading MODH:MODL returns the modulo values that are taking effect whenever in normal run mode or in BDM mode.

Address: FFFF_81F0h base + 4h offset = FFFF_81F4h



MTIM0_MODH field descriptions

Field	Description
7–0 MODH	MTIM16 modulo (high byte) These 8 read/write bits contain the modulo high byte value used to reset the counter and set TOF. Reset sets the register to 0x00.

31.4.6 MTIM16 modulo register low (MTIMx_MODL)

A value of 0x0000 in MODH:MODL puts the MTIM16 in free-running mode. Writing to either MODH or MODL latches the value into a buffer and the latched buffers are updated after the second byte writing. The updated values take effect and reload to the MODH:MODL registers in the next MITIM16 counter cycle, except for the first writing of modulo after a chip reset or in BDM mode. However, after a software reset, the MODH:MODL takes effect at once even if it did not take effect before the reset. On the first writing of MODH:MODL after chip reset, the counter is reset and the modulo takes effect immediately. The latching mechanism may be manually reset by setting the TRST bit of the SC register (whether BDM is active or not).

When BDM is active, the coherency mechanism is frozen so that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the modulo register are written while BDM is active. Any writing to the modulo registers bypasses the buffer latches and writes directly to the modulo register while BDM is active, and also the counter is cleared at the same time.

Reading MODH:MODL returns the modulo values that are taking effect whenever in normal run mode or in BDM mode.

Address: FFFF_81F0h base + 5h offset = FFFF_81F5h

Bit	7	6	5	4	3	2	1	0
Read Write	MODL							
Reset	0	0	0	0	0	0	0	0

MTIM0_MODL field descriptions

Field	Description
7–0 MODL	MTIM16 modulo (low byte) These 8 read/write bits contain the modulo low byte value used to reset the counter and set TOF. Reset sets the register to 0x00.

31.5 Functional Description

The MTIM16 is composed of a main 16-bit up-counter with 16-bit modulo register, a clock source selector, and a prescaler block with nine selectable values. The module also contains software selectable interrupt logic.

The MTIM16 counter (CNTH:CNTL registers) has three modes of operation: stopped, free-running, and modulo. The counter is stopped out of reset. If the counter starts without writing a new value to the modulo registers, it will be in free-running mode. The counter is in modulo mode when a value other than 0x0000 is in the modulo registers.

After an MCU reset, the counter stops and resets to 0x0000, and the modulo also resets to 0x0000. The bus clock functions as the default clock source, and the prescale value is divided by 1. To start the MTIM16 in free-running mode, write to the MTIM16 status and control (SC) register and clear the MTIM16 stop (TSTP) bit.

Clock sources are software selectable:

- the internal bus clock
- the fixed frequency clock (XCLK)
- an external clock on the TCLK pin that is selectable as incrementing on either rising or falling edges

The MTIM16 clock select (CLKS) field in the CLK register selects the desired clock source. If the counter is active (the TSTP bit is 0) when a new clock source is selected, the counter continues counting from the previous value using the new clock source.

Nine prescale values are software selectable: clock source divided by 1, 2, 4, 8, 16, 32, 64, 128, or 256. The prescaler select bits (PS[3:0]) in the CLK register select the desired prescale value. If the counter is active (TSTP = 0) when a new prescaler value is selected, the counter continues counting from the previous value using the new prescaler value.

The MTIM16 modulo register (MODH:MODL) allows the overflow compare value to be set to any value from 0x0001 to 0xFFFF. Reset clears the modulo value to 0x0000, which results in a free-running counter.

When the counter is active (the TSTP bit is 0), it increases at the selected rate until the count matches the modulo value. When these values match, the counter overflows to 0x0000 and continues counting. The MTIM16 overflow flag (TOF) is set whenever the counter overflows. The flag sets on the transition from the modulo value to 0x0000.

Clearing TOF is a two-step process. The first step is to read the SC register while TOF is set. The second step is to write a 0 to TOF. If another overflow occurs between the first and second steps, the clearing process is reset and TOF stays set after the second step is performed. This will prevent the second occurrence from being missed. TOF is also cleared when a 1 is written to TRST.

The MTIM16 module allows for an optional interrupt to be generated whenever TOF is set. To enable the MTIM16 overflow interrupt, set the MTIM16 overflow interrupt enable (TOIE) bit in the SC register. The TOIE bit should never be written to be 1 while TOF is 1. Instead, TOF should be cleared first, and then the TOIE bit can be set to 1.

31.5.1 MTIM16 Operation Example

This section shows an example of the MTIM16 module's operation as the counter reaches a matching value from the modulo register.

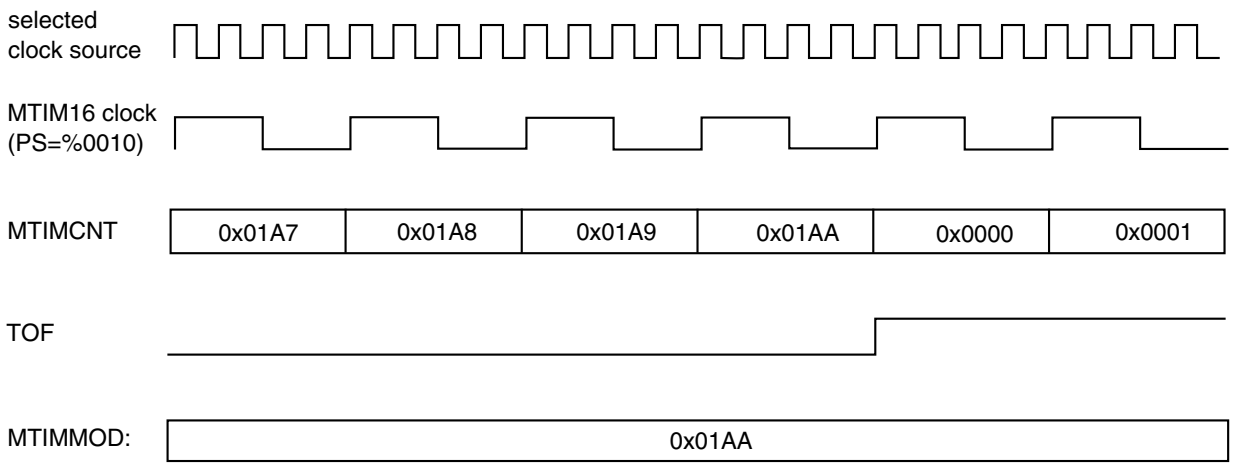


Figure 31-14. MTIM16 Counter Overflow Example

In this figure, the selected clock source could be any of the four possible choices. The prescaler is set to divide-by-4 (the PS field is 0010). The modulo value in the MODH:MODL register is set to 01AAh. When the counter (CNTH:CNTL registers) reaches the modulo value of 01AAh, the counter overflows to 0000h and continues counting. The timer overflow flag, TOF, sets when the counter value changes from 01AAh to 0000h. An MTIM16 overflow interrupt is generated when TOF is set, if the TOIE bit is 1.

Chapter 32
Low Power Timer (LPTMR)

32.1 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.

32.1.1 Features

The features of the LPTMR module include:

- 16-bit time counter or pulse counter with compare
  - Optional interrupt can generate asynchronous wakeup from any low-power mode
  - Hardware trigger output
  - Counter supports free-running mode or reset on compare
- Configurable clock source for prescaler/glitch filter
- Configurable input source for pulse counter
  - Rising-edge or falling-edge

32.1.2 Modes of operation

The following table describes the operation of the LPTMR module in various modes.

Table 32-1. 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.

Table continues on the next page...

Table 32-1. Modes of operation (continued)

Modes	Description
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.

32.2 LPTMR signal descriptions

Table 32-2. LPTMR signal descriptions

Signal	I/O	Description
LPTMR_ALT*n*	I	Pulse Counter Input pin

32.2.1 Detailed signal descriptions

Table 32-3. 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. 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.

32.3 Memory map and register definition

LPTMR memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_83A0	Low Power Timer Control Status Register (LPTMR0_CSR)	32	R/W	0000_0000h	32.3.1/593
FFFF_83A4	Low Power Timer Prescale Register (LPTMR0_PSR)	32	R/W	0000_0000h	32.3.2/594
FFFF_83A8	Low Power Timer Compare Register (LPTMR0_CMR)	32	R/W	0000_0000h	32.3.3/596
FFFF_83AC	Low Power Timer Counter Register (LPTMR0_CNR)	32	R	0000_0000h	32.3.4/596

32.3.1 Low Power Timer Control Status Register (LPTMRx_CSR)

Address: FFFF_83A0h base + 0h offset = FFFF_83A0h

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 configuration details for information on the connections to these inputs. 00 Pulse counter input 0 is selected.

Table continues on the next page...

LPTMRx_CSR field descriptions (continued)

Field	Description
	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.

32.3.2 Low Power Timer Prescale Register (LPTMRx_PSR)

Address: FFFF_83A0h base + 4h offset = FFFF_83A4h

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								PRESCALE					PBYP	PCS	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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. 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.
1–0 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.

Table continues on the next page...

LPTMRx_PSR field descriptions (continued)

Field	Description
	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.

32.3.3 Low Power Timer Compare Register (LPTMRx_CMR)

Address: FFFF_83A0h base + 8h offset = FFFF_83A8h

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.
15–0 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.

32.3.4 Low Power Timer Counter Register (LPTMRx_CNR)

Address: FFFF_83A0h base + Ch offset = FFFF_83ACh

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.
15–0 COUNTER	Counter Value

32.4 Functional description

32.4.1 LPTMR power and reset

The LPTMR remains powered in all power modes, including low-leakage modes. If the LPTMR is not required to remain operating during a low-power mode, then it must be disabled before entering the mode.

The LPTMR is reset only on global Power On Reset (POR) or Low Voltage Detect (LVD). When configuring the LPTMR registers, the CSR must be initially written with the timer disabled, before configuring the PSR and CMR. Then, CSR[TIE] must be set as the last step in the initialization. This ensures the LPTMR is configured correctly and the LPTMR counter is reset to zero following a warm reset.

32.4.2 LPTMR clocking

The LPTMR prescaler/glitch filter can be clocked by one of the four clocks. The clock source must be enabled before the LPTMR is enabled.

NOTE

The clock source selected may need to be configured to remain enabled in low-power modes, otherwise the LPTMR will not operate during low-power modes.

In Pulse Counter mode with the prescaler/glitch filter bypassed, the selected input source directly clocks the CNR and no other clock source is required. To minimize power in this case, configure the prescaler clock source for a clock that is not toggling.

NOTE

The clock source or pulse input source selected for the LPTMR should not exceed the frequency f_{LPTMR} defined in the device datasheet.

32.4.3 LPTMR prescaler/glitch filter

The LPTMR prescaler and glitch filter share the same logic which operates as a prescaler in Time Counter mode and as a glitch filter in Pulse Counter mode.

NOTE

The prescaler/glitch filter configuration must not be altered when the LPTMR is enabled.

32.4.3.1 Prescaler enabled

In Time Counter mode, when the prescaler is enabled, the output of the prescaler directly clocks the CNR. When the LPTMR is enabled, the CNR will increment every 2^2 to 2^{16} prescaler clock cycles. After the LPTMR is enabled, the first increment of the CNR will take an additional one or two prescaler clock cycles due to synchronization logic.

32.4.3.2 Prescaler bypassed

In Time Counter mode, when the prescaler is bypassed, the selected prescaler clock increments the CNR on every clock cycle. When the LPTMR is enabled, the first increment will take an additional one or two prescaler clock cycles due to synchronization logic.

32.4.3.3 Glitch filter

In Pulse Counter mode, when the glitch filter is enabled, the output of the glitch filter directly clocks the CNR. When the LPTMR is first enabled, the output of the glitch filter is asserted, that is, logic 1 for active-high and logic 0 for active-low. The following table shows the change in glitch filter output with the selected input source.

If	Then
The selected input source remains deasserted for at least 2^1 to 2^{15} consecutive prescaler clock rising edges	The glitch filter output will also deassert.
The selected input source remains asserted for at least 2^1 to 2^{15} consecutive prescaler clock rising-edges	The glitch filter output will also assert.

NOTE

The input is only sampled on the rising clock edge.

The CNR will increment each time the glitch filter output asserts. In Pulse Counter mode, the maximum rate at which the CNR can increment is once every 2^2 to 2^{16} prescaler clock edges. When first enabled, the glitch filter will wait an additional one or two prescaler clock edges due to synchronization logic.

32.4.3.4 Glitch filter bypassed

In Pulse Counter mode, when the glitch filter is bypassed, the selected input source increments the CNR every time it asserts. Before the LPTMR is first enabled, the selected input source is forced to be asserted. This prevents the CNR from incrementing if the selected input source is already asserted when the LPTMR is first enabled.

32.4.4 LPTMR compare

When the CNR equals the value of the CMR and increments, the following events occur:

- CSR[TCF] is set.
- LPTMR interrupt is generated if CSR[TIE] is also set.
- LPTMR hardware trigger is generated.
- CNR is reset if CSR[TFC] is clear.

When the LPTMR is enabled, the CMR can be altered only when CSR[TCF] is set. When updating the CMR, the CMR must be written and CSR[TCF] must be cleared before the LPTMR counter has incremented past the new LPTMR compare value.

32.4.5 LPTMR counter

The CNR increments by one on every:

- Prescaler clock in Time Counter mode with prescaler bypassed
- Prescaler output in Time Counter mode with prescaler enabled
- Input source assertion in Pulse Counter mode with glitch filter bypassed
- Glitch filter output in Pulse Counter mode with glitch filter enabled

The CNR is reset when the LPTMR is disabled or if the counter register overflows. If CSR[TFC] is cleared, then the CNR is also reset whenever CSR[TCF] is set.

The CNR continues incrementing when the core is halted in Debug mode.

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.

32.4.6 LPTMR hardware trigger

The LPTMR hardware trigger asserts at the same time the CSR[TCF] is set and can be used to trigger hardware events in other peripherals without software intervention. The hardware trigger is always enabled.

When	Then
The CMR is set to 0 with CSR[TFC] clear	The LPTMR hardware trigger will assert on the first compare and does not deassert.
The CMR is set to a nonzero value, or, if CSR[TFC] is set	The LPTMR hardware trigger will assert on each compare and deassert on the following increment of the CNR.

32.4.7 LPTMR interrupt

The LPTMR interrupt is generated whenever CSR[TIE] and CSR[TCF] are set. CSR[TCF] is cleared by disabling the LPTMR or by writing a logic 1 to it.

CSR[TIE] can be altered and CSR[TCF] can be cleared while the LPTMR is enabled.

The LPTMR interrupt is generated asynchronously to the system clock and can be used to generate a wakeup from any low-power mode, including the low-leakage modes, provided the LPTMR is enabled as a wakeup source.

Chapter 33
FlexTimer (FTM)

33.1 Introduction

The FlexTimer module is a two to eight channel timer which supports input capture, output compare, and the generation of PWM signals to control electric motor and power management applications. The FTM time reference is a 16-bit counter that can be used as an unsigned or signed counter.

33.1.1 FlexTimer philosophy

The FlexTimer is built upon a very simple timer used for many years on Freescale's 8-bit microcontrollers, the HCS08 Timer PWM Module – TPM. The FlexTimer extends the functionality to meet the demands of motor control, digital lighting solutions, and power conversion, while providing low cost and backwards compatibility with the TPM module.

Several key enhancements are made: signed up-counter, dead time insertion hardware, fault control inputs, enhanced triggering functionality and initialization, and polarity control.

All of the features common with the TPM module have fully backwards compatible register assignments and the FlexTimer can use code on the same core platform without change to perform the same functions. A small exception to this is when the FlexTimer clock frequency is twice bus clock frequency to provide extra resolution for high speed PWM applications.

Motor control and power conversion features have been added through a dedicated set of registers. The new features, such as hardware dead time insertion, polarity, fault control, and masking, greatly reduce loading on the execution software and are usually each controlled by a group of registers. All of the new features are disabled after reset by default.

Flextimer input triggers can come directly from other modules integrated on the chip, such as comparators or ADCs, to automatically initiate timer functions. These triggers can be linked in a variety of ways during integration of the modules so please note carefully the options available for used FlexTimer configuration.

All main user access registers are buffered to ease the load on the executing software. A number of trigger options exist to determine which registers are updated with this user defined data.

33.1.2 Features

The FTM features include:

- Selectable FTM source clock:
  - Source clock can be the system clock, the fixed frequency clock, or an external clock
  - Fixed frequency clock is an additional clock input to allow the selection of an on chip clock source other than the system clock
  - Selecting external clock connects FTM clock to a chip level input pin therefore allowing to synchronize the FTM counter with an off chip clock source
- Prescaler divide-by 1, 2, 4, 8, 16, 32, 64, or 128
- FTM has a 16-bit counter
  - It can be a free-running counter or a counter with initial and final value
  - The counting can be up or up-down
- Each channel can be configured for input capture, output compare, or edge-aligned PWM mode
- In input capture mode:
  - The capture can occur on rising edges, falling edges or both edges
  - An input filter can be selected for some channels
- In output compare mode the output signal can be set, cleared, or toggled on match
- All channels can be configured for center-aligned PWM mode
- Each pair of channels can be combined to generate a PWM signal with independent control of both edges of PWM signal

- The FTM channels can operate as pairs with equal outputs, pairs with complementary outputs, or independent channels with independent outputs
- The deadtime insertion is available for each complementary pair
- Generation of triggers (match trigger)
- Software control of PWM outputs
- Up to four fault inputs for global fault control
- The polarity of each channel is configurable
- The generation of an interrupt per channel
- The generation of an interrupt when the counter overflows
- The generation of an interrupt when the fault condition is detected
- Synchronized loading of write buffered FTM registers
- Write protection for critical registers
- Backwards compatible with TPM
- Testing of input captures for a stuck at zero and one conditions
- Dual edge capture for pulse and period width measurement
- For instances of the module that support it: Quadrature decoder with input filters, relative position counting and interrupt on position count or capture of position count on external event

33.1.3 Modes of operation

When the MCU is in active BDM background or BDM foreground mode, the FTM temporarily suspends all counting until the MCU returns to normal user operating mode. During stop mode, all FTM input clocks are stopped, so the FTM is effectively disabled until clocks resume. During wait mode, the FTM continues to operate normally. If the FTM does not need to produce a real time reference or provide the interrupt sources needed to wake the MCU from wait mode, the power can then be saved by disabling FTM functions before entering wait mode.

33.1.4 Block diagram

The FTM uses one input/output (I/O) pin per channel, CHn (FTM channel (n)) where n is the channel number (0–7).

The following figure shows the FTM structure. The central component of the FTM is the 16-bit counter with programmable initial and final values and its counting can be up or up-down.

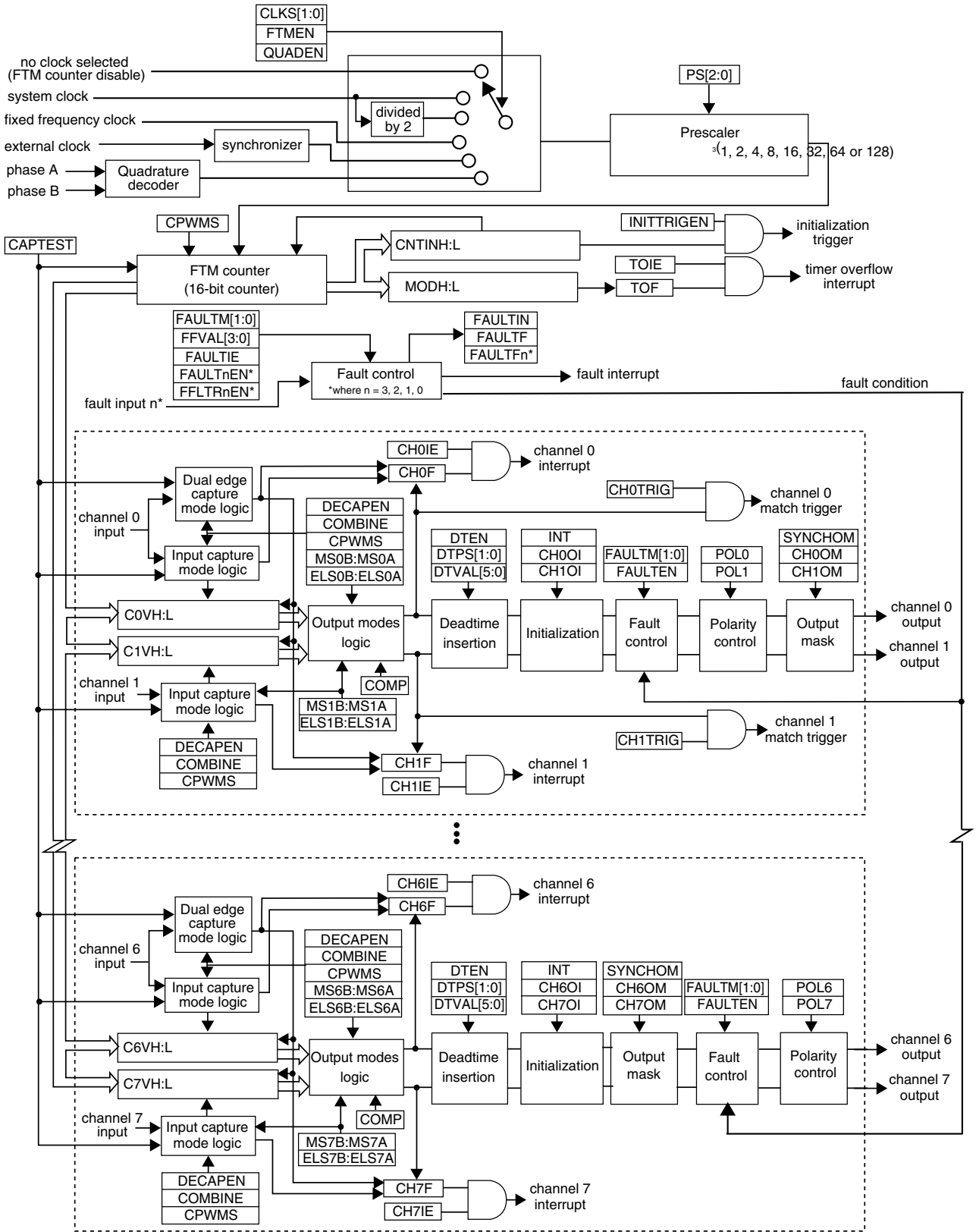


Figure 33-1. FTM block diagram

33.2 Signal description

The following table shows the user-accessible signals for the FTM.

NOTE

The PHA and PHB signals are user accessible if the quadrature decoder feature is supported.

Table 33-1. Signal properties

Name	Function
EXTCLK	External clock – FTM external clock can be selected to drive the FTM counter.
CHn[1]	Channel (n) – I/O pin associated with FTM channel (n).
FAULTj[2]	Fault input (j) – input pin associated with fault input (j).
PHA	Quadrature decoder phase A input – input pin associated with quadrature decoder phase A.
PHB	Quadrature decoder phase B input – input pin associated with quadrature decoder phase B.

1. n = channel number (0 to 7)
2. j = fault input (0 to 3)

33.2.1 EXTCLK — FTM external clock

The external clock input signal is used as the FTM counter clock if selected by CLKS[1:0] bits in the SC register. This clock signal must not exceed 1/4 of system clock frequency. The FTM counter prescaler selection and settings are also used when an external clock is selected.

33.2.2 CHn — FTM channel (n) I/O pin

Each FTM channel can be configured to operate either as input or output. The direction associated with each channel, input or output, is selected according to the mode assigned for that channel.

33.2.3 FAULTj — FTM fault input

The fault input signals are used to control the CHn channel output state. If a fault is detected, the FAULTj signal is asserted and the channel output is put in a safe state. The behavior of the fault logic is defined by the FAULTM[1:0] control bits in the MODE register and FAULTEN bit in the COMBINEm register. Note that each FAULTj input

may affect all channels selectively since FAULTM[1:0] and FAULTEN control bits are defined for each pair of channels. Each FAULTj input is activated by its corresponding FAULTjEN bit in the FLTCTRL register.

33.2.4 PHA — FTM Quadrature Decoder Phase A Input

The quadrature decoder phase A input is used when the quadrature decoder mode is selected (if the quadrature decoder feature is supported). The phase A input signal is one of the signals that control the FTM counter increment or decrement in the quadrature decoder mode (Quadrature Decoder Mode).

33.2.5 PHB — FTM Quadrature Decoder Phase B Input

The quadrature decoder phase B input is used when the quadrature decoder mode is selected (if the quadrature decoder feature is supported). The phase B input signal is one of the signals that control the FTM counter increment or decrement in the quadrature decoder mode (Quadrature Decoder Mode).

33.3 Memory map and register definition

This section provides a detailed description of all FTM registers.

33.3.1 Module memory map

This section presents a high-level summary of the FTM registers and how they are mapped.

The FTM memory map can be split into two sets of registers. The first set has the original TPM registers.

Starting with Counter Initial Value High (CNTINH), the second set has the FTM specific registers. Any second set registers, or bits within these registers, that are used by an unavailable function in the FTM configuration remain in the memory map and in the reset value even though they have no active function.

Note

Do not write to the FTM specific registers (second set registers) when FTMEN = 0.

33.3.2 Register descriptions

This section consists of register descriptions in address order.

FTM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_8300	Status and Control (FTM0_SC)	8	R/W	00h	33.3.3/611
FFFF_8301	Counter High (FTM0_CNTH)	8	R/W	00h	33.3.4/612
FFFF_8302	Counter Low (FTM0_CNTL)	8	R/W	00h	33.3.5/613
FFFF_8303	Modulo High (FTM0_MODH)	8	R/W	00h	33.3.6/614
FFFF_8304	Modulo Low (FTM0_MODL)	8	R/W	00h	33.3.7/614
FFFF_8305	Channel Status and Control (FTM0_C0SC)	8	R/W	00h	33.3.8/615
FFFF_8306	Channel Value High (FTM0_C0VH)	8	R/W	00h	33.3.9/618
FFFF_8307	Channel Value Low (FTM0_C0VL)	8	R/W	00h	33.3.10/ 619
FFFF_8308	Channel Status and Control (FTM0_C1SC)	8	R/W	00h	33.3.8/615
FFFF_8309	Channel Value High (FTM0_C1VH)	8	R/W	00h	33.3.9/618
FFFF_830A	Channel Value Low (FTM0_C1VL)	8	R/W	00h	33.3.10/ 619
FFFF_830B	Channel Status and Control (FTM0_C2SC)	8	R/W	00h	33.3.8/615
FFFF_830C	Channel Value High (FTM0_C2VH)	8	R/W	00h	33.3.9/618
FFFF_830D	Channel Value Low (FTM0_C2VL)	8	R/W	00h	33.3.10/ 619
FFFF_830E	Channel Status and Control (FTM0_C3SC)	8	R/W	00h	33.3.8/615
FFFF_830F	Channel Value High (FTM0_C3VH)	8	R/W	00h	33.3.9/618
FFFF_8310	Channel Value Low (FTM0_C3VL)	8	R/W	00h	33.3.10/ 619
FFFF_8311	Channel Status and Control (FTM0_C4SC)	8	R/W	00h	33.3.8/615
FFFF_8312	Channel Value High (FTM0_C4VH)	8	R/W	00h	33.3.9/618
FFFF_8313	Channel Value Low (FTM0_C4VL)	8	R/W	00h	33.3.10/ 619
FFFF_8314	Channel Status and Control (FTM0_C5SC)	8	R/W	00h	33.3.8/615
FFFF_8315	Channel Value High (FTM0_C5VH)	8	R/W	00h	33.3.9/618
FFFF_8316	Channel Value Low (FTM0_C5VL)	8	R/W	00h	33.3.10/ 619
FFFF_8320	Counter Initial Value High (FTM0_CNTINH)	8	R/W	00h	33.3.11/ 619
FFFF_8321	Counter Initial Value Low (FTM0_CNTINL)	8	R/W	00h	33.3.12/ 620
FFFF_8322	Capture and Compare Status (FTM0_STATUS)	8	R/W	00h	33.3.13/ 621

Table continues on the next page...

FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_8323	Features Mode Selection (FTM0_MODE)	8	R/W	04h	33.3.14/ 622
FFFF_8324	Synchronization (FTM0_SYNC)	8	R/W	00h	33.3.15/ 624
FFFF_8325	Initial State for Channel Output (FTM0_OUTINIT)	8	R/W	00h	33.3.16/ 626
FFFF_8326	Output Mask (FTM0_OUTMASK)	8	R/W	00h	33.3.17/ 627
FFFF_8327	Function for Linked Channels (FTM0_COMBINE0)	8	R/W	00h	33.3.18/ 629
FFFF_8328	Function for Linked Channels (FTM0_COMBINE1)	8	R/W	00h	33.3.18/ 629
FFFF_8329	Function for Linked Channels (FTM0_COMBINE2)	8	R/W	00h	33.3.18/ 629
FFFF_832B	Deadtime Insertion Control (FTM0_DEADTIME)	8	R/W	00h	33.3.19/ 630
FFFF_832C	External Trigger (FTM0_EXTTRIG)	8	R/W	00h	33.3.20/ 631
FFFF_832D	Channels Polarity (FTM0_POL)	8	R/W	00h	33.3.21/ 632
FFFF_832E	Fault Mode Status (FTM0_FMS)	8	R/W	00h	33.3.22/ 634
FFFF_832F	Input Capture Filter Control (FTM0_FILTER0)	8	R/W	00h	33.3.23/ 636
FFFF_8330	Input Capture Filter Control (FTM0_FILTER1)	8	R/W	00h	33.3.23/ 636
FFFF_8331	Fault Input Filter Control (FTM0_FLTFILTER)	8	R/W	00h	33.3.24/ 637
FFFF_8332	Fault Input Control (FTM0_FLTCTRL)	8	R/W	00h	33.3.25/ 637
FFFF_8333	Quadrature Decoder Control and Status (FTM0_QDCTRL)	8	R/W	00h	33.3.26/ 639
FFFF_8340	Status and Control (FTM1_SC)	8	R/W	00h	33.3.3/611
FFFF_8341	Counter High (FTM1_CNTH)	8	R/W	00h	33.3.4/612
FFFF_8342	Counter Low (FTM1_CNTL)	8	R/W	00h	33.3.5/613
FFFF_8343	Modulo High (FTM1_MODH)	8	R/W	00h	33.3.6/614
FFFF_8344	Modulo Low (FTM1_MODL)	8	R/W	00h	33.3.7/614
FFFF_8345	Channel Status and Control (FTM1_C0SC)	8	R/W	00h	33.3.8/615
FFFF_8346	Channel Value High (FTM1_C0VH)	8	R/W	00h	33.3.9/618
FFFF_8347	Channel Value Low (FTM1_C0VL)	8	R/W	00h	33.3.10/ 619
FFFF_8348	Channel Status and Control (FTM1_C1SC)	8	R/W	00h	33.3.8/615
FFFF_8349	Channel Value High (FTM1_C1VH)	8	R/W	00h	33.3.9/618

Table continues on the next page...

FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_834A	Channel Value Low (FTM1_C1VL)	8	R/W	00h	33.3.10/ 619
FFFF_834B	Channel Status and Control (FTM1_C2SC)	8	R/W	00h	33.3.8/615
FFFF_834C	Channel Value High (FTM1_C2VH)	8	R/W	00h	33.3.9/618
FFFF_834D	Channel Value Low (FTM1_C2VL)	8	R/W	00h	33.3.10/ 619
FFFF_834E	Channel Status and Control (FTM1_C3SC)	8	R/W	00h	33.3.8/615
FFFF_834F	Channel Value High (FTM1_C3VH)	8	R/W	00h	33.3.9/618
FFFF_8350	Channel Value Low (FTM1_C3VL)	8	R/W	00h	33.3.10/ 619
FFFF_8351	Channel Status and Control (FTM1_C4SC)	8	R/W	00h	33.3.8/615
FFFF_8352	Channel Value High (FTM1_C4VH)	8	R/W	00h	33.3.9/618
FFFF_8353	Channel Value Low (FTM1_C4VL)	8	R/W	00h	33.3.10/ 619
FFFF_8354	Channel Status and Control (FTM1_C5SC)	8	R/W	00h	33.3.8/615
FFFF_8355	Channel Value High (FTM1_C5VH)	8	R/W	00h	33.3.9/618
FFFF_8356	Channel Value Low (FTM1_C5VL)	8	R/W	00h	33.3.10/ 619
FFFF_8360	Counter Initial Value High (FTM1_CNTINH)	8	R/W	00h	33.3.11/ 619
FFFF_8361	Counter Initial Value Low (FTM1_CNTINL)	8	R/W	00h	33.3.12/ 620
FFFF_8362	Capture and Compare Status (FTM1_STATUS)	8	R/W	00h	33.3.13/ 621
FFFF_8363	Features Mode Selection (FTM1_MODE)	8	R/W	04h	33.3.14/ 622
FFFF_8364	Synchronization (FTM1_SYNC)	8	R/W	00h	33.3.15/ 624
FFFF_8365	Initial State for Channel Output (FTM1_OUTINIT)	8	R/W	00h	33.3.16/ 626
FFFF_8366	Output Mask (FTM1_OUTMASK)	8	R/W	00h	33.3.17/ 627
FFFF_8367	Function for Linked Channels (FTM1_COMBINE0)	8	R/W	00h	33.3.18/ 629
FFFF_8368	Function for Linked Channels (FTM1_COMBINE1)	8	R/W	00h	33.3.18/ 629
FFFF_8369	Function for Linked Channels (FTM1_COMBINE2)	8	R/W	00h	33.3.18/ 629
FFFF_836B	Deadtime Insertion Control (FTM1_DEADTIME)	8	R/W	00h	33.3.19/ 630
FFFF_836C	External Trigger (FTM1_EXTTRIG)	8	R/W	00h	33.3.20/ 631

Table continues on the next page...

FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_836D	Channels Polarity (FTM1_POL)	8	R/W	00h	33.3.21/ 632
FFFF_836E	Fault Mode Status (FTM1_FMS)	8	R/W	00h	33.3.22/ 634
FFFF_836F	Input Capture Filter Control (FTM1_FILTER0)	8	R/W	00h	33.3.23/ 636
FFFF_8370	Input Capture Filter Control (FTM1_FILTER1)	8	R/W	00h	33.3.23/ 636
FFFF_8371	Fault Input Filter Control (FTM1_FLTFILTER)	8	R/W	00h	33.3.24/ 637
FFFF_8372	Fault Input Control (FTM1_FLTCTRL)	8	R/W	00h	33.3.25/ 637
FFFF_8373	Quadrature Decoder Control and Status (FTM1_QDCTRL)	8	R/W	00h	33.3.26/ 639

33.3.3 Status and Control (FTMx_SC)

SC contains the overflow status flag and control bits used to configure the interrupt enable, FTM configuration, clock source, and prescaler factor. These controls relate to all channels within this module.

Address: Base address + 0h offset

Bit	7	6	5	4	3	2	1	0
Read	TOF	TOIE	CPWMS	CLKS		PS		
Write	0							
Reset	0	0	0	0	0	0	0	0

FTMx_SC field descriptions

Field	Description
7 TOF	Timer Overflow Flag Set by hardware when the FTM counter passes the value in the Counter Modulo registers. The TOF bit is cleared by reading the SC register while TOF is set and then writing a 0 to TOF bit. Writing a 1 to TOF has no effect. If another FTM overflow occurs between the read and write operations, the write operation has no effect; therefore, TOF remains set indicating an overflow has occurred. In this case a TOF interrupt request is not lost due to the clearing sequence for a previous TOF. 0 FTM counter has not overflowed. 1 FTM counter has overflowed.
6 TOIE	Timer Overflow Interrupt Enable

Table continues on the next page...

FTMx_SC field descriptions (continued)

Field	Description
	Enables FTM overflow interrupts. 0 Disable TOF interrupts. Use software polling. 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 FTM to operate in up-down counting mode. CPWMS is write protected. It can be written only when MODE[WPDIS] = 1. 0 FTM counter operates in up counting mode. 1 FTM counter operates in up-down counting mode.
4–3 CLKS	Clock Source Selection Selects one of the three FTM counter clock sources. CLKS is write protected. It can be written only when MODE[WPDIS] = 1. 00 No clock selected (this in effect disables the FTM counter). 01 If MODE[FTMEN] = 0, the System clock divided by 2 is selected. If MODE[FTMEN] = 1, the System clock is selected. 10 Fixed frequency clock 11 External clock
2–0 PS	Prescale Factor Selection Selects one of 8 division factors for the clock source selected by CLKS. The new prescaler factor affects the clock source on the next system clock cycle after the new value is updated into the register bits. PS is write protected. It can be written only when MODE[WPDIS] = 1. 000 Divide by 1 001 Divide by 2 010 Divide by 4 011 Divide by 8 100 Divide by 16 101 Divide by 32 110 Divide by 64 111 Divide by 128

33.3.4 Counter High (FTMx_CNTH)

The Counter registers contain the high and low bytes of the counter value. Reading either byte latches the contents of both bytes into a buffer where they remain latched until the other half is read. This allows coherent 16-bit reads in either big-endian or little-endian order which makes this more friendly to various compiler implementations. The coherency mechanism is automatically restarted by an MCU reset or any write to the Status and Control register.

Writing any value to COUNT_H or COUNT_L updates the FTM counter with its initial 16-bit value (contained in the Counter Initial Value registers) and resets the read coherency mechanism, regardless of the data involved in the write.

When BDM is active, the FTM counter is frozen (this is the value that you may read); the read coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both counter bytes are read while BDM is active. This assures that if you were in the middle of reading a 16-bit register when BDM became active, it reads the appropriate value from the other half of the 16-bit value after returning to normal execution.

Address: Base address + 1h offset

Bit	7	6	5	4	3	2	1	0
Read Write	COUNT_H							
Reset	0	0	0	0	0	0	0	0

FTMx_CNTH field descriptions

Field	Description
7–0 COUNT_H	Counter value high byte

33.3.5 Counter Low (FTMx_CNTL)

See the description for the Counter High register.

Address: Base address + 2h offset

Bit	7	6	5	4	3	2	1	0
Read Write	COUNT_L							
Reset	0	0	0	0	0	0	0	0

FTMx_CNTL field descriptions

Field	Description
7–0 COUNT_L	Counter value low byte

33.3.6 Modulo High (FTMx_MODH)

The Modulo registers contain the high and low bytes of the modulo value for the FTM counter. After the FTM counter reaches the modulo value, the overflow flag (TOF) becomes set at the next clock, and the next value of FTM counter depends on the selected counting method (Counter).

Writing to either byte latches the value into a buffer. The register is updated with the value of their write buffer according to Update of the registers with write buffers.

If MODE[FTMEN] = 0, this write coherency mechanism may be manually reset by writing to the SC register whether BDM is active or not.

When BDM is active, this write coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both bytes of the modulo register are written while BDM is active. Any write to the modulo register bypasses the buffer latches and directly writes to the modulo register while BDM is active.

It is recommended to initialize the FTM counter, by writing to CNTH or CNTL, before writing to the FTM modulo register to avoid confusion about when the first counter overflow will occur.

Address: Base address + 3h offset

Bit	7	6	5	4	3	2	1	0
Read Write	MOD_H							
Reset	0	0	0	0	0	0	0	0

FTMx_MODH field descriptions

Field	Description
7–0 MOD_H	High byte of the modulo value

33.3.7 Modulo Low (FTMx_MODL)

See the description for the Modulo High register.

Address: Base address + 4h offset

Bit	7	6	5	4	3	2	1	0
Read Write	MOD_L							
Reset	0	0	0	0	0	0	0	0

FTMx_MODL field descriptions

Field	Description
7–0 MOD_L	Low byte of the modulo value

33.3.8 Channel Status and Control (FTMx_C*n*SC)

CnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function.

Table 33-71. Mode, edge, and level selection

DECAPEN	COMBINE	CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	X	X	XX	00	None	Pin not used for FTM

Table continues on the next page...

Table 33-71. Mode, edge, and level selection (continued)

DECAPEN	COMBINE	CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
0	0	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
			1X	10	Edge-aligned PWM	High-true pulses (clear Output on match)
				X1		Low-true pulses (set Output on match)
		1	XX	10	Center-aligned PWM	High-true pulses (clear Output on match-up)
				X1		Low-true pulses (set Output on match-up)
	1	0	XX	10	Combine PWM	High-true pulses (set on channel (n) match, and clear on channel (n+1) match)
				X1		Low-true pulses (clear on channel (n) match, and set on channel (n +1) match)
1	0	0	X0	See the following table.	Dual Edge Capture Mode	One-shot capture mode
			X1			Continuous capture mode

Table 33-72. Dual edge capture mode — edge polarity selection

ELSnB	ELSnA	Channel Port Enable	Detected Edges
0	0	Disabled	No edge

Table continues on the next page...

Table 33-72. Dual edge capture mode — edge polarity selection (continued)

ELSnB	ELSnA	Channel Port Enable	Detected Edges
0	1	Enabled	Rising edge
1	0	Enabled	Falling edge
1	1	Enabled	Rising and falling edges

Address: Base address + 5h offset + (3d × i), where i=0d to 5d

Bit	7	6	5	4	3	2	1	0
Read	CHF	CHIE	MSB	MSA	ELSB	ELSA	0	DMA
Write	0							
Reset	0	0	0	0	0	0	0	0

FTMx_C*n*SC field descriptions

Field	Description
7 CHF	Channel Flag Set by hardware when an event occurs on the channel. CHF is cleared by reading the CnSC register while CHnF is set and then writing a 0 to the CHF bit. Writing a 1 to CHF has no effect. If another event occurs between the read and write operations, the write operation has no effect; therefore, CHF remains set indicating an event has occurred. In this case a CHF interrupt request is not lost due to the clearing sequence for a previous CHF. 0 No channel event has occurred. 1 A channel event has occurred.
6 CHIE	Channel Interrupt Enable Enables channel interrupts. 0 Disable channel interrupts. Use software polling. 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. See the table in the register description. MSB is write protected. It can be written only when MODE[WPDIS] = 1.
4 MSA	Channel Mode Select Used for further selections in the channel logic. Its functionality is dependent on the channel mode. See the table in the register description. MSA is write protected. It can be written only when MODE[WPDIS] = 1.
3 ELSB	Edge or Level Select The functionality of ELSB and ELSA depends on the channel mode. See the table in the register description. ELSB is write protected. It can be written only when MODE[WPDIS] = 1.
2 ELSA	Edge or Level Select

Table continues on the next page...

FTMx_C*n*SC field descriptions (continued)

Field	Description
	The functionality of ELSB and ELSA depends on the channel mode. See the table in the register description. ELSA is write protected. It can be written only when MODE[WPDIS] = 1.
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.

33.3.9 Channel Value High (FTMx_C*n*VH)

These registers contain the captured FTM counter value of the input capture function or the match value for the output modes.

In input capture, capture test, and dual edge capture modes, reading a single byte in CnV latches the contents into a buffer where they remain latched until the other byte is read. This latching mechanism also resets, or becomes unlatched, when the CnSC register is written whether BDM mode is active or not. Any write to the channel registers is ignored during these input modes.

When BDM is active, the read coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both bytes of the channel value register are read while BDM is active. This ensures that if you were in the middle of reading a 16-bit register when BDM became active, it reads the appropriate value from the other half of the 16-bit value after returning to normal execution. Any read of the CnV registers in BDM mode bypasses the buffer latches and returns the value of these registers and not the value of their read buffer.

In output modes, writing to CnV latches the value into a buffer. The registers are updated with the value of their write buffer according to Update of the registers with write buffers.

If MODE[FTMEN] = 0, this write coherency mechanism may be manually reset by writing to the CnSC register whether BDM mode is active or not. This latching mechanism allows coherent 16-bit writes in either big-endian or little-endian order, which is friendly to various compiler implementations.

When BDM is active, the write coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active even if one or both bytes of the channel value register are written while BDM is active. Any write to the CnV registers bypasses the buffer latches and writes directly to the register while BDM is active. The values written to the channel value registers while BDM is active are used in output modes operation after normal execution resumes. Writes to the channel value registers while BDM is active do not interfere with the partial completion of a coherency sequence. After the write coherency mechanism has been fully exercised, the channel value registers are updated using the buffered values while BDM was not active.

Address: Base address + 6h offset + (3d × i), where i=0d to 5d

Bit	7	6	5	4	3	2	1	0
Read Write	VAL_H							
Reset	0	0	0	0	0	0	0	0

FTMx_CnVH field descriptions

Field	Description
7–0 VAL_H	Channel Value High Byte Captured FTM counter value of the input capture function or the match value for the output modes

33.3.10 Channel Value Low (FTMx_CnVL)

See the description for the Channel Value High register.

Address: Base address + 7h offset + (3d × i), where i=0d to 5d

Bit	7	6	5	4	3	2	1	0
Read Write	VAL_L							
Reset	0	0	0	0	0	0	0	0

FTMx_CnVL field descriptions

Field	Description
7–0 VAL_L	Channel Value Low Byte Captured FTM counter value of the input capture function or the match value for the output modes

33.3.11 Counter Initial Value High (FTMx_CNTINH)

The Counter Initial Value registers contain the high and low bytes of the initial value for the FTM counter.

Writing to either byte latches the value into a buffer. The registers are updated with the value of their write buffer.

When BDM is active, the write coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both bytes of the counter initial value register are written while BDM is active. Any write to the counter initial value registers bypasses the buffer latches and writes directly to the counter initial value register while BDM is active.

The first time that the FTM clock is selected (first write to change the CLKS bits to a non-zero value), FTM counter starts with the value 0x0000. To avoid this behavior, before the first write to select the FTM clock, write the new value to the Counter Initial Value registers and then initialize the FTM counter by writing any value to CNT).

Address: Base address + 20h offset

Bit	7	6	5	4	3	2	1	0
Read Write	INIT_H							
Reset	0	0	0	0	0	0	0	0

FTMx_CNTINH field descriptions

Field	Description
7–0 INIT_H	Counter Initial Value High Byte

33.3.12 Counter Initial Value Low (FTMx_CNTINL)

See the description for the Counter Initial Value High register.

Address: Base address + 21h offset

Bit	7	6	5	4	3	2	1	0
Read Write	INIT_L							
Reset	0	0	0	0	0	0	0	0

FTMx_CNTINL field descriptions

Field	Description
7–0 INIT_L	Counter Initial Value Low Byte

33.3.13 Capture and Compare Status (FTMx_STATUS)

STATUS contains a copy of the status flag CHnF bit, in CnSC, for each FTM channel for software convenience.

Each CHnF bit in STATUS is a mirror of CHnF bit in CnSC. All CHnF bits can be checked using only one read of STATUS. All CHnF bits can be cleared by reading STATUS followed by writing 0x00 to STATUS.

Hardware sets the individual channel flags when an event occurs on the channel. CHF is cleared by reading STATUS while CHnF is set and then writing a 0 to the CHF bit. Writing a 1 to CHF has no effect.

If another event occurs between the read and write operations, the write operation has no effect; therefore, CHF remains set indicating an event has occurred. In this case, a CHF interrupt request is not lost due to the clearing sequence for a previous CHF.

NOTE

The use of STATUS register is available only when (MODE[FTMEN] = 1), (COMBINE = 1), and (CPWMS = 0). The use of this register with (MODE[FTMEN] = 0), (COMBINE = 0), or (CPWMS = 1) is not recommended and its results are not guaranteed.

Address: Base address + 22h offset

Bit	7	6	5	4	3	2	1	0
Read	CH7F	CH6F	CH5F	CH4F	CH3F	CH2F	CH1F	CH0F
Write	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0

FTMx_STATUS field descriptions

Field	Description
7 CH7F	Channel 7 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
6 CH6F	Channel 6 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.

Table continues on the next page...

FTMx_STATUS field descriptions (continued)

Field	Description
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.

33.3.14 Features Mode Selection (FTMx_MODE)

This register contains the control bits used to configure the fault interrupt and fault control, capture test mode, PWM synchronization, write protection, channel output initialization, and enable the enhanced features of the FTM. These controls relate to all channels within this module.

Address: Base address + 23h offset

Bit	7	6	5	4	3	2	1	0
Read Write	FAULTIE	FAULTM		CAPTEST	PWMSYNC	WPDIS	INIT	FTMEN
Reset	0	0	0	0	0	1	0	0

FTMx_MODE field descriptions

Field	Description
7 FAULTIE	Fault Interrupt Enable Enables the generation of an interrupt when a fault is detected by FTM and the FTM fault control is enabled. 0 Fault control interrupt is disabled. 1 Fault control interrupt is enabled.
6–5 FAULTM	Fault Control Mode Defines the FTM fault control mode. FAULTM is write protected. These bits can be written only if MODE[WPDIS] = 1. 00 Fault control is disabled for all channels. 01 Fault control is enabled for even channels only (channels 0, 2, 4, and 6), and the selected mode is the manual fault clearing. 10 Fault control is enabled for all channels, and the selected mode is the manual fault clearing. 11 Fault control is enabled for all channels, and the selected mode is the automatic fault clearing.
4 CAPTEST	Capture Test Mode Enable Enables the capture test mode. CAPTEST bit is write protected. This bit can be written only if WPDIS = 1. 0 Capture test mode is disabled. 1 Capture test mode is enabled.
3 PWMSYNC	PWM Synchronization Mode Selects which triggers can be used by MOD, CV, CHnOM, and FTM counter synchronization (PWM synchronization). 0 No restrictions. Software and hardware triggers can be used by MOD, CV, CHnOM, and FTM counter synchronization. 1 Software trigger can be used only by MOD and CV synchronization, and hardware triggers can be used only by CHnOM and FTM counter synchronization.
2 WPDIS	Write Protection Disable When write protection is enabled (MODE[WPDIS] = 0), write protected bits can not be written. When write protection is disabled (MODE[WPDIS] = 1), write protected bits can be written. The WPDIS bit is the negation of the WPEN bit. WPDIS is cleared when 1 is written to WPEN. WPDIS is set when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPDIS has no effect. 0 Write protection is enabled. 1 Write protection is disabled.
1 INIT	Initialize the Output Channels When a 1 is written to INIT bit the output channels are initialized according to the state of their corresponding bit in the OUTINIT register. Writing a 0 to INIT bit has no effect. The INIT bit is always read as 0.
0 FTMEN	FTM Enable This bit is write protected, and can be written only if WPDIS = 1.

Table continues on the next page...

FTMx_MODE field descriptions (continued)

Field	Description
	0 Only the TPM-compatible registers (first set of registers) can be used without any restriction. Do not use the FTM-specific registers. 1 All registers including the FTM-specific registers (second set of registers) are available for use with no restrictions.

33.3.15 Synchronization (FTMx_SYNC)

This register configures the PWM synchronization.

A synchronization event can perform the synchronized update of MOD, CV, and OUTMASK registers with the value of their write buffer and the FTM counter initialization.

NOTE

The software trigger (SWSYNC bit) and hardware triggers (TRIG0, TRIG1, and TRIG2 bits) have a potential conflict if used together. Use only hardware or software triggers but not both at the same time, otherwise unpredictable behavior is likely to happen.

The selection of the boundary cycle (CNTMAX and CNTMIN bits) is intended to provide the update of MOD, CNTIN, and CV across all enabled channels simultaneously. The use of the boundary cycle selection together with TRIG0, TRIG1, or TRIG2 bits is likely to result in unpredictable behavior.

The MODE[PWMSYNC] bit determines which type of trigger event controls the functions enabled by the SYNC register.

Address: Base address + 24h offset

Bit	7	6	5	4	3	2	1	0
Read Write	SWSYNC	TRIG2	TRIG1	TRIG0	SYNCHOM	REINIT	CNTMAX	CNTMIN
Reset	0	0	0	0	0	0	0	0

FTMx_SYNC field descriptions

Field	Description
7 SWSYNC	PWM Synchronization Software Trigger Selects the software trigger as the PWM synchronization trigger. The software trigger occurs when a 1 is written to SWSYNC bit.

Table continues on the next page...

FTMx_SYNC field descriptions (continued)

Field	Description
	0 Software trigger is not selected. 1 Software trigger is selected.
6 TRIG2	PWM Synchronization External Trigger 2 Selects external trigger 2 as the PWM synchronization trigger. External trigger 2 occurs when the FTM detects a rising edge in the trigger 2 input signal. 0 External trigger 2 is not selected. 1 External trigger 2 is selected.
5 TRIG1	PWM Synchronization External Trigger 1 Selects external trigger 1 as the PWM synchronization trigger. External trigger 1 occurs when the FTM detects a rising edge in the trigger 1 input signal. 0 External trigger 1 is not selected. 1 External trigger 1 is selected.
4 TRIG0	PWM Synchronization External Trigger 0 Selects external trigger 0 as the PWM synchronization trigger. External trigger 0 occurs when the FTM detects a rising edge in the trigger 0 input signal. 0 External trigger 0 is not selected. 1 External trigger 0 is selected.
3 SYNCHOM	Output Mask Synchronization Selects when the CHnOM bits in register OUTMASK are updated with the value of their write buffer. 0 CHnOM bits are updated with the value of the OUTMASK write buffer in all rising edges of the system clock. 1 CHnOM bits are updated with the value of the OUTMASK write buffer only by the PWM synchronization.
2 REINIT	FTM Counter Reinitialization by Synchronization (See “FTM Counter Synchronization”) Determines if the FTM counter is reinitialized when the selected trigger for the synchronization is detected. 0 FTM counter continues to count normally. 1 FTM counter is updated with its initial value when the selected trigger is detected.
1 CNTMAX	Maximum Boundary Cycle Enable Determines when the MOD, CNTIN, and CV registers are updated with their write buffer contents following a PWM synchronization event. If CNTMAX is enabled, the registers are updated when the FTM counter reaches its maximum value MOD. 0 The maximum boundary cycle is disabled. 1 The maximum boundary cycle is enabled.
0 CNTMIN	Minimum Boundary Cycle Enable Determines when the MOD and CV registers are updated with their write buffer contents after a PWM synchronization event. If CNTMIN is enabled, the registers are updated when the FTM counter reaches its minimum value CNTIN.

Table continues on the next page...

FTMx_SYNC field descriptions (continued)

Field	Description
	0 The minimum boundary cycle is disabled. 1 The minimum boundary cycle is enabled.

33.3.16 Initial State for Channel Output (FTMx_OUTINIT)

Address: Base address + 25h offset

Bit	7	6	5	4	3	2	1	0
Read Write	CH7OI	CH6OI	CH5OI	CH4OI	CH3OI	CH2OI	CH1OI	CH0OI
Reset	0	0	0	0	0	0	0	0

FTMx_OUTINIT field descriptions

Field	Description
7 CH7OI	Channel 7 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
6 CH6OI	Channel 6 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
5 CH5OI	Channel 5 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
4 CH4OI	Channel 4 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
3 CH3OI	Channel 3 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
2 CH2OI	Channel 2 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs.

Table continues on the next page...

FTMx_OUTINIT field descriptions (continued)

Field	Description
	0 The initialization value is 0. 1 The initialization value is 1.
1 CH1OI	Channel 1 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
0 CH0OI	Channel 0 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.

33.3.17 Output Mask (FTMx_OUTMASK)

This register provides a mask for each FTM channel. The mask of a channel determines if its output responds, that is, it is masked or not, when a match occurs. This feature is used for BLDC control applications where the PWM signal is presented to an electric motor at specific times to provide electronic commutation.

Any write to the OUTMASK register stores the value into a write buffer. The register is updated with the value of its write buffer according to PWM synchronization.

Address: Base address + 26h offset

Bit	7	6	5	4	3	2	1	0
Read Write	CH7OM	CH6OM	CH5OM	CH4OM	CH3OM	CH2OM	CH1OM	CH0OM
Reset	0	0	0	0	0	0	0	0

FTMx_OUTMASK field descriptions

Field	Description
7 CH7OM	Channel 7 Output Mask Defines if the channel output is masked (forced to its inactive state) or unmasked (it continues to operate normally). 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
6 CH6OM	Channel 6 Output Mask Defines if the channel output is masked (forced to its inactive state) or unmasked (it continues to operate normally).

Table continues on the next page...

FTMx_OUTMASK field descriptions (continued)

Field	Description
	0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
5 CH5OM	Channel 5 Output Mask Defines if the channel output is masked (forced to its inactive state) or unmasked (it continues to operate normally). 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
4 CH4OM	Channel 4 Output Mask Defines if the channel output is masked (forced to its inactive state) or unmasked (it continues to operate normally). 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
3 CH3OM	Channel 3 Output Mask Defines if the channel output is masked (forced to its inactive state) or unmasked (it continues to operate normally). 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
2 CH2OM	Channel 2 Output Mask Defines if the channel output is masked (forced to its inactive state) or unmasked (it continues to operate normally). 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
1 CH1OM	Channel 1 Output Mask Defines if the channel output is masked (forced to its inactive state) or unmasked (it continues to operate normally). 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
0 CH0OM	Channel 0 Output Mask Defines if the channel output is masked (forced to its inactive state) or unmasked (it continues to operate normally). 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.

33.3.18 Function for Linked Channels (FTMx_COMBINE*n*)

This register contains the control bits used to configure the fault control, synchronization, deadtime, dual edge capture mode, complementary, and combine features of channels (n) and (n+1).

- COMBINE0 supports channels 0 and 1.
- COMBINE1 supports channels 2 and 3.
- COMBINE2 supports channels 4 and 5.

NOTE

The channel (n) is the even channel and the channel (n+1) is the odd channel of a pair of channels.

Address: Base address + 27h offset + (1d × i), where i=0d to 2d

Bit	7	6	5	4	3	2	1	0
Read	0	FAULTEN	SYNCEN	DTEN	DECAP	DECAPEN	COMP	COMBINE
Write								
Reset	0	0	0	0	0	0	0	0

FTMx_COMBINE*n* field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 FAULTEN	Fault Control Enable Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault control in this pair of channels is disabled. 1 The fault control in this pair of channels is enabled.
5 SYNCEN	Synchronization Enable Enables PWM synchronization of registers C(n)V and C(n+1)V. 0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.
4 DTEN	Deadtime Enable Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.
3 DECAP	Dual Edge Capture Mode Captures

Table continues on the next page...

FTMx_COMBINE*n* field descriptions (continued)

Field	Description
	Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when MODE[FTMEN] = 1 and DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture one-shot mode is selected and when the capture of channel (n+1) event is made. 0 The dual edge captures are inactive. 1 The dual edge captures are active.
2 DECAPEN	Dual Edge Capture Mode Enable Enables the dual edge capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA, and ELS(n+1)B:ELS(n+1)A bits in dual edge capture mode according to the table Mode, Edge, and Level Selection in the description of the CnSC register. This field applies only when MODE[FTMEN] = 1. DECAPEN is write protected, this bit can be written only if MODE[WPDIS] = 1. 0 The dual edge capture mode in this pair of channels is disabled. 1 The dual edge capture mode in this pair of channels is enabled.
1 COMP	Complement of Channel (n) Enables complementary mode for the combined channels. In complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output.
0 COMBINE	Combine Channels Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.

33.3.19 Deadtime Insertion Control (FTMx_DEADTIME)

This register selects the deadtime prescaler factor and deadtime value. All FTM channels use this clock prescaler and this deadtime value for the deadtime insertion.

Address: Base address + 2Bh offset

Bit	7	6	5	4	3	2	1	0
Read Write	DTPS		DTVAL					
Reset	0	0	0	0	0	0	0	0

FTMx_DEADTIME field descriptions

Field	Description
7–6 DTPS	Deadtime Prescaler Value Selects the division factor of the system clock. This prescaled clock is used by the deadtime counter. DTPS is write protected. It can be written only when MODE[WPDIS] = 1. 0x Divide the system clock by 1. 10 Divide the system clock by 4. 11 Divide the system clock by 16.
5–0 DTVAL	Deadtime Value Selects the deadtime insertion value for the deadtime counter. The deadtime counter is clocked by a scaled version of the system clock. See the description of DTPS. Deadtime insert value = (DTPS × DTVAL). DTVAL selects the number of deadtime counts inserted as follows: • When DTVAL is 0, no counts are inserted. • When DTVAL is 1, 1 count is inserted. • When DTVAL is 2, 2 counts are inserted. This pattern continues up to a possible 63 counts. DTVAL is write protected. It can be written only when MODE[WPDIS] = 1.

33.3.20 External Trigger (FTMx_EXTTRIG)

This register indicates when a channel trigger was generated, enables the generation of a trigger when the FTM counter is equal to its initial value, and selects which channels are used in the generation of the channel triggers. Several FTM channels can be selected to generate multiple triggers in one PWM period.

Channels 6 and 7 are not used to generate channel triggers.

Address: Base address + 2Ch offset

Bit	7	6	5	4	3	2	1	0
Read Write	TRIGF	INITTRIGEN	CH1TRIG	CH0TRIG	CH5TRIG	CH4TRIG	CH3TRIG	CH2TRIG
Reset	0	0	0	0	0	0	0	0

FTMx_EXTTRIG field descriptions

Field	Description
7 TRIGF	Channel Trigger Flag Set by hardware when a channel trigger is generated. Clear TRIGF by reading EXTTRIG while TRIGF is set and then writing a 0 to TRIGF. Writing a 1 to TRIGF has no effect. If another channel trigger is generated before the clearing sequence is completed, the sequence is reset so TRIGF remains set after the clear sequence is completed for the earlier TRIGF.

Table continues on the next page...

FTMx_EXTTRIG field descriptions (continued)

Field	Description
	0 No channel trigger was generated. 1 A channel trigger was generated.
6 INITTRIGEN	Initialization Trigger Enable Enables the generation of the trigger when the FTM counter is equal to its initial value. 0 The generation of initialization trigger is disabled. 1 The generation of initialization trigger is enabled.
5 CH1TRIG	Channel 1 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
4 CH0TRIG	Channel 0 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
3 CH5TRIG	Channel 5 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
2 CH4TRIG	Channel 4 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
1 CH3TRIG	Channel 3 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
0 CH2TRIG	Channel 2 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.

33.3.21 Channels Polarity (FTMx_POL)

This register defines the output polarity of the FTM channels.

NOTE

The safe value that is driven in a channel output when the fault control is enabled and a fault condition is detected is the inactive state of the channel. That is, the safe value of a channel is the value of its POL bit.

Address: Base address + 2Dh offset

Bit	7	6	5	4	3	2	1	0
Read Write	POL7	POL6	POL5	POL4	POL3	POL2	POL1	POL0
Reset	0	0	0	0	0	0	0	0

FTMx_POL field descriptions

Field	Description
7 POL7	Channel 7 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
6 POL6	Channel 6 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
5 POL5	Channel 5 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
4 POL4	Channel 4 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
3 POL3	Channel 3 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
2 POL2	Channel 2 Polarity

Table continues on the next page...

FTMx_POL field descriptions (continued)

Field	Description
	Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
1 POL1	Channel 1 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
0 POL0	Channel 0 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.

33.3.22 Fault Mode Status (FTMx_FMS)

This register contains the fault detection flags, write protection enable bit, and the logic OR of the enable fault inputs.

Address: Base address + 2Eh offset

Bit	7	6	5	4	3	2	1	0
Read	FAULTF	WPEN	FAULTIN	0	FAULTF3	FAULTF2	FAULTF1	FAULTF0
Write	0				0	0	0	0
Reset	0	0	0	0	0	0	0	0

FTMx_FMS field descriptions

Field	Description
7 FAULTF	Fault Detection Flag Represents the logic OR of the individual FAULTFn bits. Clear FAULTF by reading the FMS register while FAULTF is set and then writing a 0 to FAULTF while there is no existing fault condition at the enabled fault inputs. Writing a 1 to FAULTF has no effect. If another fault condition is detected in an enabled fault input before the clearing sequence is completed, the sequence is reset so FAULTF remains set after the clearing sequence is completed for the earlier fault condition. FAULTF is also cleared when FAULTFn bits are cleared individually. 0 No fault condition was detected. 1 A fault condition was detected.

Table continues on the next page...

FTMx_FMS field descriptions (continued)

Field	Description
6 WPEN	Write Protection Enable The WPEN bit is the negation of the WPDIS bit. WPEN is set when 1 is written to it. WPEN is cleared when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPEN has no effect. 0 Write protection is disabled. Write protected bits can be written. 1 Write protection is enabled. Write protected bits cannot be written.
5 FAULTIN	Fault Inputs Represents the logic OR of the enabled fault input after its filter, if its filter is enabled, when fault control is enabled. 0 The value of the fault input is 0. 1 The value of the fault input is 1.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 FAULTF3	Fault Detection Flag 3 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected in the fault input. Clear FAULTF by reading the FMS register while FAULTFn is set and then writing a 0 to FAULTFn FAULTF while there is no existing fault condition at the fault input n. Writing a 1 to FAULTFn has no effect. FAULTFn bit is also cleared when FAULTF bit is cleared. If another fault condition is detected at fault input n before the clearing sequence is completed, the sequence is reset so FAULTFn remains set after the clearing sequence is completed for the earlier fault condition. 0 No fault condition was detected in the fault input. 1 A fault condition was detected in the fault input.
2 FAULTF2	Fault Detection Flag 2 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected in the fault input. Clear FAULTF by reading the FMS register while FAULTFn is set and then writing a 0 to FAULTFn FAULTF while there is no existing fault condition at the fault input n. Writing a 1 to FAULTFn has no effect. FAULTFn bit is also cleared when FAULTF bit is cleared. If another fault condition is detected at fault input n before the clearing sequence is completed, the sequence is reset so FAULTFn remains set after the clearing sequence is completed for the earlier fault condition. 0 No fault condition was detected in the fault input. 1 A fault condition was detected in the fault input.
1 FAULTF1	Fault Detection Flag 1 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected in the fault input. Clear FAULTF by reading the FMS register while FAULTFn is set and then writing a 0 to FAULTFn FAULTF while there is no existing fault condition at the fault input n. Writing a 1 to FAULTFn has no effect. FAULTFn bit is also cleared when FAULTF bit is cleared.

Table continues on the next page...

FTMx_FMS field descriptions (continued)

Field	Description
	If another fault condition is detected at fault input n before the clearing sequence is completed, the sequence is reset so FAULTFn remains set after the clearing sequence is completed for the earlier fault condition. 0 No fault condition was detected in the fault input. 1 A fault condition was detected in the fault input.
0 FAULTF0	Fault Detection Flag 0 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected in the fault input. Clear FAULTF by reading the FMS register while FAULTFn is set and then writing a 0 to FAULTFn FAULTF while there is no existing fault condition at the fault input n. Writing a 1 to FAULTFn has no effect. FAULTFn bit is also cleared when FAULTF bit is cleared. If another fault condition is detected at fault input n before the clearing sequence is completed, the sequence is reset so FAULTFn remains set after the clearing sequence is completed for the earlier fault condition. 0 No fault condition was detected in the fault input. 1 A fault condition was detected in the fault input.

33.3.23 Input Capture Filter Control (FTMx_FILTER*n*)

This register selects the filter value for the inputs of channels.

- FILTER0 supports Channels 0 and 1.
- FILTER1 supports Channels 2 and 3.
- Channels 4 and 5 do not have an input filter.

NOTE

Writing to this register has immediate effect and must be done only when the input capture modes of the affected channels are disabled. Failure to do this could result in a missing valid signal.

Address: Base address + 2Fh offset + (1d × i), where i=0d to 1d

Bit	7	6	5	4	3	2	1	0
Read Write	CHoddFVAL				CHevenFVAL			
Reset	0	0	0	0	0	0	0	0

FTMx_FILTER*n* field descriptions

Field	Description
7–4 CHoddFVAL	Input Filter for Odd Channel Selects the filter value for the odd-numbered channel input.

Table continues on the next page...

FTMx_FILTER*n* field descriptions (continued)

Field	Description
	The filter is disabled when the value is zero.
3–0 CHevenFVAL	Input Filter for Even Channel Selects the filter value for the even-numbered channel input. The filter is disabled when the value is zero.

33.3.24 Fault Input Filter Control (FTMx_FLTFILTER)

This register selects the fault inputs and enables the fault input filter.

Address: Base address + 31h offset

Bit	7	6	5	4	3	2	1	0
Read	0				FFVAL			
Write								
Reset	0	0	0	0	0	0	0	0

FTMx_FLTFILTER field descriptions

Field	Description
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–0 FFVAL	Fault Input Filter Selects the filter value for the fault inputs. The fault filter is disabled when the value is zero. **NOTE:** Writing to this field has immediate effect and must be done only when the fault control or the fault input is disabled. Failure to do so could result in a missing fault detection.

33.3.25 Fault Input Control (FTMx_FLTCTRL)

This register selects the fault inputs and enables the fault input filter.

Address: Base address + 32h offset

Bit	7	6	5	4	3	2	1	0
Read Write	FFLTR3EN	FFLTR2EN	FFLTR1EN	FFLTR0EN	FAULT3EN	FAULT2EN	FAULT1EN	FAULT0EN
Reset	0	0	0	0	0	0	0	0

FTMx_FLTCTRL field descriptions

Field	Description
7 FFLTR3EN	Fault Input 3 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input filter is disabled. 1 Fault input filter is enabled.
6 FFLTR2EN	Fault Input 2 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input filter is disabled. 1 Fault input filter is enabled.
5 FFLTR1EN	Fault Input 1 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input filter is disabled. 1 Fault input filter is enabled.
4 FFLTR0EN	Fault Input 0 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input filter is disabled. 1 Fault input filter is enabled.
3 FAULT3EN	Fault Input 3 Enable Enables the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input is disabled. 1 Fault input is enabled.
2 FAULT2EN	Fault Input 2 Enable Enables the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input is disabled. 1 Fault input is enabled.
1 FAULT1EN	Fault Input 1 Enable Enables the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input is disabled. 1 Fault input is enabled.

Table continues on the next page...

FTMx_FLTCTRL field descriptions (continued)

Field	Description
0 FAULT0EN	Fault Input 0 Enable Enables the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input is disabled. 1 Fault input is enabled.

33.3.26 Quadrature Decoder Control and Status (FTMx_QDCTRL)

This register has the control and status bits for the quadrature decoder mode.

NOTE

Do not write to this register when the quadrature decoder feature is not supported.

Address: Base address + 33h offset

Bit	7	6	5	4	3	2	1	0
Read	PHAFLTREN	PHBFLTREN	PHAPOL	PHBPOL	QUADMODE	QUADIR	TOFDIR	QUADEN
Write								
Reset	0	0	0	0	0	0	0	0

FTMx_QDCTRL field descriptions

Field	Description
7 PHAFLTREN	Phase A Input Filter Enable Enables the filter for the quadrature decoder phase A input. The filter value for the phase A input is defined by the CH(n)FVAL field of FTMFILTER. The phase A filter is also disabled when CH(n)FVAL is zero. 0 Phase A input filter is disabled. 1 Phase A input filter is enabled.
6 PHBFLTREN	Phase B Input Filter Enable Enables the filter for the quadrature decoder phase B input. The filter value for the phase B input is defined by the CH(n+1)FVAL field of FTMFILTER. The phase B filter is also disabled when CH(n+1)FVAL is zero. 0 Phase B input filter is disabled. 1 Phase B input filter is enabled.
5 PHAPOL	Phase A Input Polarity Selects the polarity for the quadrature decoder phase A input. 0 Normal polarity. Phase A input signal is not inverted in FTM. 1 Inverted polarity. Phase A input signal is inverted in FTM.

Table continues on the next page...

FTMx_QDCTRL field descriptions (continued)

Field	Description
4 PHBPOL	Phase B Input Polarity Selects the polarity for the quadrature decoder phase B input. 0 Normal polarity. Phase B input signal is not inverted in FTM. 1 Inverted polarity. Phase B input signal is inverted in FTM.
3 QUADMODE	Quadrature Decoder Mode Selects the encoding mode used in the quadrature decoder mode. 0 Phase A and phase B encoding mode. 1 Count and direction encoding mode.
2 QUADIR	FTM Counter Direction in Quadrature Decoder Mode Indicates the counting direction and it is updated according to selected encoding mode. 0 Counting direction is decreasing. FTM counter decrement. 1 Counting direction is increasing. FTM counter increment.
1 TOFDIR	Timer Overflow Direction in Quadrature Decoder Mode Indicates if the TOF bit was set on the top or the bottom of counting. 0 TOF bit was set on the bottom of counting. There was an FTM counter decrement and FTM counter changes from its minimum value to its maximum value. 1 TOF bit was set on the top of counting. There was an FTM counter increment and FTM counter changes from its maximum value to its minimum value.
0 QUADEN	Quadrature Decoder Mode Enable Enables the quadrature decoder mode. In this mode, the phase A and B input signals control the FTM counter direction. The quadrature decoder mode has precedence over the other modes. (See the table Mode, Edge, and Level Selection in the description of the CnSC register.) This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Quadrature decoder mode is disabled. 1 Quadrature decoder mode is enabled.

33.4 Functional Description

The following sections describe the FTM features.

The notation used in this document to represent the counters and the generation of the signals is shown in the following figure.

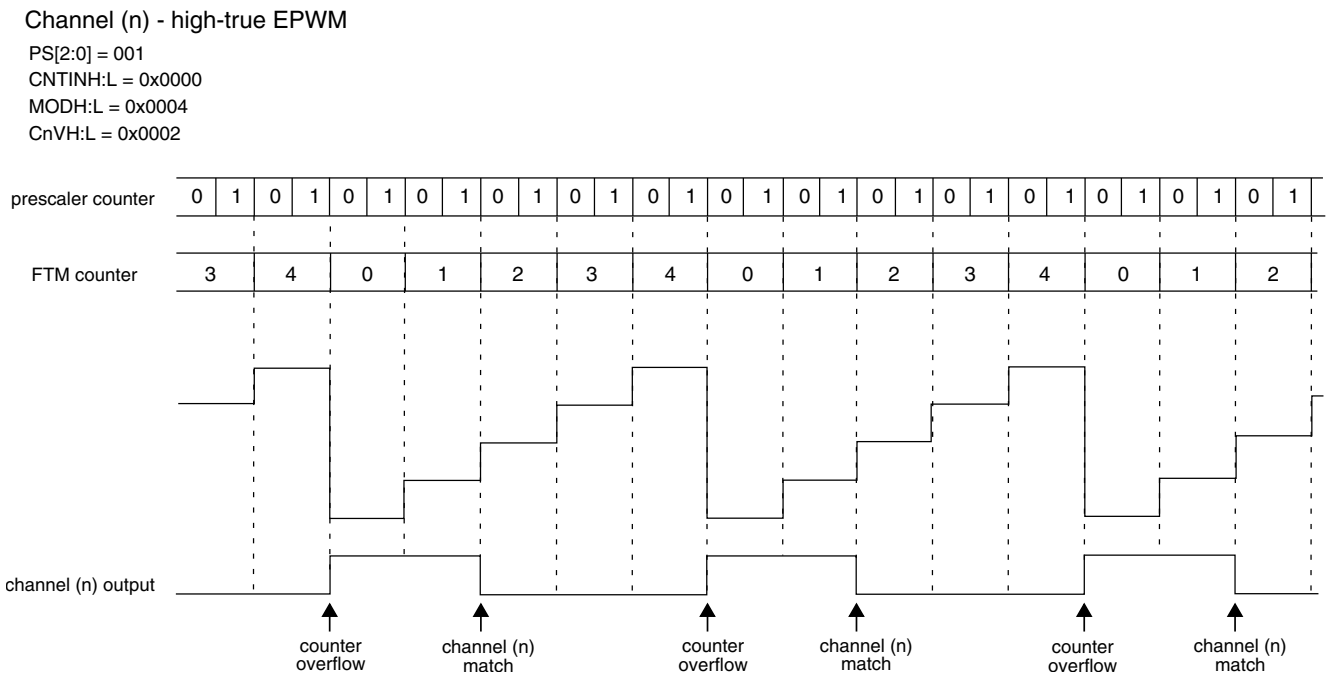


Figure 33-143. Notation used

33.4.1 Clock Source

FTM module has only one clock domain that is the system clock.

33.4.1.1 Counter Clock Source

The CLKS[1:0] bits in the SC register select one of three possible clock sources for the FTM counter or disable the FTM counter. After any MCU reset, CLKS[1:0] = 0:0 so no clock source is selected.

The CLKS[1:0] bits may be read or written at any time. Disabling the FTM counter by writing 0:0 to the CLKS[1:0] bits does not affect the FTM counter value or other registers.

The fixed frequency clock is an alternative clock source for the FTM counter that allows the selection of a clock other than the system clock or an external clock. This clock input is defined by chip integration. Refer to chip specific documentation for further information. Due to FTM hardware implementation limitations, the frequency of the fixed frequency clock must not exceed the system clock frequency.

The external clock passes through a synchronizer clocked by the system clock to ensure that counter transitions are properly aligned to system clock transitions. Therefore, to meet the Nyquist criteria and account for jitter, the frequency of the external clock source must not exceed 1/4 of the system clock frequency.

33.4.2 Prescaler

The selected counter clock source passes through a prescaler that is a 7-bit counter. The value of the prescaler is selected by the PS[2:0] bits. The following figure shows an example of the prescaler counter and FTM counter.

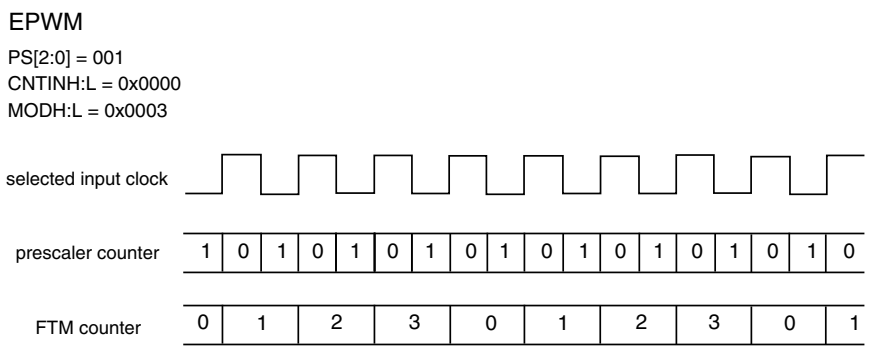


Figure 33-144. Example of the prescaler counter

33.4.3 Counter

The FTM has a 16-bit counter that is used by the channels either for input or output modes. The FTM counter clock is the selected clock divided by the prescaler (see Prescaler).

The FTM counter has these modes of operation:

- up counting (see Up counting)
- up-down counting (see Up-down counting)
- quadrature mode, if it is supported (see Quadrature Decoder Mode)

33.4.3.1 Up counting

Up counting is selected when (CPWMS = 0) and, if the quadrature decoder feature is supported, when (QUADEN = 0).

CNTINH:L defines the starting value of the count and MODH:L defines the final value of the count; see the following figure. The value of CNTINH:L is loaded into the FTM counter, and the counter increments until the value of MODH:L is reached, at which point the counter is reloaded with CNTINH:L.

The FTM period when using up counting is (MODH:L – CNTINH:L + 0x0001) × period of the FTM counter clock.

The TOF bit is set when the FTM counter changes from MODH:L to CNTINH:L.

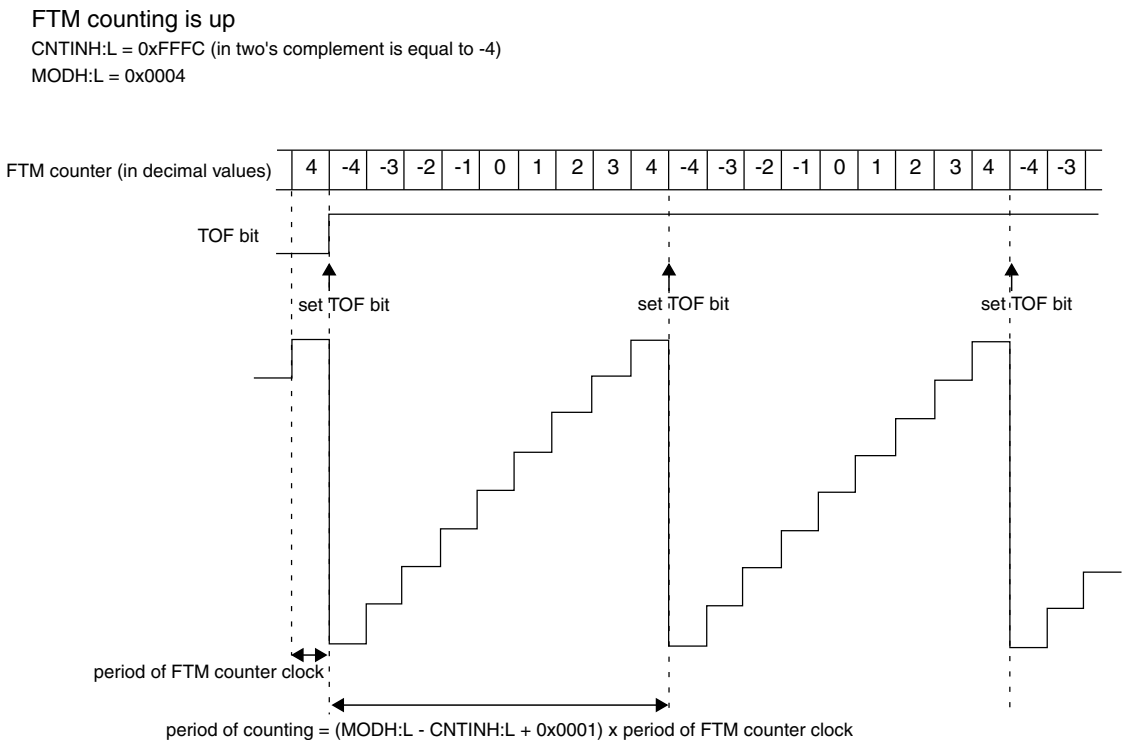


Figure 33-145. Example of FTM up and signed counting

If (CNTINH:L = 0x0000), the FTM counting is equivalent to TPM up counting; that is, up and unsigned counting. See the following figure. If (CNTINH[7] = 1), then the initial value of the FTM counter is a negative number in two's complement format, so the FTM counting is up and signed. Conversely, if (CNTINH[7] = 0 and CNTINH:L ≠ 0x0000), then the initial value of the FTM counter is a positive number, therefore the FTM counting is up and unsigned.

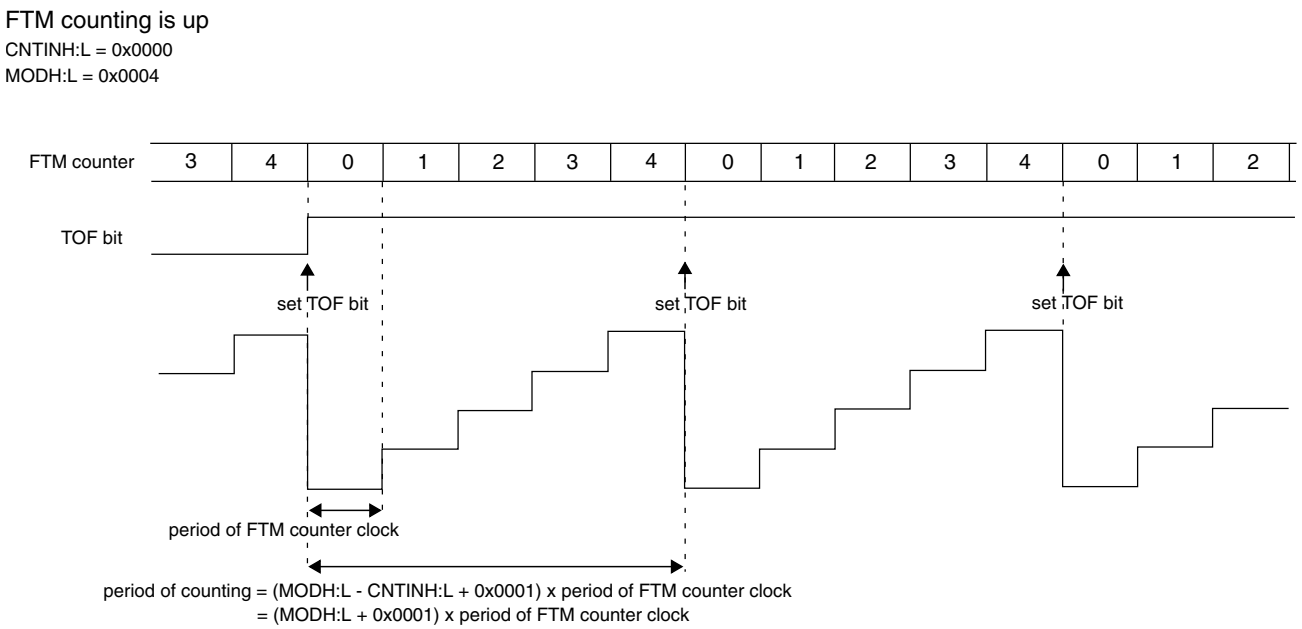


Figure 33-146. Example of FTM up counting with CNTIN = 0x0000

Note

- FTM operation is valid only when the value of the CNTINH:L registers is less than the value of the MODH:L registers, either in the unsigned counting or signed counting.. Software must ensure that the values in the CNTINH:L and MODH:L registers meet this requirement. Any values of CNTINH:L and MODH:L that do not satisfy this criteria can result in unpredictable behavior.
- MODH:L = CNTINH:L is a redundant condition. In this case, the FTM counter is always equal to MODH:L and the TOF bit is set in each rising edge of the FTM counter clock.
- When MODH:L = 0x0000, CNTINH:L = 0x0000 (for example after reset), and FTMEN = 1, the FTM counter remains stopped at 0x0000 until a non-zero value is written into the MODH:L or CNTINH:L registers.
- Setting CNTINH:L to be greater than the value of MODH:L is not recommended as this unusual setting may make the FTM operation difficult to comprehend. However, there is no restriction on this configuration, and an example is shown in the following figure.

FTM counting is up
MODH:L = 0x0005
CNTINH:L = 0x0015

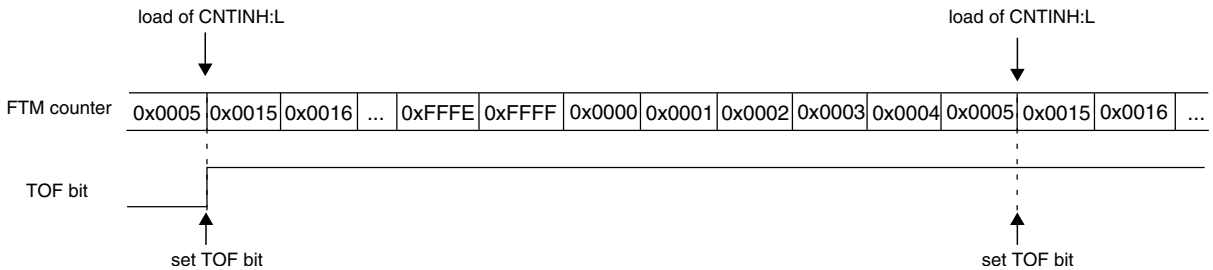


Figure 33-147. Example of up counting when the value of CNTIN registers is greater than the value of MOD registers

33.4.3.2 Up-down counting

Up-down counting is selected when (CPWMS = 1) and, if the quadrature decoder feature is supported, when (QUADEN = 0).

CNTINH:L defines the starting value of the count and MODH:L defines the final value of the count. The value of CNTINH:L is loaded into the FTM counter, and the counter increments until the value of MODH:L is reached, at which point the counter is decremented until it returns to the value of CNTINH:L and the up-down counting restarts.

The FTM period when using up-down counting is 2 × (MODH:L – CNTINH:L) × period of the FTM counter clock.

The TOF bit is set when the FTM counter changes from MODH:L to (MODH:L – 1).

If (CNTINH:L = 0x0000), the FTM counting is equivalent to TPM up-down counting; that is, up-down and unsigned counting. See the following figure.

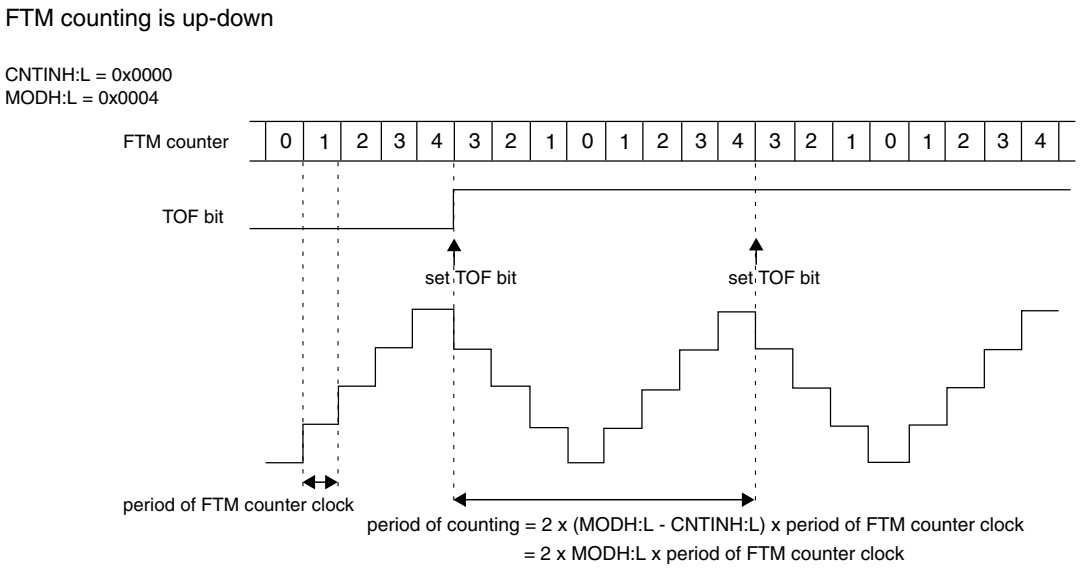


Figure 33-148. Example of up-down counting when CNTIN = 0x0000

Note

- The up-down counting is available only when (CNTINH:L = 0x0000).
- The configuration with (CNTINH:L ≠ 0x0000) when (CPWMS = 1) is not recommended and its results are not guaranteed.

33.4.3.3 Free running counter

If (FTMEN = 0) and (MODH:L = 0x0000 or MODH:L = 0xFFFF), the FTM counter is a free running counter. In this case, the FTM counter runs free from 0x0000 through 0xFFFF and the TOF bit is set when the FTM counter changes from 0xFFFF to 0x0000 See the following figure.

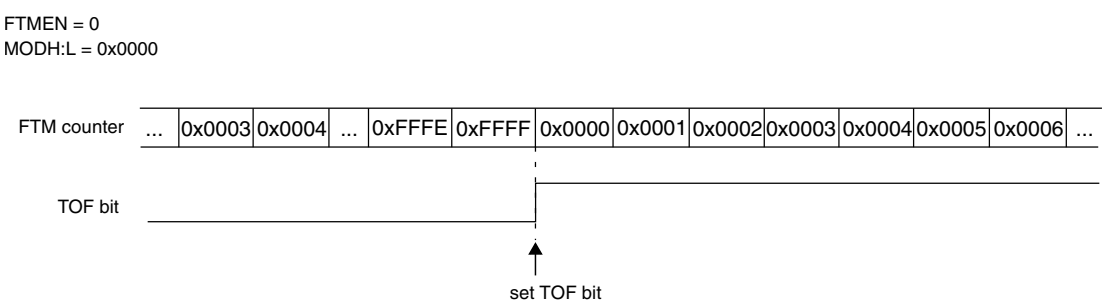


Figure 33-149. Example when the FTM counter is a free running

The FTM counter is also a free running counter when all of the following apply:

- (FTMEN = 1)

- (CPWMS = 0)
- (CNTINH:L = 0x0000)
- (MODH:L = 0xFFFF)
- (QUADEN = 0) if the quadrature decoder feature is supported

In this case, the FTM counter runs free from 0x0000 through 0xFFFF and the TOF bit is set when the FTM counter changes from 0xFFFF to 0x0000.

33.4.3.4 Counter reset

Any write to CNTH or CNTL register resets the FTM counter to the value of CNTINH:L and the channels output to its initial value, except for channels in output compare mode.

The FTM counter synchronization can also be used to force the value of CNTINH:L into the FTM counter and the channels output to its initial value, except for channels in output compare mode.

33.4.4 Input capture mode

The input capture mode is selected when (DECAPEN = 0), (COMBINE = 0), (CPWMS = 0), (MSnB:MSnA = 0:0), and (ELSnB:ELSnA ≠ 0:0).

When a selected edge occurs on the channel input, the current value of the FTM counter is captured into the CnVH:L registers. 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 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 system clock divided by four, which is required to meet Nyquist criteria for signal sampling.

When either half of the 16-bit capture register (CnVH:L) is read, the other half is latched into a buffer to support coherent 16-bit access in big-endian or little-endian order. This read coherency mechanism can be manually reset by writing to CnSC register.

Writes to the CnVH:L registers are ignored in input capture mode.

While in BDM, the input capture function works as configured. When a selected edge event occurs, the FTM counter value, which is frozen because of BDM, is captured into the CnVH:L registers and the CHnF bit is set.

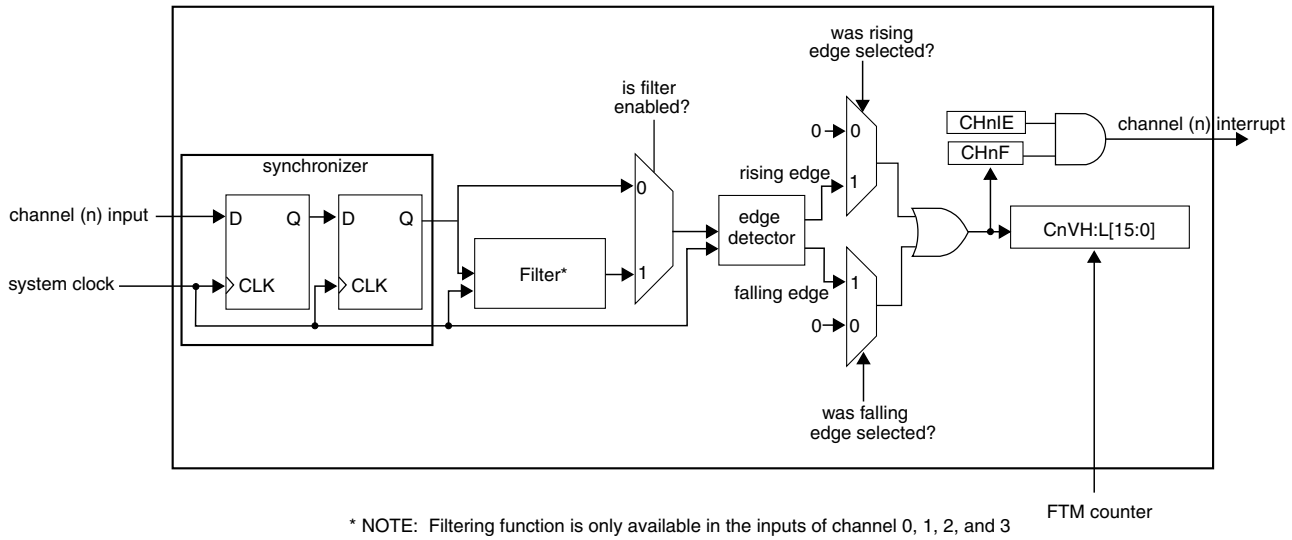


Figure 33-150. Input capture mode

If the channel input does not have a filter enabled, then the input signal is always delayed three rising edges of the system clock; that is, two rising edges to the synchronizer plus one more rising edge to the edge detector. In other words, the CHnF bit is set on the third rising edge of the system clock after a valid edge occurs on the channel input.

Note

- Input capture mode is available only with (CNTINH:L = 0x0000).
- Input capture mode with (CNTINH:L ≠ 0x0000) is not recommended and its results are not guaranteed.

33.4.4.1 Filter for input capture mode

The filter function is available only on channels 0, 1, 2, and 3.

Firstly, the input signal is synchronized by the system clock. Following synchronization, the input signal enters the filter block; see the following figure. When there is a state change in the input signal, the 5-bit counter is reset and starts counting up. As long as the new state is stable on the input, the counter continues to increment. If the 5-bit counter overflows (the counter exceeds the value of the CHnFVAL[3:0] bits), the state change of the input signal is validated. It is then transmitted as a pulse edge to the edge detector.

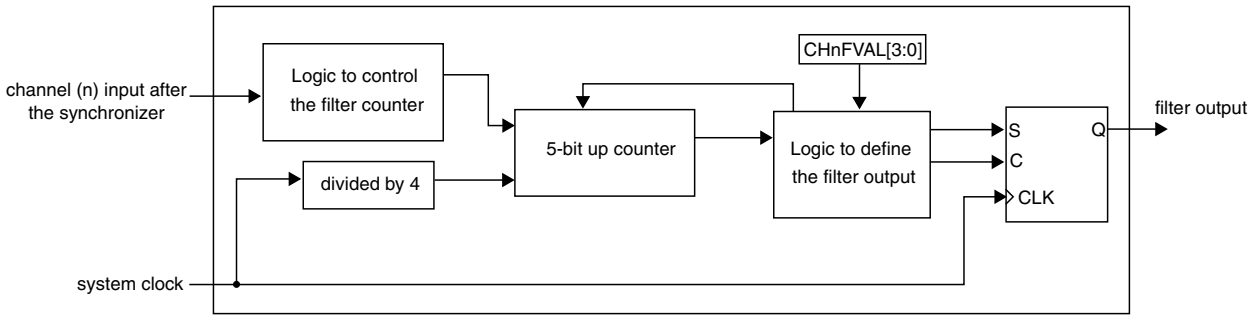


Figure 33-151. Channel input filter

If the opposite edge appears on the input signal before validation, the counter is reset. At the next input transition, the counter starts counting again. Any pulse shorter than the minimum valid width (CHnFVAL[3:0] bits × 4 system clocks) is regarded as a glitch and is not passed on to the edge detector. A timing diagram of the input filter is shown in the following figure.

The filter function is disabled when CHnFVAL[3:0] bits are zero. In this case, the input signal is delayed three rising edges of the system clock. If (CHnFVAL[3:0] ≠ 0000), then the input signal is delayed by the minimum pulse width (CHnFVAL[3:0] × 4 system clocks) plus a further four rising edges of the system clock (two rising edges to the synchronizer, one rising edge to the filter output plus one more to the edge detector). In other words, CHnF is set (4 + 4 × CHnFVAL[3:0]) system clock periods after a valid edge occurs on the channel input.

The clock for the 5-bit counter in the channel input filter is the system clock divided by 4.

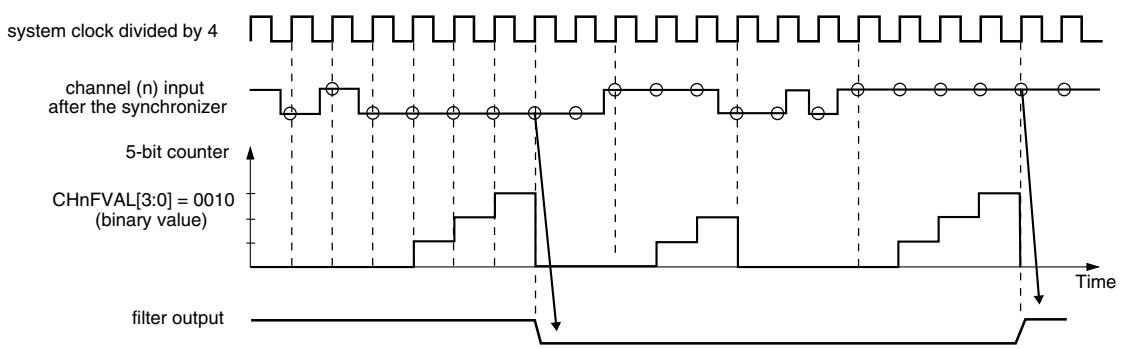


Figure 33-152. Channel input filter example

33.4.5 Output compare mode

The output compare mode is selected when (DECAPEN = 0), (COMBINE = 0), (CPWMS = 0) and (MSnB:MSnA = 0:1).

In output compare mode, the FTM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CnVH:CnVL registers of an output compare channel, the channel (n) output can be set, cleared, or toggled.

When a channel is initially configured to toggle mode, the previous value of the channel output is held until the first output compare event occurs.

The CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (FTM counter = CnVH:CnVL).

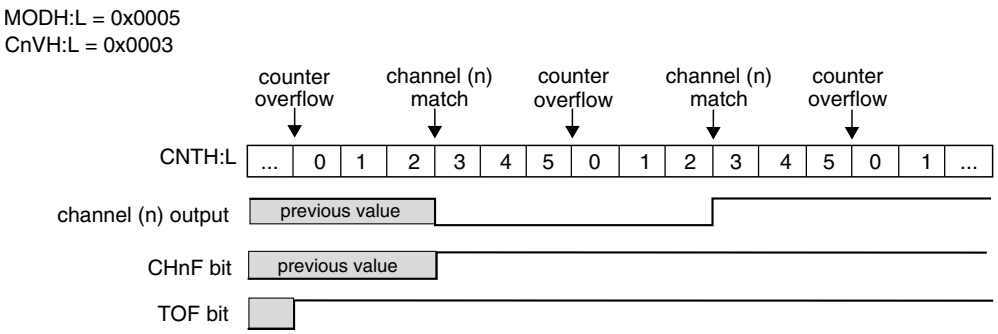


Figure 33-153. Example of the output compare mode when the match toggles the channel output

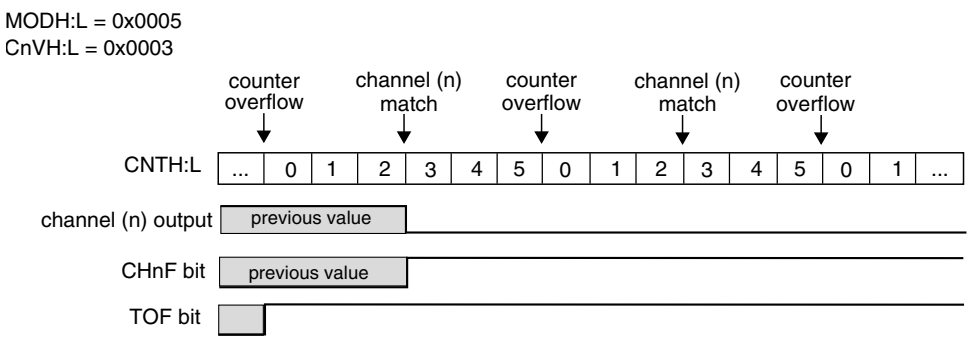


Figure 33-154. Example of the output compare mode when the match clears the channel output

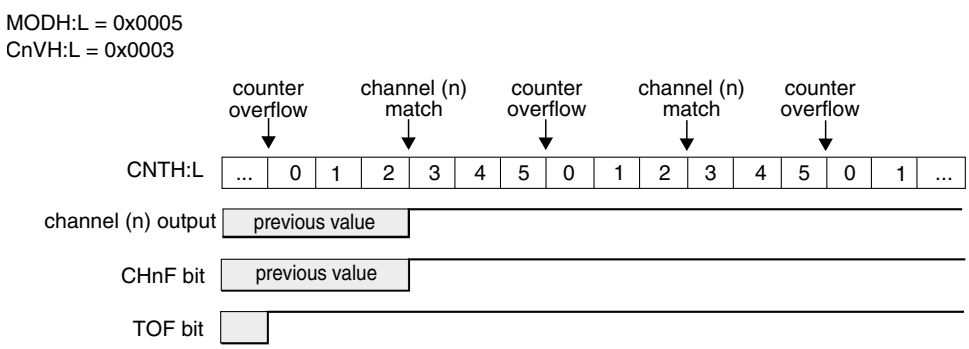


Figure 33-155. Example of the output compare mode when the match sets the channel output

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 CnVH:CnVL registers, the CHnF bit is set and the channel (n) interrupt is generated, if CHnIE = 1. However, the channel (n) output is not modified and controlled by FTM.

Note

- Output compare mode is available only with (CNTINH:CNTINL = 0x0000).
- Output compare mode with (CNTINH:CNTINL ≠ 0x0000) is not recommended and its results are not guaranteed.

33.4.6 Edge-aligned PWM (EPWM) mode

The edge-aligned mode is selected when all of the following apply:

- (DECAPEN = 0)
- (COMBINE = 0)
- (CPWMS = 0)
- (MSnB = 1)
- (QUADEN = 0) if the quadrature decoder feature is supported

The EPWM period is determined by (MODH:L – CNTINH:L + 0x0001) and the pulse width (duty cycle) is determined by (CnVH:L – CNTINH:L).

The CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (FTM counter = CnVH:L), that is, at the end of the pulse width.

This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within an FTM.

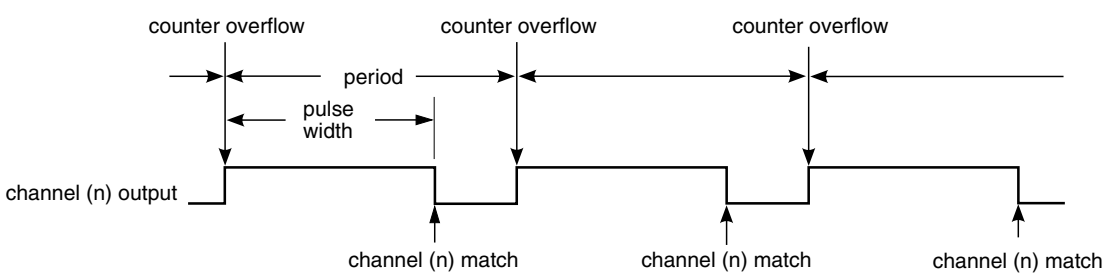


Figure 33-156. EPWM period and pulse width with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = 0:0) when the counter reaches the value in the CnVH:L registers, the CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1), however, the channel (n) output is not controlled by FTM.

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced high at the counter overflow, when the value of CNTINH:L is loaded into the FTM counter. Additionally, it is forced low at the channel (n) match, when the FTM counter = CnVH:L. See the following figure.

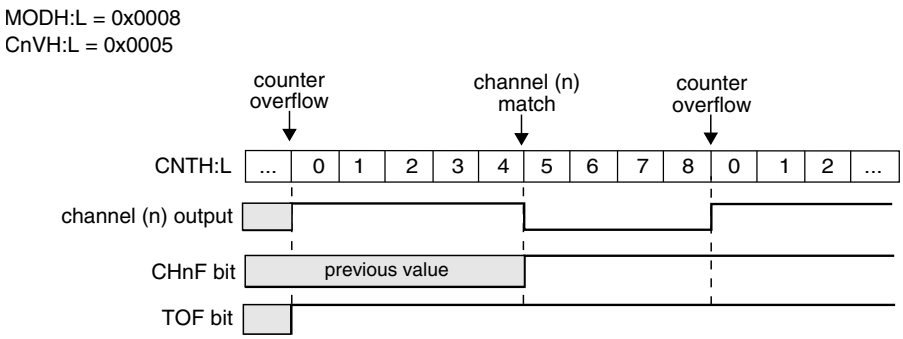


Figure 33-157. EPWM signal with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced low at the counter overflow, when the value of CNTINH:L is loaded into the FTM counter. Additionally, it is forced high at the channel (n) match, when the FTM counter = CnVH:L. See the following figure.

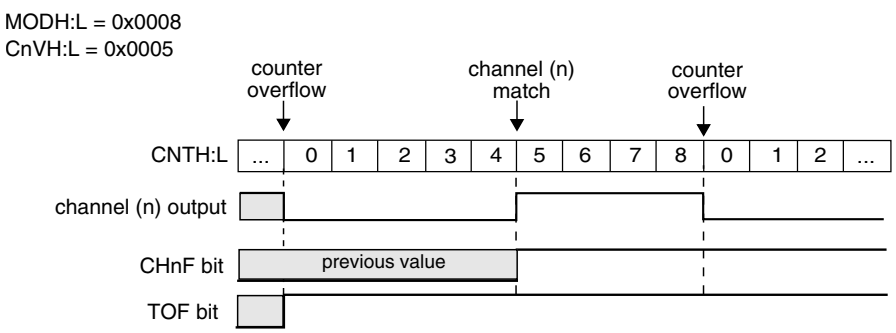


Figure 33-158. EPWM signal with ELSnB:ELSnA = X:1

If (CnVH:L = 0x0000), then the channel (n) output is a 0% duty cycle EPWM signal and CHnF bit is not set, even when there is the channel (n) match. If (CnVH:L > MODH:L), then the channel (n) output is a 100% duty cycle EPWM signal and CHnF bit is not set, even when there is the channel (n) match. Therefore, MODH:MODL must be less than 0xFFFF in order to get a 100% duty cycle EPWM signal.

Note

- EPWM mode is available only with (CNTINH:L = 0x0000).
- EPWM mode with (CNTINH:L ≠ 0x0000) is not recommended and its results are not guaranteed.

33.4.7 Center-aligned PWM (CPWM) mode

The center-aligned mode is selected when all of the following apply:

- (QUADEN = 0) if the quadrature decoder feature is supported
- (DECAPEN = 0)
- (COMBINE = 0)
- (CPWMS = 1)

The CPWM pulse width (duty cycle) is determined by 2 × (CnVH:L – CNTINH:L). The period is determined by 2 × (MODH:L – CNTINH:L). See the following figure. MODH:L must be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results.

In the CPWM mode, the FTM counter counts up until it reaches MODH:L and then counts down until it reaches the value of CNTINH:L.

The CHnF bit is set and channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (FTM counter = CnVH:L) when the FTM counting is down, at the begin of the pulse width, and when the FTM counting is up, at the end of the pulse width.

This type of PWM signal is called center-aligned because the pulse width centers for all channels are aligned with the value of CNTINH:L.

The other channel modes are not compatible with the up-down counter (CPWMS = 1). Therefore, all FTM channels must be used in CPWM mode when (CPWMS = 1).

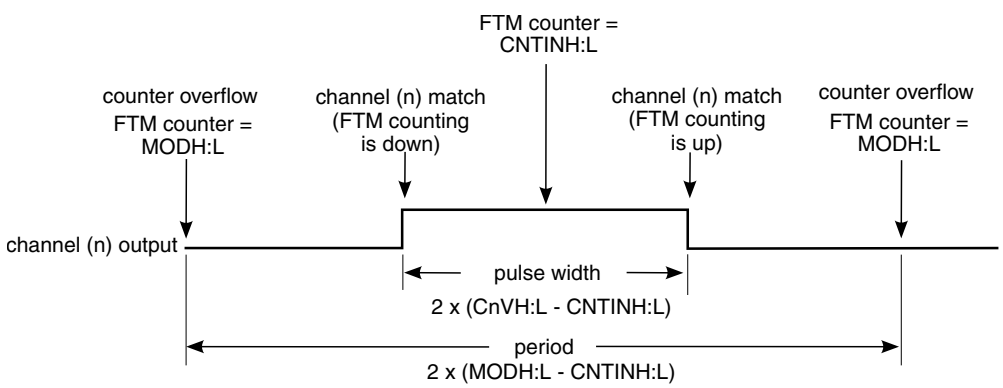


Figure 33-159. CPWM period and pulse width with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = 0:0) when the counter reaches the value in the CnVH:L registers, the CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1), however the channel (n) output is not controlled by FTM.

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced high at the channel (n) match (FTM counter = CnVH:L) when counting down, and it is forced low at the channel (n) match when counting up; see the following figure.

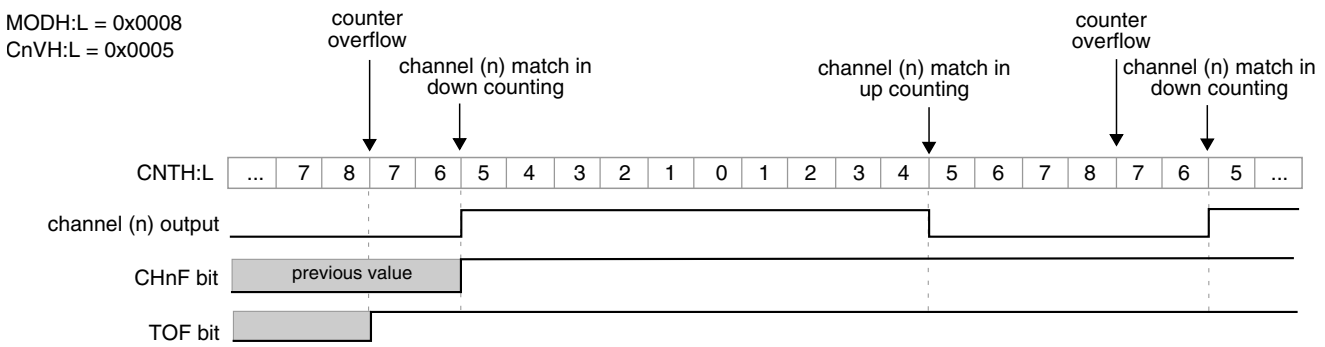


Figure 33-160. CPWM signal with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced low at the channel (n) match (FTM counter = CnVH:L) when counting down, and it is forced high at the channel (n) match when counting up; see the following figure.

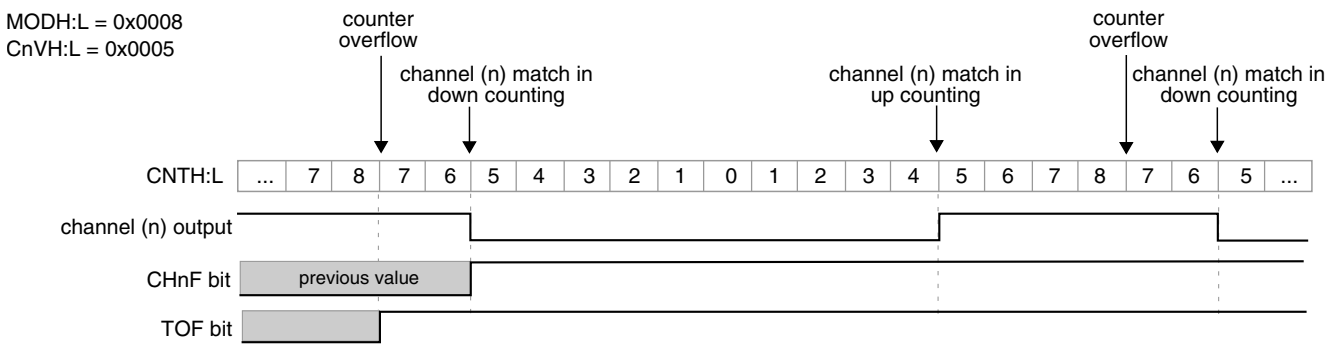


Figure 33-161. CPWM signal with ELSnB:ELSnA = X:1

If (CnVH:L = 0x0000) or (CnVH:L is a negative value, that is, CnVH[7] = 1) then the channel (n) output is a 0% duty cycle CPWM signal and CHnF bit is not set even when there is the channel (n) match.

If (CnVH:L is a positive value, that is, CnVH[7] = 0), (CnVH:L ≥ MODH:L), and (MODH:L ≠ 0x0000), then the channel (n) output is a 100% duty cycle CPWM signal and CHnF bit is not set even when there is the channel (n) match. This implies that the usable range of periods set by MODH:L is 0x0001 through 0x7FFE, or 0x7FFF if you do not need to generate a 100% duty cycle CPWM signal. This is not a significant limitation because the resulting period is much longer than required for normal applications.

The CPWM mode must not be used when the FTM counter is a free running counter.

Note

- CPWM mode is available only with (CNTINH:L = 0x0000).
- CPWM mode with (CNTINH:L ≠ 0x0000) is not recommended and its results are not guaranteed.

33.4.8 Combine mode

The combine mode is selected when all of the following apply:

- (FTMEN = 1)
- (QUADEN = 0) if the quadrature decoder feature is supported
- (DECAPEN = 0)
- (COMBINE = 1)
- (CPWMS = 0)

In combine mode, the 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 (MODH:L – CNTINH:L + 0x0001) and the PWM pulse width (duty cycle) is determined by (|C(n+1)VH:L – C(n)VH:L|).

The CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (FTM counter = C(n)VH:L). The CH(n+1)F bit is set and the channel (n+1) interrupt is generated (if CH(n+1)IE = 1) at the channel (n+1) match (FTM counter = C(n+1)VH:C(n+1)VL).

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced low at the beginning of the period (FTM counter = CNTINH:L) and at the channel (n+1) match (FTM counter = C(n+1)VH:L). It is forced high at the channel (n) match (FTM counter = C(n)VH:L). See the following figure.

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced high at the beginning of the period (FTM counter = CNTINH:L) and at the channel (n+1) match (FTM counter = C(n+1)VH:L). It is forced low at the channel (n) match (FTM counter = C(n)VH:L). See the following figure.

In combine mode, the ELS(n+1)B and ELS(n+1)A bits are not used in the generation of the channels (n) and (n+1) output.

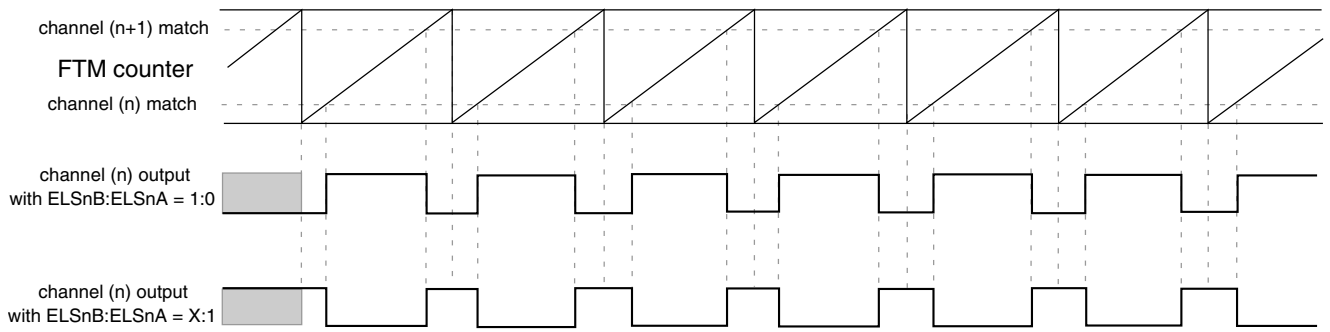


Figure 33-162. Combine mode

The following figures illustrate the generation of PWM signals using combine mode.

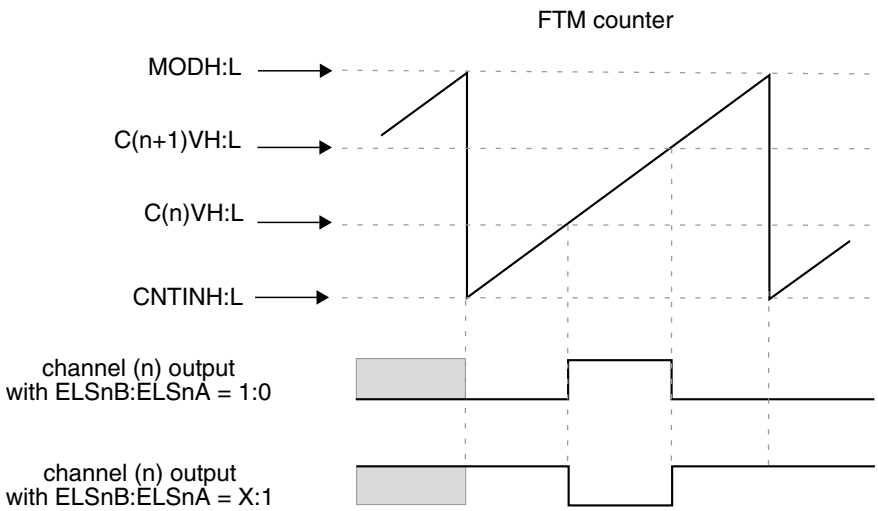


Figure 33-163. Channel (n) output if (CNTIN < C(n)V < MOD) and (CNTIN < C(n+1)V < MOD) and (C(n)V < C(n+1)V)

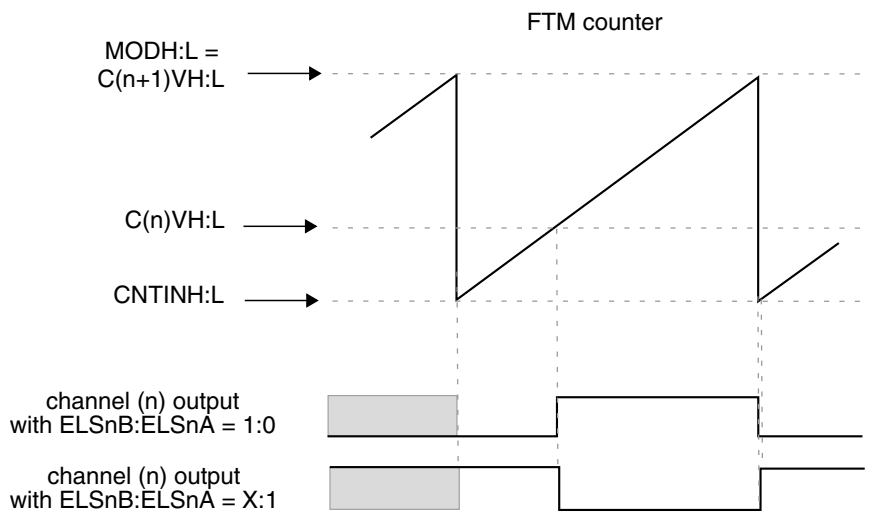


Figure 33-164. Channel (n) output if (CNTIN < C(n)V < MOD) and (C(n+1)V = MOD)

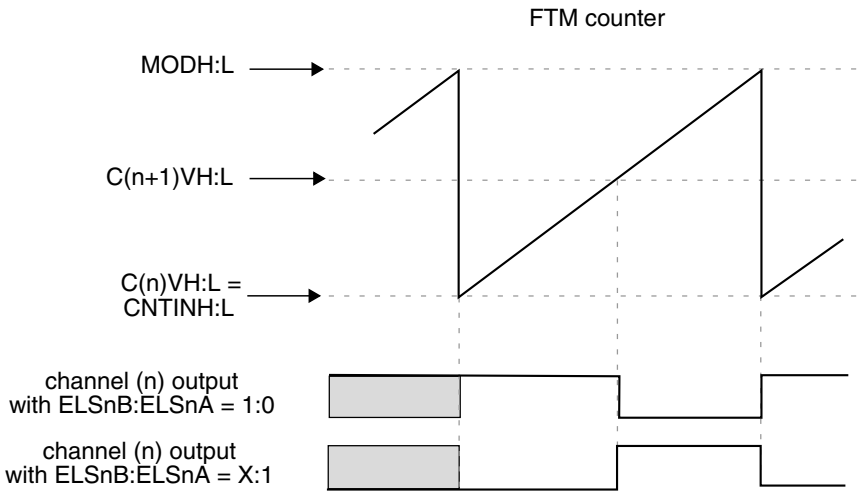


Figure 33-165. Channel (n) output if (C(n)V = CNTIN) and (CNTIN < C(n+1)V < MOD)

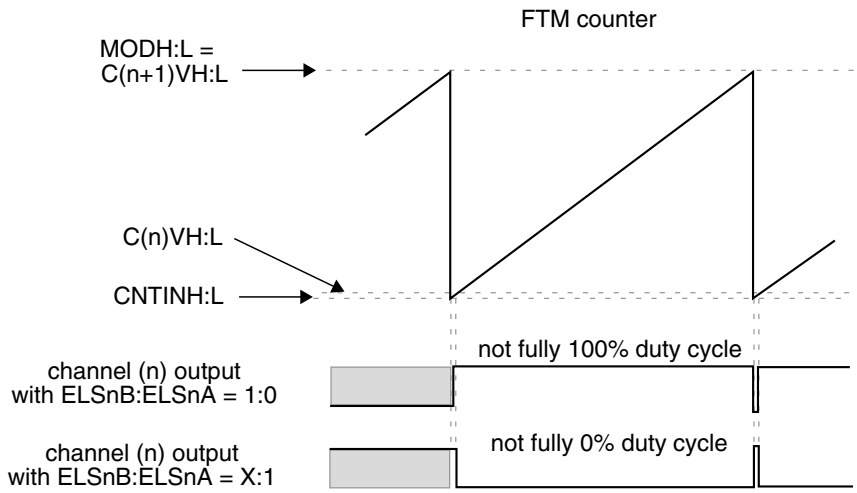


Figure 33-166. Channel (n) output if (CNTIN < C(n)V < MOD) and (C(n)V is almost equal to CNTIN) and (C(n+1)V = MOD)

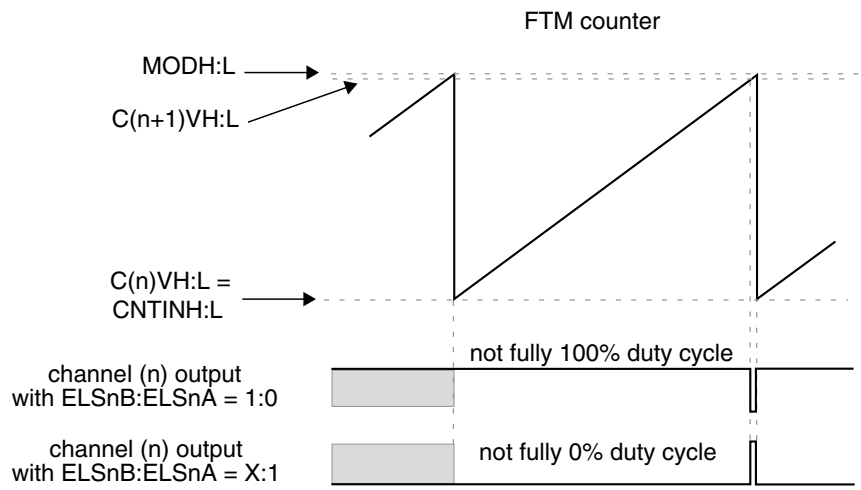


Figure 33-167. Channel (n) output if (C(n)V = CNTIN) and (CNTIN < C(n+1)V < MOD) and (C(n+1)V is almost equal to MOD)

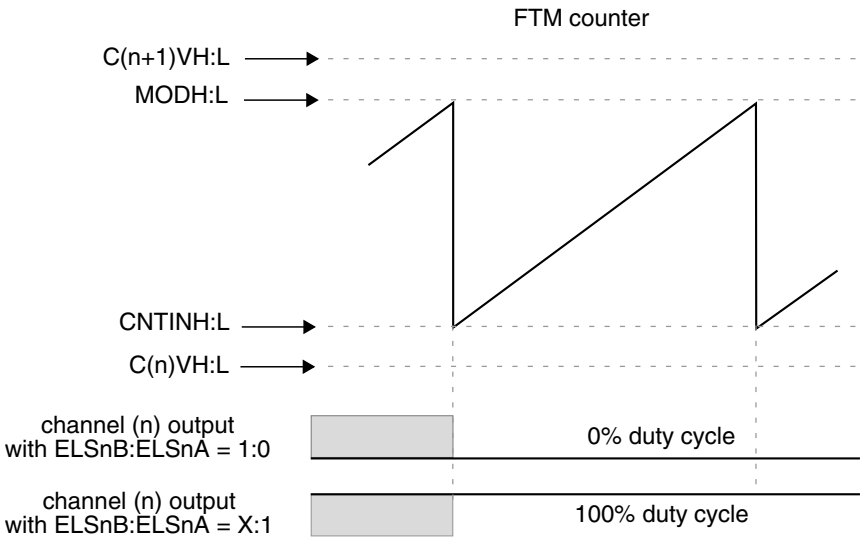


Figure 33-168. Channel (n) output if C(n)V and C(n+1)V are not between CNTIN and MOD

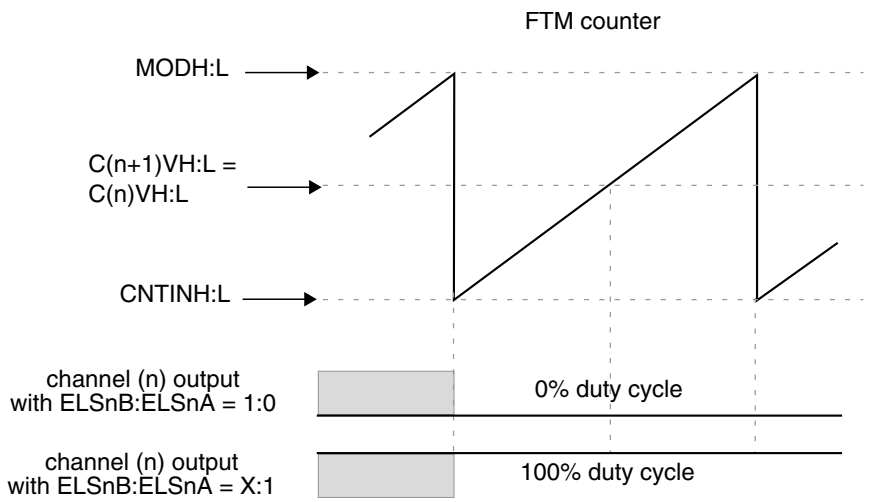


Figure 33-169. Channel (n) output if (CNTIN < C(n)V < MOD) and (CNTIN < C(n+1)V < MOD) and (CnV = C(n+1)V)

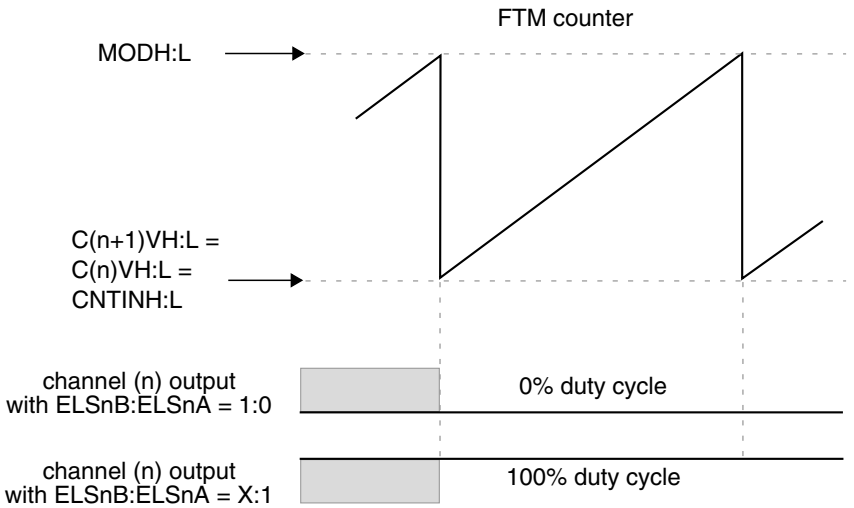


Figure 33-170. Channel (n) output if (C(n)V = C(n+1)V = CNTIN)

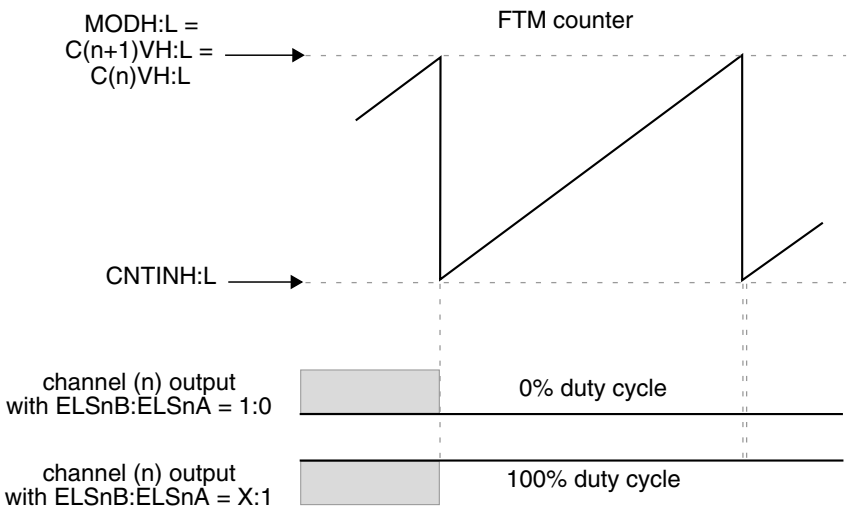


Figure 33-171. Channel (n) output if (C(n)V = C(n+1)V = MOD)

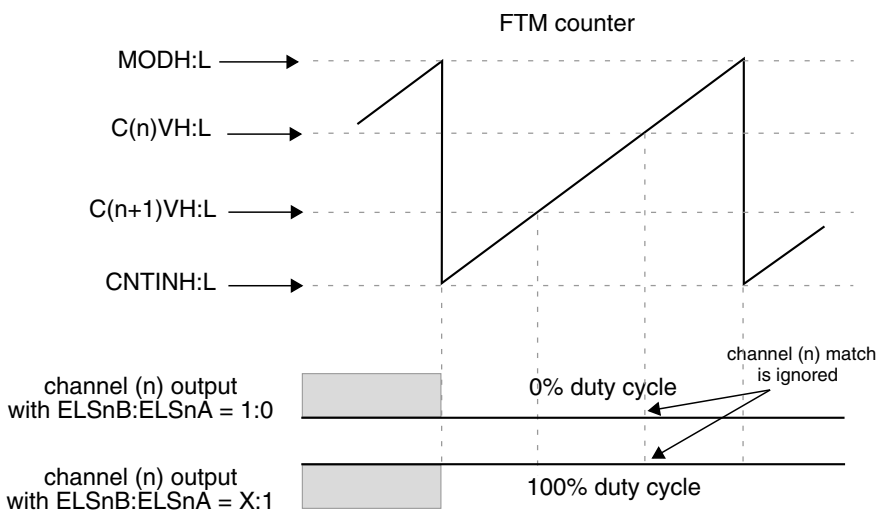


Figure 33-172. Channel (n) output if (CNTIN < C(n)V < MOD) and (CNTIN < C(n+1)V < MOD) and (C(n)V > C(n+1)V)

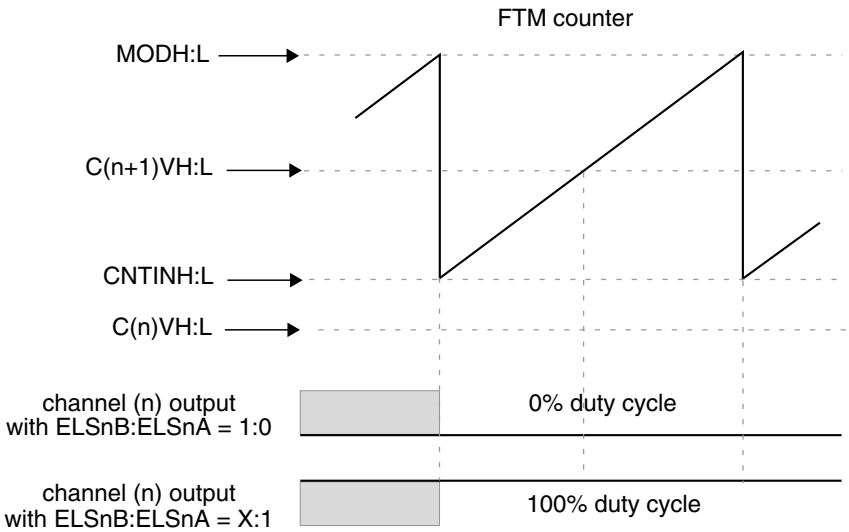


Figure 33-173. Channel (n) output if (C(n)V < CNTIN) and (CNTIN < C(n+1)V < MOD)

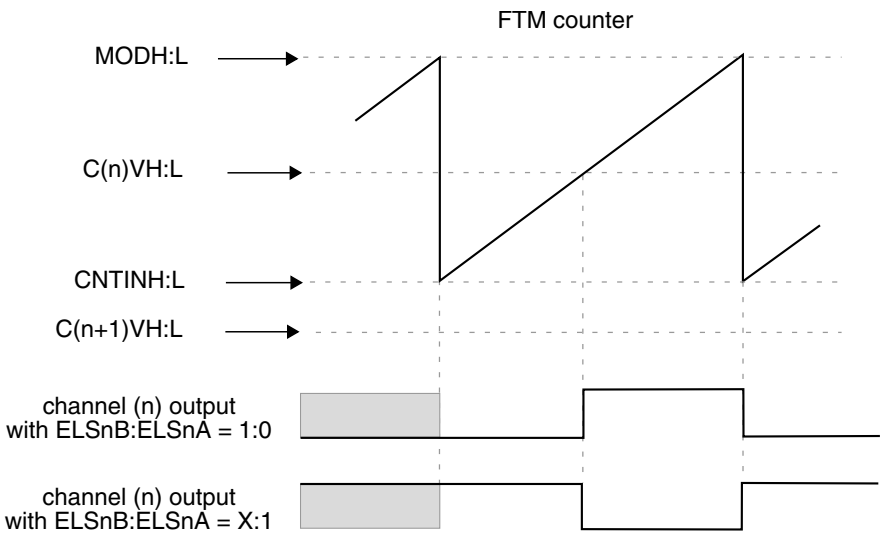


Figure 33-174. Channel (n) output if (C(n+1)V < CNTIN) and (CNTIN < C(n)V < MOD)

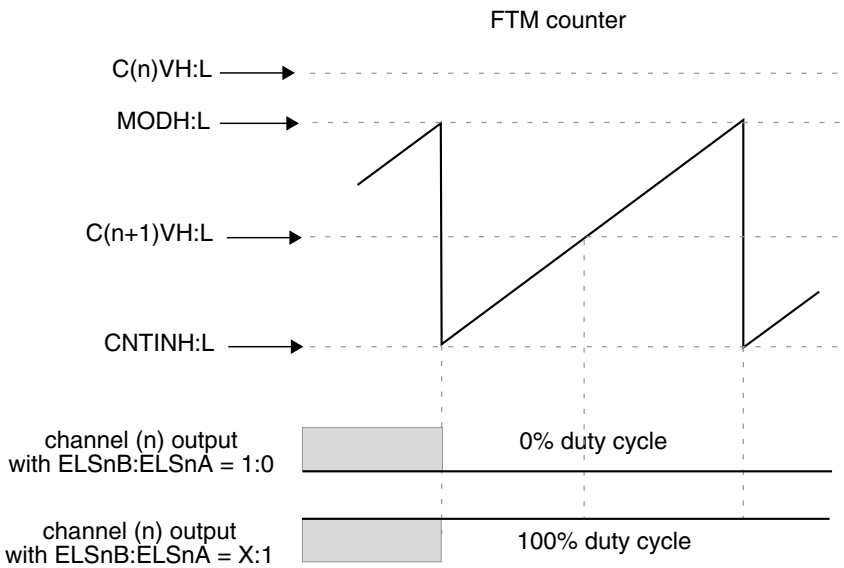


Figure 33-175. Channel (n) output if (C(n)V > MOD) and (CNTIN < C(n+1)V < MOD)

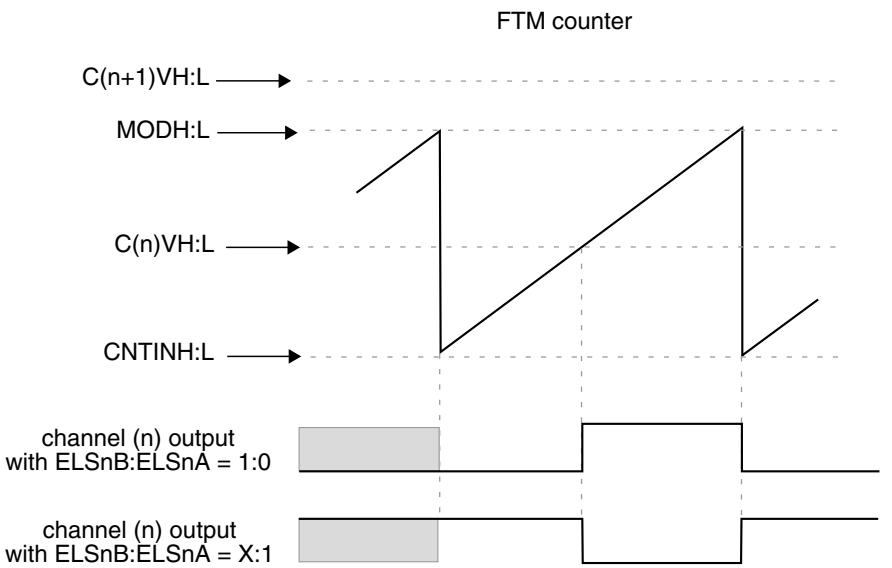


Figure 33-176. Channel (n) output if (C(n+1)V > MOD) and (CNTIN < C(n)V < MOD)

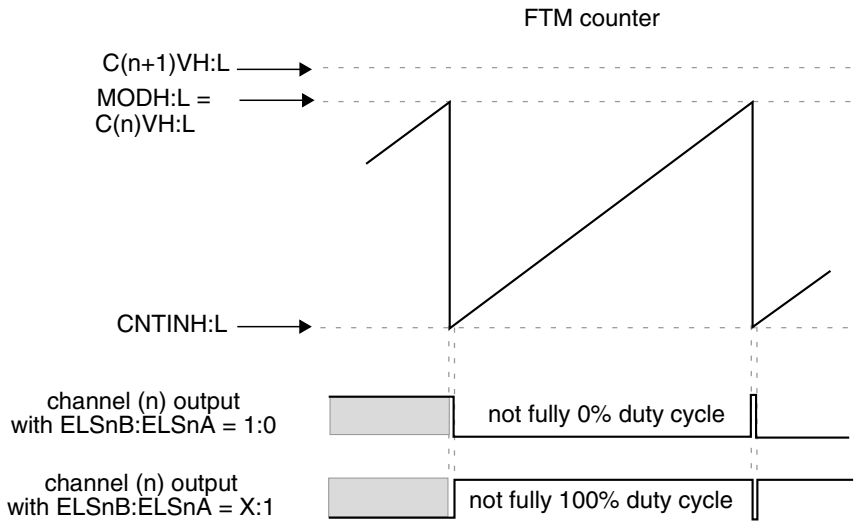


Figure 33-177. Channel (n) output if (C(n+1)V > MOD) and (CNTIN < C(n)V = MOD)

33.4.8.1 Asymmetrical PWM

In the combine mode, the control of the PWM signal first edge (when the channel (n) match occurs, that is, FTM counter = C(n)VH:L) is independent of the control of the PWM signal second edge (when the channel (n+1) match occurs, that is, FTM counter = C(n+1)VH:L). So, the combine mode allows to generate asymmetrical PWM signals.

33.4.9 Complementary mode

The complementary mode is selected when all of the following apply:

- (FTMEN = 1)
- (QUADEN = 0) if the quadrature decoder feature is supported
- (DECAPEN = 0)
- (COMBINE = 1)
- (CPWMS = 0)
- (COMP = 1)

In complementary mode the channel (n+1) output is the inverse of the channel (n) output.

The channel (n+1) output is the same as the channel (n) output if all of the following apply:

- (FTMEN = 1)
- (QUADEN = 0) if the quadrature decoder feature is supported
- (DECAPEN = 0)
- (COMBINE = 1)

- (CPWMS = 0)
- (COMP = 0)

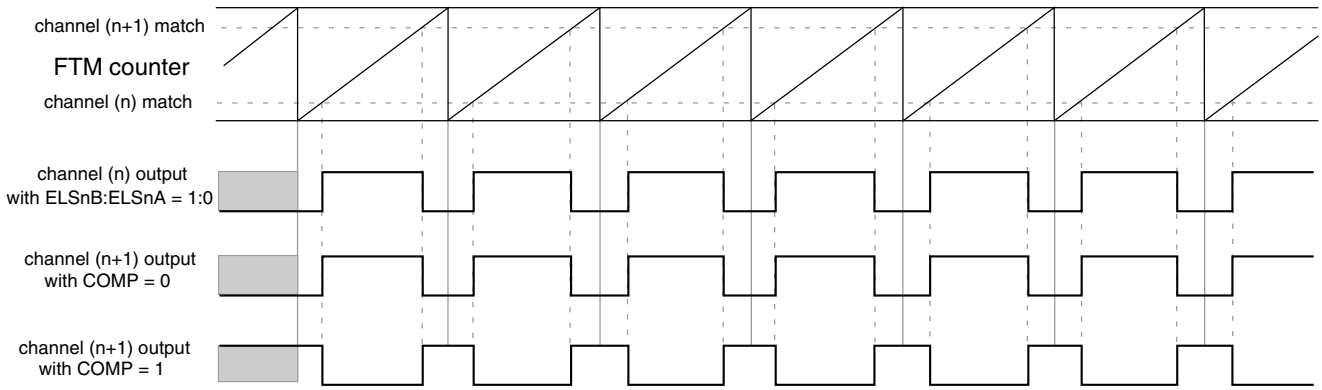


Figure 33-178. Channel (n+1) output in complementary mode with (ELSnB:ELSnA = 1:0)

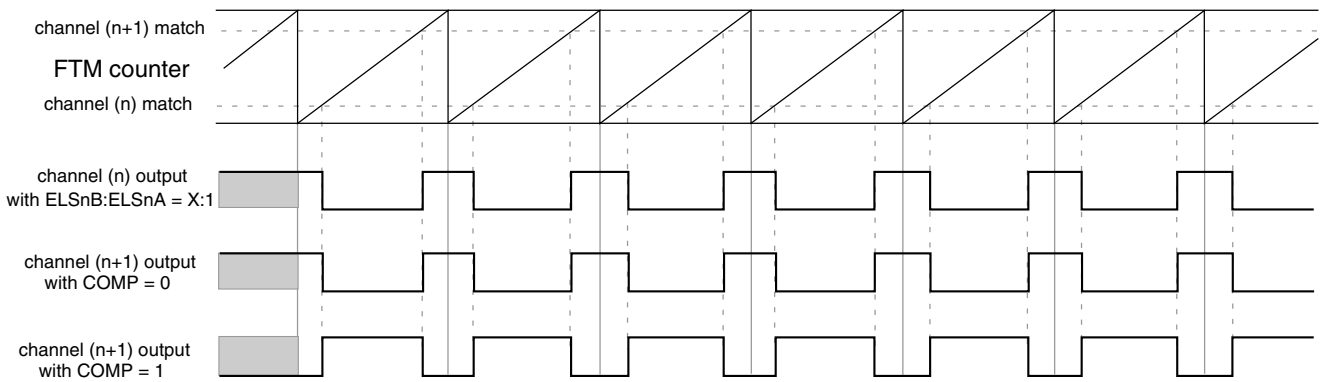


Figure 33-179. Channel (n+1) output in complementary mode with (ELSnB:ELSnA = X:1)

33.4.10 Update of the registers with write buffers

This section describes the updating of registers that have write buffers.

33.4.10.1 CNTINH:L registers

CNTINH:L registers are always updated with their write buffer after both bytes have been written.

33.4.10.2 MODH:L registers

If (CLKS[1:0] = 0:0), then MODH:L registers are updated when their second byte is written, independent of FTMEN bit.

If (CLKS[1:0] ≠ 0:0 and FTMEN = 0), then MODH:L registers are updated according to the CPWMS bit:

- If the selected mode is not CPWM mode, then MODH:L registers are updated after both bytes have been written and the FTM counter changes from (MODH:L) to (CNTINH:L). If the FTM counter is a free-running counter, then this update is made when the FTM counter changes from 0xFFFF to 0x0000.
- If the selected mode is CPWM mode, then MODH:L registers are updated after both bytes have been written and the FTM counter changes from MODH:L to (MODH:L – 0x0001).

If (CLKS[1:0] ≠ 0:0 and FTMEN = 1), then MODH:L registers are updated by PWM synchronization. See MODH:L registers synchronization.

33.4.10.3 CnVH:L registers

If (CLKS[1:0] = 0:0), then CnVH:L registers are updated when their second byte is written, independent of FTMEN bit.

If (CLKS[1:0] ≠ 0:0 and FTMEN = 0), then CnVH:L registers are updated according to the selected mode:

- If the selected mode is output compare mode, then CnVH:L registers are updated after their second byte is written and on the next change of the FTM counter.
- If the selected mode is EPWM mode, the CnVH:L registers are updated after both bytes have been written and the FTM counter changes from MODH:L to CNTINH:L. If the FTM counter is a free running counter, then this update is made when the FTM counter changes from 0xFFFF to 0x0000.
- If the selected mode is CPWM mode, then CnVH:L registers are updated after both bytes have been written and the FTM counter changes from MODH:L to (MODH:L – 0x0001).

If (CLKS[1:0] ≠ 0:0 and FTMEN = 1), then CnVH:L registers are updated according to the selected mode:

- If the selected mode is output compare mode, then CnVH:L registers are updated according to the SYNCEN bit. If (SYNCEN = 0), then CnVH:L registers are updated after their second byte is written and on the next change of the FTM counter. If (SYNCEN = 1), then CnVH:L registers are updated by PWM synchronization. See CnVH:L registers synchronization.
- If the selected mode is not output compare mode and (SYNCEN = 1), then CnVH:L registers are updated by PWM synchronization. See CnVH:L registers synchronization.

33.4.11 PWM synchronization

PWM synchronization provides an opportunity to update registers with the contents of their write buffers. It can also be used to synchronize two or more FlexTimer modules on the same MCU.

PWM synchronization updates the MODH:L and CnVH:L registers with their write buffers. It is also possible to force the FTM counter to its initial value and update the CHnOM bits in OUTMASK using PWM synchronization.

Note

PWM synchronization is available only in combine mode.

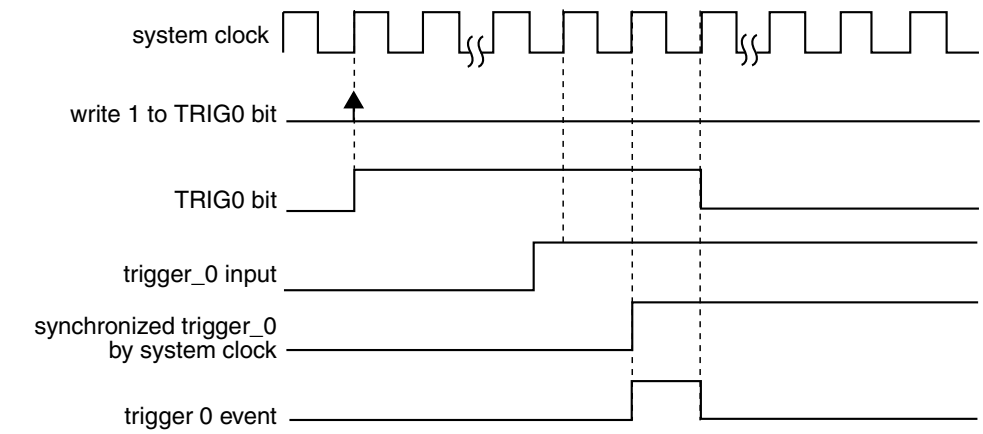
33.4.11.1 Hardware trigger

Each hardware trigger is synchronized by the system clock. The input signals are: trigger_0, trigger_1, and trigger_2.

A rising edge on the selected hardware trigger input (trigger n event) initiates PWM synchronization. A hardware trigger is selected when its enable bit is set (TRIGn = 1 where n = 0, 1, or 2). The TRIGn bit is cleared when 0 is written to it or when the trigger n event is detected.

For example, if TRIG0 and TRIG1 are enabled and only the trigger 1 event occurs, only the TRIG1 bit is cleared.

If a trigger n event occurs together with a write to set the TRIGn bit, then the synchronization is made, but the TRIGn bit remains set because of the last write.



Notes
- All hardware trigger (input signals: trigger_0, trigger_1, and trigger_2) have this same behavior

Figure 33-180. Hardware trigger event

33.4.11.2 Software trigger

A software trigger event occurs when 1 is written to the SWSYNC bit. The SWSYNC bit is cleared when 0 is written to it or when the PWM synchronization, which is initiated by the software event, is completed.

If the software trigger event occurs together with the event that clears the SWSYNC bit, then the synchronization is made using this trigger event and the SWSYNC bit remains set because of the last write.

For example, if PWMSYNC = 0 and REINIT = 0 and there is a software trigger event, then the load of MODH:L and CnVH:L registers is made only at the boundary cycle (CNTMIN and CNTMAX). In this case, the SWSYNC bit is cleared only at the boundary cycle, so you do not know when this bit is cleared. Therefore, it is possible a new write to set SWSYNC happens when FTM is clearing the SWSYNC because it is the selected boundary cycle of PWM synchronization that was started previously by the software trigger event.

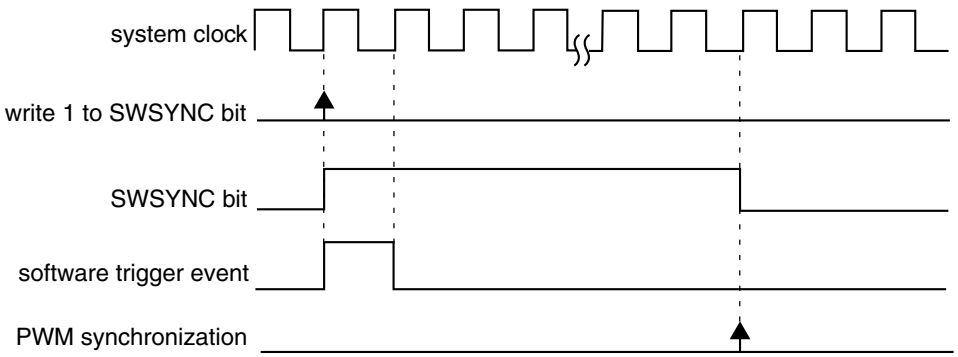


Figure 33-181. Software Trigger event

33.4.11.3 Boundary cycle

The CNTMAX and CNTMIN bits select the boundary cycle when the MODH:L and CnVH:L registers are updated with the value of their write buffer by PWM synchronization, except if (PWMSYNC = 0 and REINIT = 1).

If CNTMIN = 1, then the boundary cycle is the CNTINH:L value. MODH:L and CnVH:L registers are updated when the FTM counter reaches the CNTINH:L value. If CPWMS = 0, then CNTINH:L is reached when the FTM counter changes from MODH:L to CNTINH:L. If CPWMS = 1, then CNTINH:L is reached when the FTM counter changes from (CNTINH:L + 0x0001) to CNTINH:L.

If CNTMAX = 1, then the boundary cycle is the MODH:L value. MODH:L and CnVH:L registers are updated when the FTM counter reaches the MODH:L value. MODH:L is reached when the FTM counter changes from (MODH:L – 0x0001) to MODH:L, regardless of the CPWMS configuration.

If no boundary cycle was selected (CNTMAX = 0 and CNTMIN = 0), then the update of the MODH:L and CnVH:L registers is not made, unless (PWMSYNC = 0 and REINIT = 1).

If both boundary cycles were selected (CNTMAX = 1 and CNTMIN = 1), then the update of the MODH:L and CnVH:L registers is made in the first boundary cycle that occurs with valid conditions for MODH:L or CnVH:L synchronization, except if (PWMSYNC = 0 and REINIT = 1).

The CNTMAX and CNTMIN bits are cleared only by software.

Note

- PWM synchronization boundary cycle is available only when (CNTMIN = 1).
- PWM synchronization with (CNTMAX = 1) is not recommended and its results are not guaranteed.

33.4.11.4 MODH:L registers synchronization

The MODH:L synchronization occurs when the MODH:L registers are updated with the value of their write buffer.

The synchronization requires both bytes of MODH:L to have been written in one of the following situations.

- If PWMSYNC = 0 and REINIT = 0, then the synchronization is made on the next selected boundary cycle after an enabled trigger event takes place. If the trigger event was a software trigger, then the SWSYNC bit is cleared on the next selected boundary cycle. See the following figure.

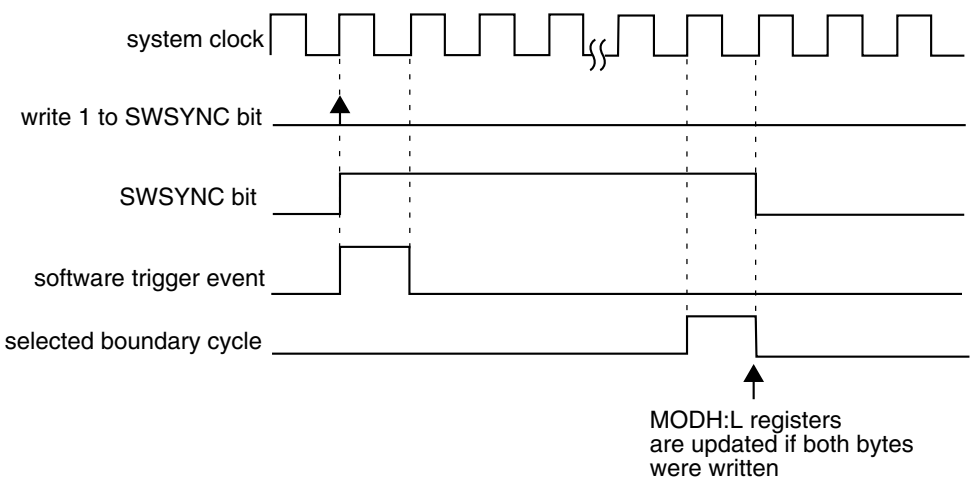


Figure 33-182. MODH:L synchronization when (PWMSYNC = 0), (REINIT = 0), and software trigger was used

If the trigger event was a hardware trigger, then the trigger enable bit (TRIGn) is cleared when the trigger n event is detected. See the following figure.

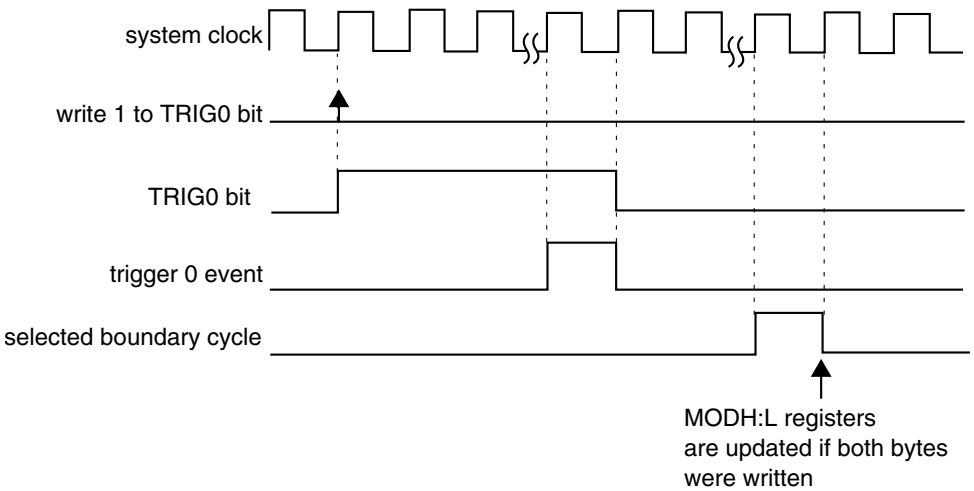


Figure 33-183. MODH:L synchronization when (PWMSYNC = 0), (REINIT = 0), and a hardware trigger was used

- If PWMSYNC = 0 and REINIT = 1, then the synchronization is made on the next enabled trigger event. If the trigger event was a software trigger, then the SWSYNC bit is cleared. See the following figure.

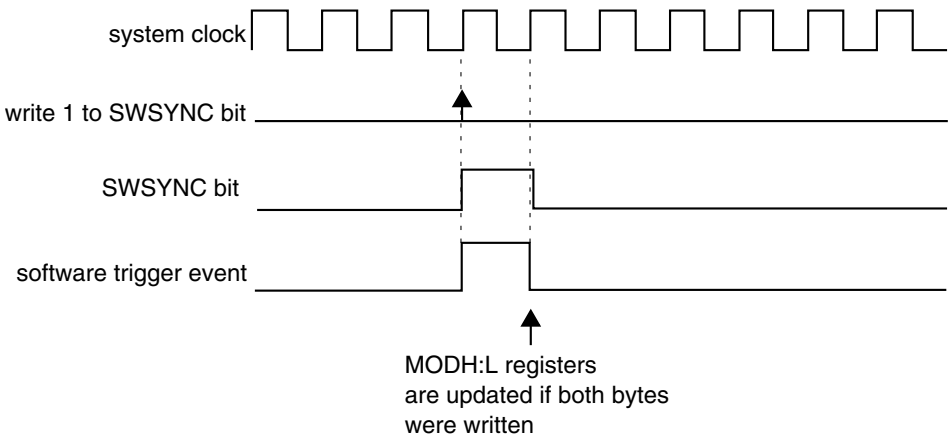


Figure 33-184. MODH:L synchronization when (PWMSYNC = 0), (REINIT = 1), and software trigger was used

If the trigger event was a hardware trigger, then the TRIGn bit is cleared. See the following figure.

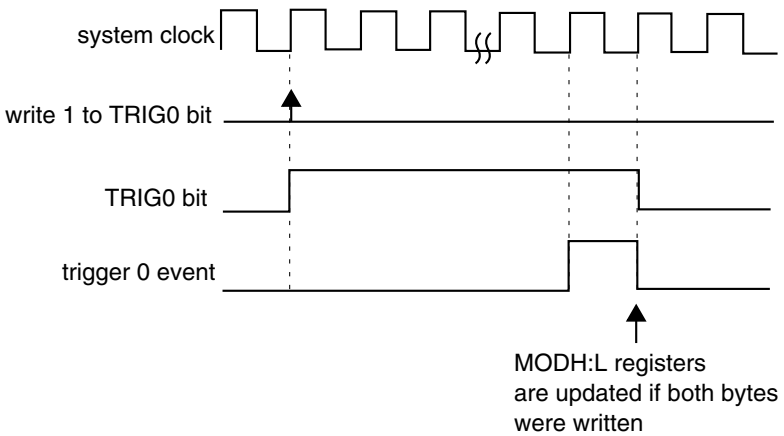


Figure 33-185. MODH:L synchronization when (PWMSYNC = 0), (REINIT = 1), and a hardware trigger was used

- If PWMSYNC = 1, then the synchronization is made on the next selected boundary cycle after the enabled software trigger event takes place. The SWSYNC bit is cleared on the next selected boundary cycle. See the following figure.

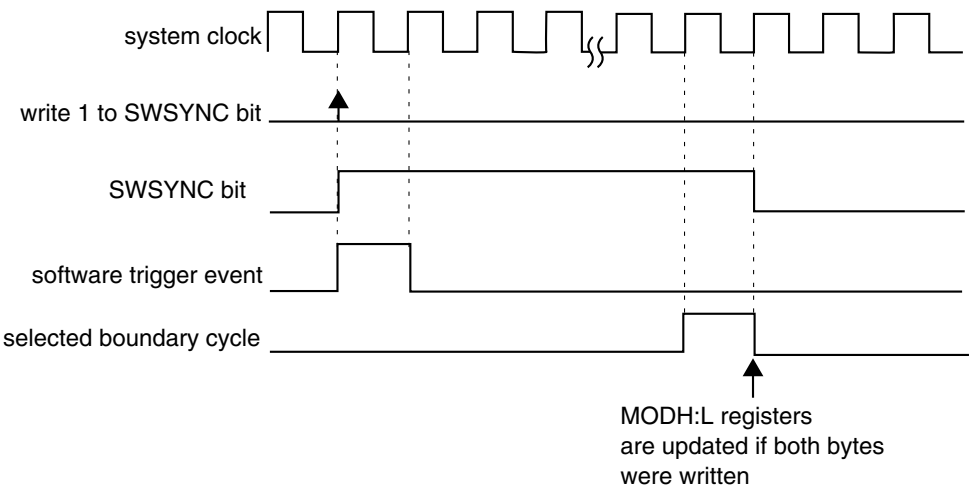


Figure 33-186. MODH:L synchronization when (PWMSYNC = 1)

33.4.11.5 CnVH:L registers synchronization

The CnVH:L synchronization occurs when the CnVH:L registers are updated with the value of their write buffer.

The synchronization requires both bytes of CnVH:L to have been written, SYNCEN = 1 and either a hardware or software trigger event as per MODH:L registers synchronization.

33.4.11.6 OUTMASK register synchronization

Any write to a CHnOM bit updates the OUTMASK write buffer. The CHnOM bit is updated with the value of its corresponding bit in the OUTMASK write buffer according to SYNCHOM and PWMSYNC bits.

- If SYNCHOM = 0, then the CHnOM bit is updated with the value of its write buffer equivalent in all rising edges of the system clock.

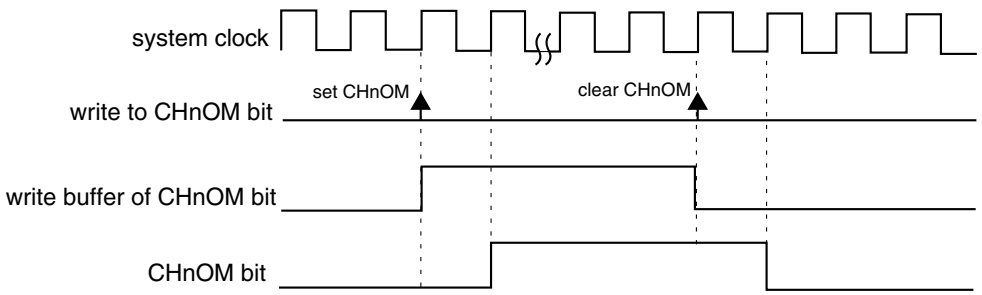


Figure 33-187. CHnOM synchronization when (SYNCHOM = 0)

- If SYNCHOM = 1 and PWMSYNC = 0, then this synchronization is made on the next enabled trigger event. If the trigger event was a software trigger, then the SWSYNC bit is cleared on the next selected boundary cycle. See the following figure.

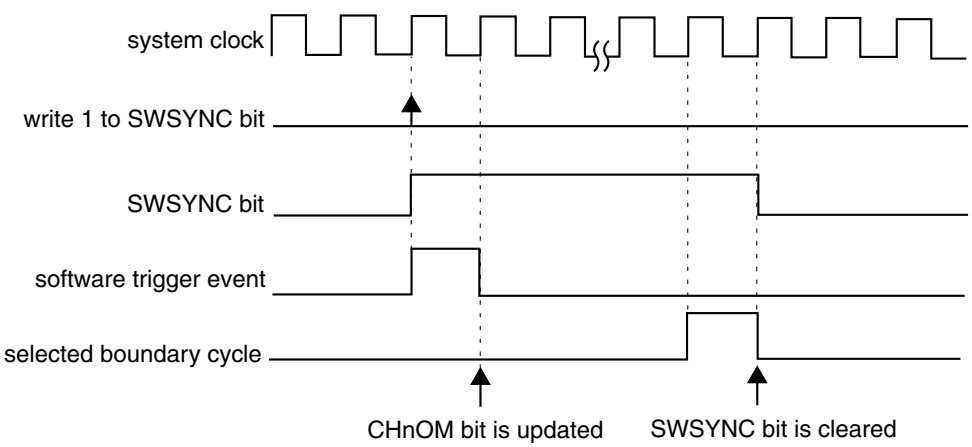


Figure 33-188. CHnOM synchronization when (SYNCHOM = 1), (PWMSYNC = 0) and software trigger was used

If the trigger event was a hardware trigger, then the trigger enable bit (TRIGn) is cleared when the trigger n event is detected. See the following figure.

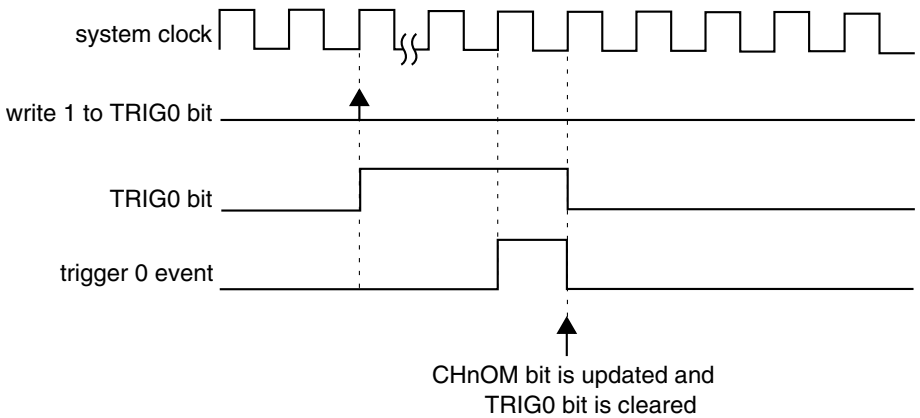


Figure 33-189. CHnOM synchronization when (SYNCHOM = 1), (PWMSYNC = 0), and a hardware trigger was used

- If SYNCHOM = 1 and PWMSYNC = 1, then this synchronization is made on the next enabled hardware trigger event. The trigger enable bit (TRIGn) is cleared when the enabled hardware trigger n event is detected. See the following figure.

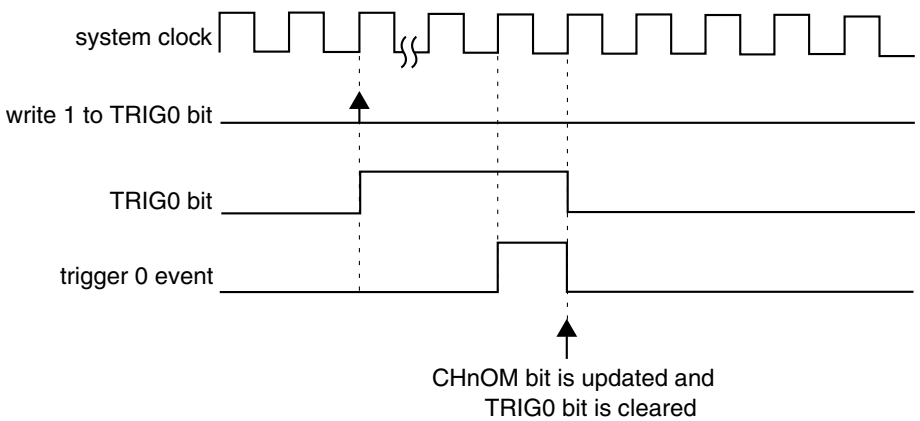


Figure 33-190. CHnOM Synchronization when (SYNCHOM = 1), (PWMSYNC = 1), and a hardware trigger was used

33.4.11.7 FTM counter synchronization

The FTM counter synchronization occurs when the FTM counter is updated with the value of the CNTINH:L registers and the channel outputs are forced to their initial value as defined by the channel configuration.

- If REINIT = 0, then this synchronization is made when the FTM counter changes from MODH:L to CNTINH:L.
- If REINIT = 1 and PWMSYNC = 0, then this synchronization is made on the next enabled trigger event. If the trigger event was a software trigger, then the SWSYNC bit is cleared. See the following figure.

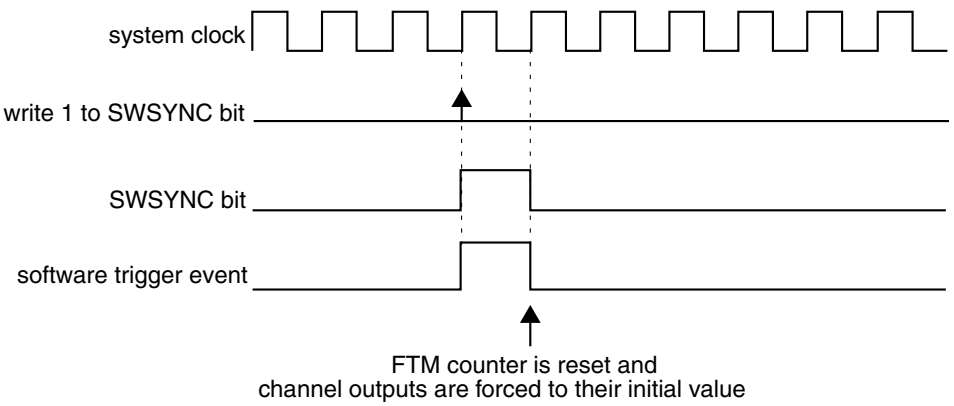


Figure 33-191. FTM counter synchronization when (REINIT = 1), (PWMSYNC = 0), and software trigger was used

If the trigger event was a hardware trigger, then the TRIGn bit is cleared. See the following figure.

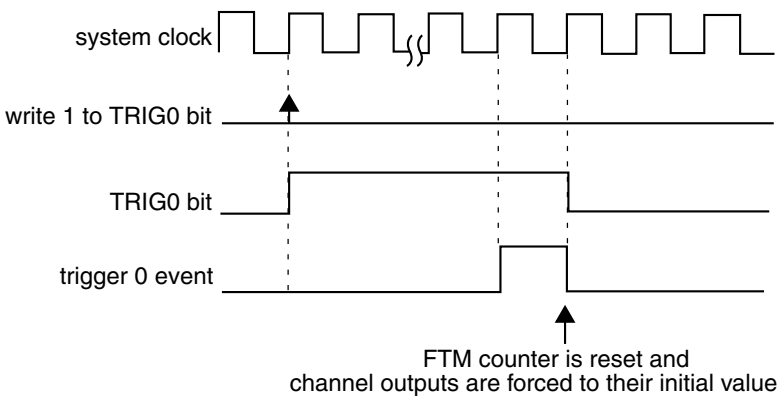


Figure 33-192. FTM counter synchronization when (REINIT = 1), (PWMSYNC = 0), and a hardware trigger was used

- If REINIT = 1 and PWMSYNC = 1, then this synchronization is made on the next enabled hardware trigger event. The trigger enable bit (TRIGn) is cleared when the enabled hardware trigger n event is detected. See the following figure.

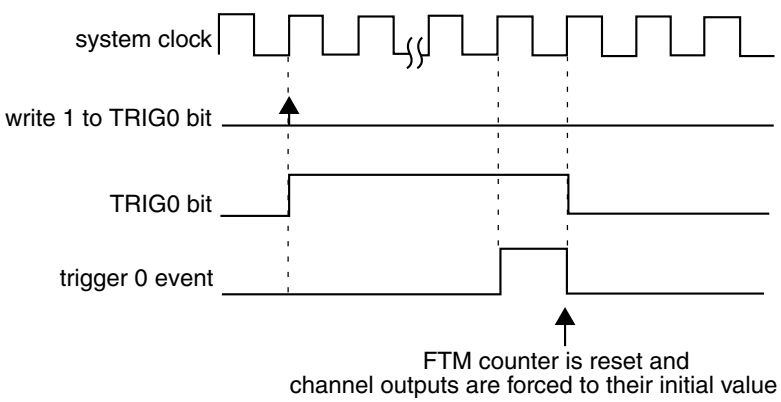


Figure 33-193. FTM counter synchronization when (REINIT = 1), (PWMSYNC = 1), and a hardware trigger was used

33.4.11.8 Summary of PWM synchronization

The following table shows the summary of PWM synchronization.

Table 33-188. Summary of PWM synchronization

Register or bit	PWMSYNC	REINIT	SYNCHOM	CNTMAX	CNTMIN	SYNCEN	Description
CNTINH:L	X	X	X	X	X	X	Changes take effect after the second byte is written. Effect is seen after the next TOF or PWM synchronization.
MODH:L	0	0	X	1	0	X	MODH:L are updated with their write buffer contents when the counter reaches its maximum value after the enabled hardware or software trigger has occurred.
	0	0	X	0	1	X	MODH:L are updated with their write buffer contents when the counter reaches its minimum value after the enabled hardware or software trigger has occurred.
	0	1	X	X	X	X	MODH:L are updated with their write buffer contents when the enabled hardware or software trigger occurs.
	1	X	X	1	0	X	MODH:L are updated with their write buffer contents when the counter reaches its maximum value after the enabled software trigger has occurred.
	1	X	X	0	1	X	MODH:L are updated with their write buffer contents when the counter reaches its minimum value after the enabled software trigger has occurred.

Table continues on the next page...

Table 33-188. Summary of PWM synchronization (continued)

Register or bit	PWMSYNC	REINIT	SYNCHOM	CNTMAX	CNTMIN	SYNCEN	Description
CnVH:L	0	0	X	1	0	1	CnVH:L are updated with their write buffer contents when the counter reaches its maximum value after the enabled hardware or software trigger has occurred.
	0	0	X	0	1	1	CnVH:L are updated with their write buffer contents when the counter reaches its minimum value after the enabled hardware or software trigger has occurred.
	0	1	X	X	X	1	CnVH:L are updated with their write buffer contents when the enabled hardware or software trigger occurs.
	1	X	X	1	0	1	CnVH:L are updated with their write buffer contents when the counter reaches its maximum value after the enabled software trigger has occurred.
	1	X	X	0	1	1	CnVH:L are updated with their write buffer contents when the counter reaches its minimum value after the enabled software trigger has occurred.
CNTH:L	0	1	X	X	X	X	CNTH:L are forced to the FTM counter initial value when the enabled hardware or software trigger occurs.
	1	1	X	X	X	X	CNTH:L are forced to the FTM counter initial value when the enabled hardware trigger occurs.
OUTMASK	X	X	0	X	X	X	Changes to OUTMASK take effect on the next rising edge of the system clock.
	0	X	1	X	X	X	OUTMASK is updated with its write buffer contents when the enabled hardware or software trigger occurs.
	1	X	1	X	X	X	OUTMASK is updated with its write buffer contents when the enabled hardware trigger occurs.

Table continues on the next page...

Table 33-188. Summary of PWM synchronization (continued)

Register or bit	PWMSYNC	REINIT	SYNCHOM	CNTMAX	CNTMIN	SYNCEN	Description
SWSYNC bit	0	0	X	1	0	X	SWSYNC bit is cleared when the counter reaches its maximum value after the enabled software trigger has occurred.
	0	0	X	0	1	X	SWSYNC bit is cleared when the counter reaches its minimum value after the enabled software trigger has occurred.
	0	1	X	X	X	X	SWSYNC bit is cleared when the enabled software trigger occurs.
	1	X	X	1	0	X	SWSYNC bit is cleared when the counter reaches its maximum value after the enabled software trigger has occurred.
	1	X	X	0	1	X	SWSYNC bit is cleared when the counter reaches its minimum value after the enabled software trigger has occurred.
TRIGn bit	X	X	X	X	X	X	TRIGn bit is cleared when the enabled hardware trigger has occurred.

33.4.12 Deadtime insertion

The deadtime insertion is enabled when (DTEN = 1) and (DTVAL[5:0] is non- zero).

DEADTIME register defines the deadtime delay that can be used for all FTM channels. The DTPS[1:0] bits define the prescaler for the system clock and the DTVAL[5:0] bits define the deadtime modulo; that is, the number of deadtime prescaler clocks).

The deadtime delay insertion ensures that no two complementary signals (channel (n) and (n+1)) drive the active state at the same time.

For POL(n) = 0, POL(n+1) = 0, and deadtime enabled, a rising edge on the output of channel (n) remains low for the duration of the deadtime delay, after which the rising edge appears on the output. Similarly, when a falling edge is due on the output of channel (n), the channel (n+1) output remains low for the duration of the deadtime delay, after which the channel (n+1) output will have a rising edge.

For POL(n) = 1, POL(n+1) = 1, and deadtime enabled, a falling edge on the output of channel (n) remains high for the duration of the deadtime delay, after which the falling edge appears on the output. Similarly, when a rising edge is due on the output of channel (n), the channel (n+1) output remains high for the duration of the deadtime delay, after which the channel (n+1) output will have a falling edge.

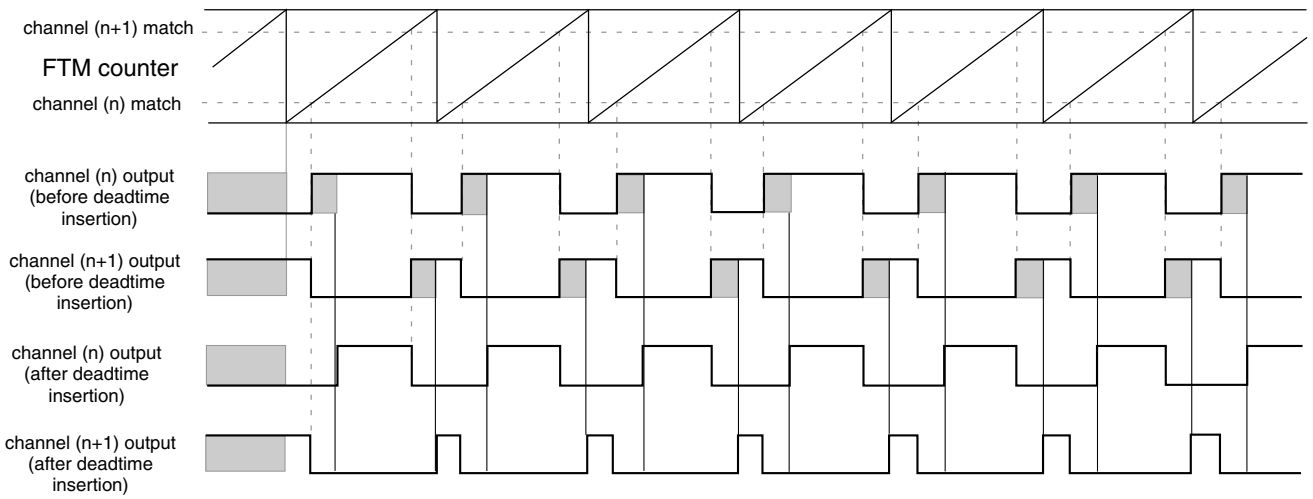


Figure 33-194. Deadtime insertion with ELSnB:ELSnA = 1:0, POL(n) = 0, and POL(n+1) = 0

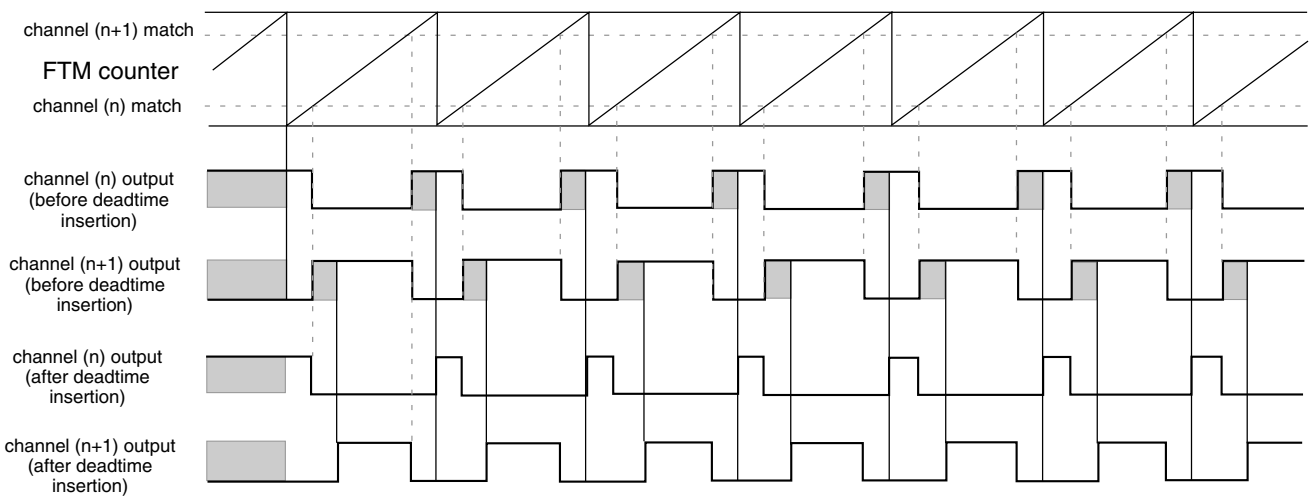


Figure 33-195. Deadtime insertion with ELSnB:ELSnA = X:1, POL(n) = 0, and POL(n+1) = 0

NOTE

Deadtime feature is available only in combine and complementary modes.

33.4.12.1 Deadtime insertion corner cases

If (PS[2:0] bits are cleared), (DTPS[1:0] = 0:0 or DTPS[1:0] = 0:1):

- and the deadtime delay is greater than or equal to the channel (n) duty cycle ((C(n +1)VH:L – C(n)VH:L) × system clock), then the channel (n) output is always the inactive value (POL(n) bit value).
- and the deadtime delay is greater than or equal to the channel (n+1) duty cycle ((MODH:L – CNTINH:L + 1 – (C(n+1)VH:L – C(n)VH:L)) × system clock), then the channel (n+1) output is always the inactive value (POL(n+1) bit value).

Although in most cases the deadtime delay is not comparable to channels (n) and (n+1) duty cycle, the following figures show examples where the deadtime delay is comparable to the duty cycle.

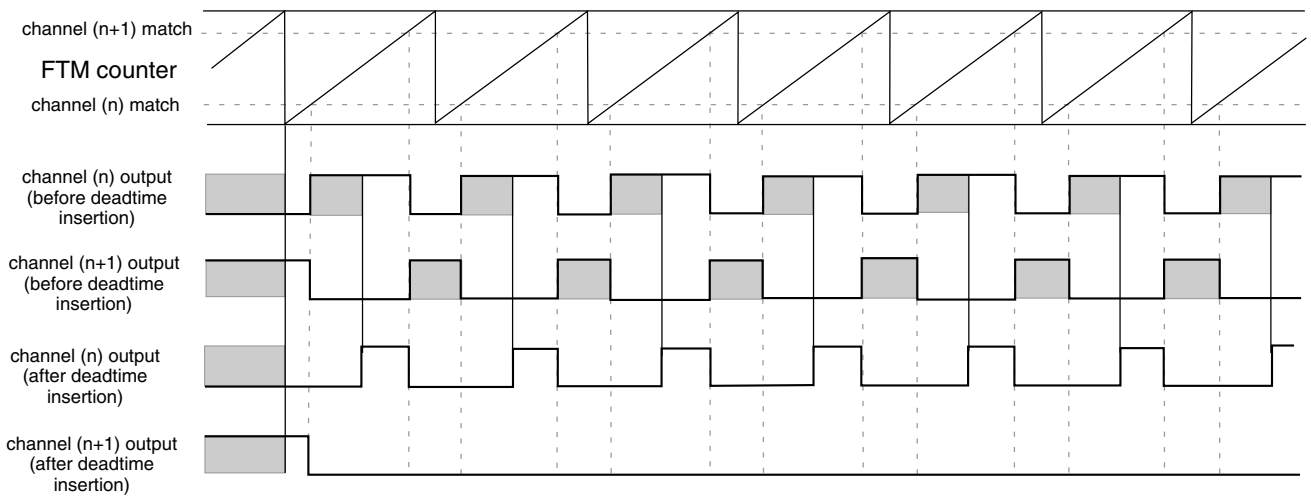


Figure 33-196. Example of the deadtime insertion (ELSnB:ELSnA = 1:0, POL(n) = 0, and POL(n+1) = 0) when the deadtime delay is comparable to channel (n+1) duty cycle

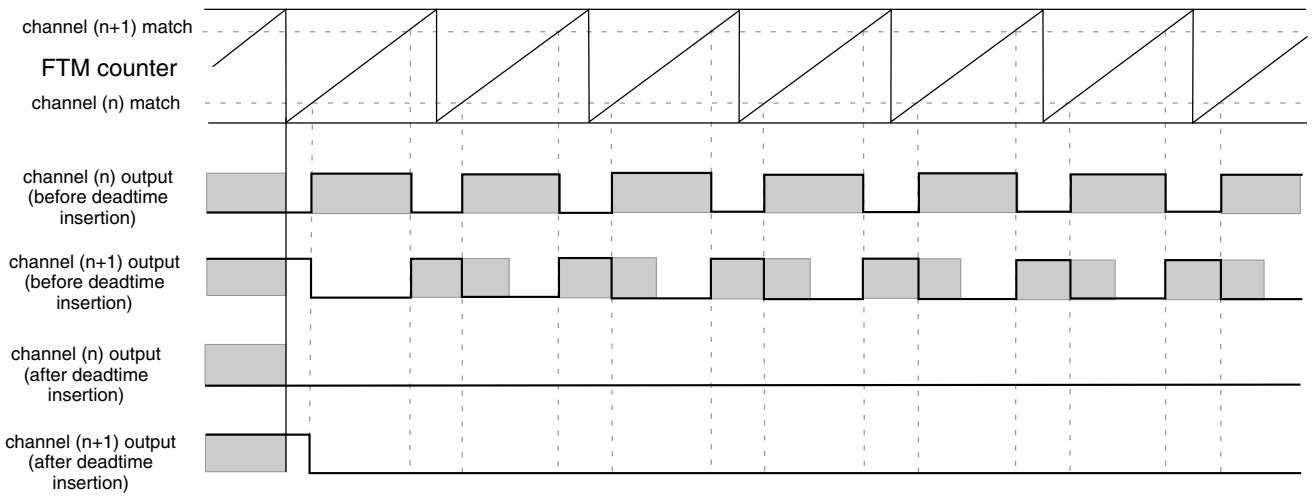


Figure 33-197. Example of the deadtime insertion (ELSnB:ELSnA = 1:0, POL(n) = 0, and POL(n+1) = 0) when the deadtime delay Is comparable to channels (n) and (n+1) duty cycle

33.4.13 Output mask

The output mask register OUTMASK can be used to force channel outputs to their inactive state through software; for example, to control a BLDC motor.

Any write to a CHnOM bit updates the OUTMASK write buffer. The CHnOM bit is updated with the value of its corresponding bit in the OUTMASK write buffer according to OUTMASK register synchronization.

If CHnOM = 1, then the channel (n) output is forced to its inactive state, defined by the POLn bit in register POL. If CHnOM = 0, then the channel (n) output is unaffected by the output mask function.

When a CHnOM bit is cleared, the channel (n) output is enabled. See the following figure.

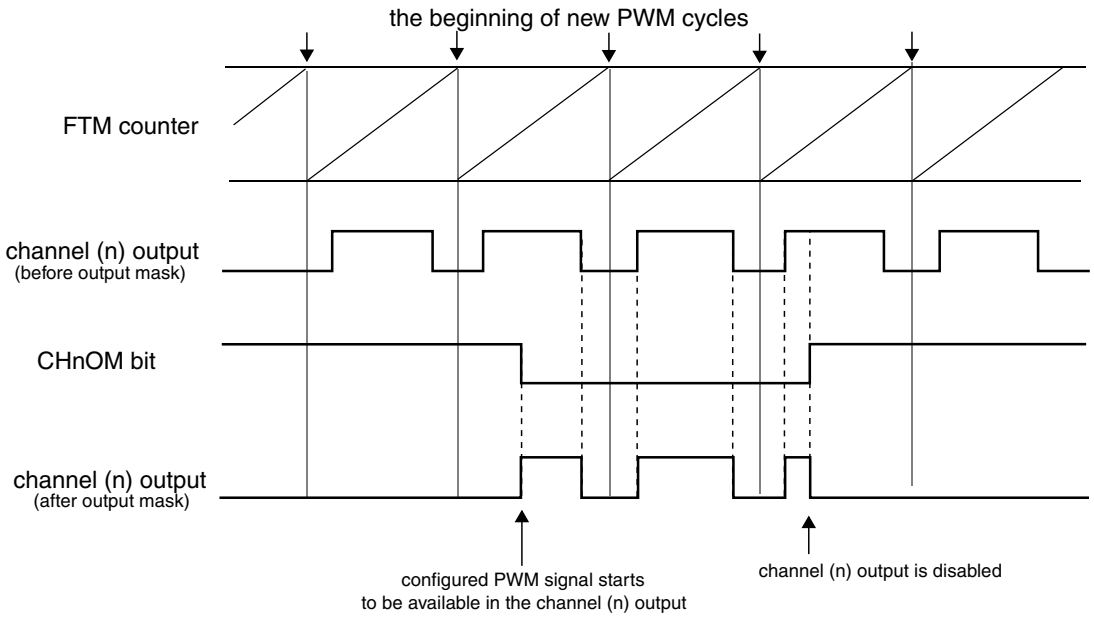


Figure 33-198. Output mask

The following table shows the output mask result before the polarity control.

Table 33-189. Output mask result for channel (n) before the polarity control

CHnOM	Output Mask Input	Output Mask Result
0	inactive state	inactive state
	active state	active state
1	inactive state	inactive state
	active state	

Note

Output mask is available only in combine mode.

33.4.14 Fault control

The fault control is enabled if (FTMEN = 1) and (FAULTM[1:0] ≠ 0:0).

FTM can have up to four fault inputs. FAULTnEN bit (where n = 0, 1, 2, 3) enables the fault input n and FFLTRnEN bit enables the fault input n filter. FFVAL[3:0] bits select the value of the enabled filter in each enabled fault input.

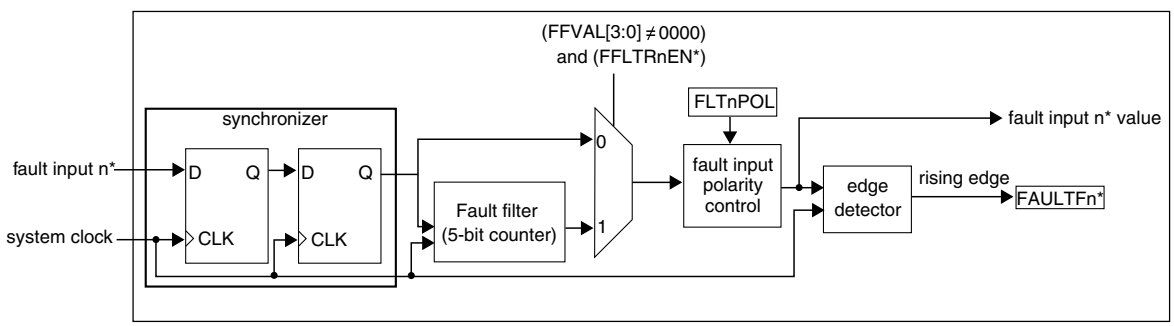
First, each fault input signal is synchronized by the system clock; see the synchronizer block in the following figure. Following synchronization, the fault input n signal enters the filter block. When there is a state change in the fault input n signal, the 5-bit counter

is reset and starts counting up. As long as the new state is stable on the fault input n, the counter continues to increment. If the 5-bit counter overflows and exceeds the value of the FFVAL[3:0] bits, the new fault input n value is validated. It is then transmitted as a pulse edge to the edge detector.

If the opposite edge appears on the fault input n signal before validation (counter overflow), the counter is reset. At the next input transition, the counter starts counting again. Any pulse that is shorter than the minimum value selected by FFVAL[3:0] bits (× system clock) is regarded as a glitch and is not passed on to the edge detector.

The fault input n filter is disabled when the FFVAL[3:0] bits are zero or when FAULTnEN = 0. In this case the fault input n signal is delayed two rising edges of the system clock and the FAULTFn bit is set on the third rising edge of the system clock after a rising edge occurs on the fault input n.

If FFVAL[3:0] ≠ 0000 and FAULTnEN = 1, then the fault input n signal is delayed (3 + FFVAL[3:0]) rising edges of the system clock; that is, the FAULTFn bit is set (4 + FFVAL[3:0]) rising edges of the system clock after a rising edge occurs on the fault input n.



* where n = 3, 2, 1, 0

Figure 33-199. Fault input n control block diagram

If the fault control and fault input n are enabled and a rising edge at the fault input n signal is detected, then the FAULTFn bit is set. The FAULTF bit is the logic OR of FAULTFn[3:0] bits. See the following figure.

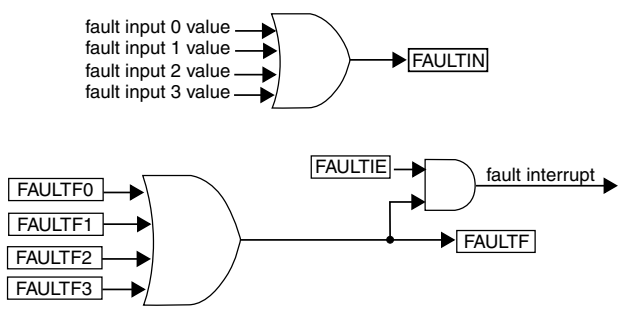


Figure 33-200. FAULTF and FAULTIN bits and fault interrupt

If the fault control is enabled (FAULTM[1:0] ≠ 0:0), a fault condition has occurred (rising edge at the logic OR of the enabled fault input) and (FAULTEN = 1), then channel (n) and (n+1) outputs are forced to their safe value (that is, the channel (n) output is forced to the value of POL(n) and the channel (n+1) is forced to the value of POL(n +1)).

The fault interrupt is generated when (FAULTF = 1) and (FAULTIE = 1). This interrupt request remains set until:

- Software clears the FAULTF bit (by reading FAULTF bit as 1 and writing 0 to it)
- Software clears the FAULTIE bit
- A reset occurs

Note

Fault control is available only in combine mode.

33.4.14.1 Automatic fault clearing

If the automatic fault clearing is selected (FAULTM[1:0] = 1:1), then the disabled channel outputs are enabled when the fault input signal (FAULTIN) returns to zero and a new PWM cycle begins. See the following figure.

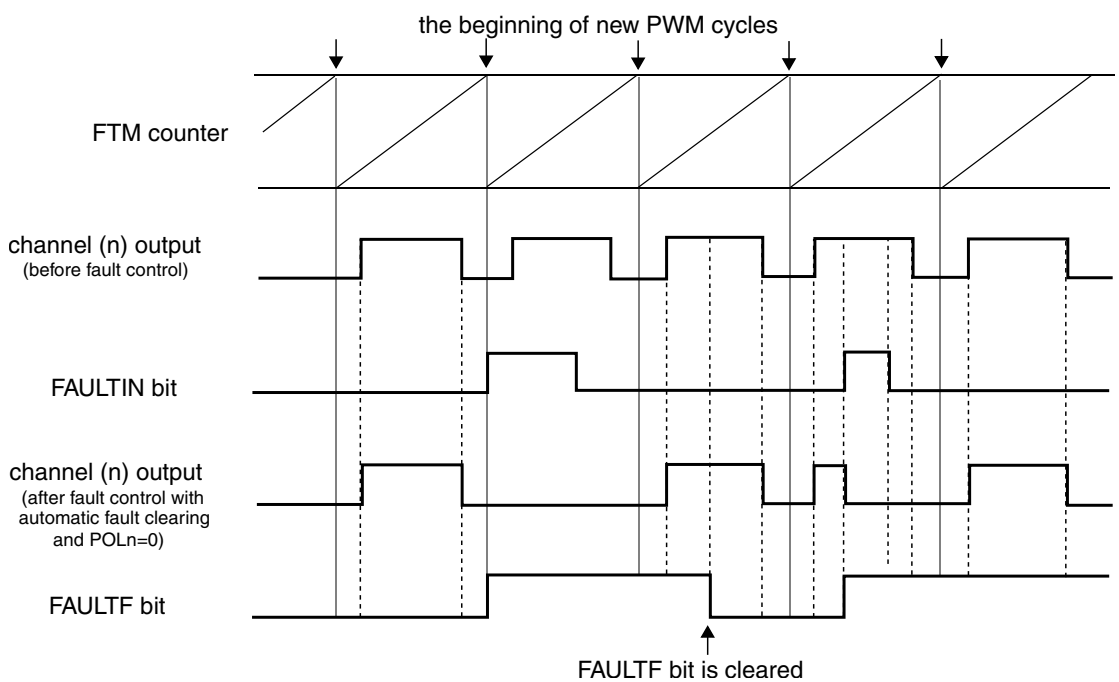


Figure 33-201. Fault control with automatic fault clearing

33.4.14.2 Manual fault clearing

If the manual fault clearing is selected (FAULTM[1:0] = 0:1 or 1:0), then disabled channel outputs are enabled when the FAULTF bit is cleared and a new PWM cycle begins. See the following figure.

It is possible to manually clear a fault by clearing the FAULTF bit, and enable disabled channels regardless of the fault input signal (FAULTIN) (the filter output if the filter is enabled or the synchronizer output if the filter is disabled). However, it is recommended to verify the value of the fault input signal (value of the FAULTIN bit) before clearing the FAULTF bit to avoid unpredictable results.

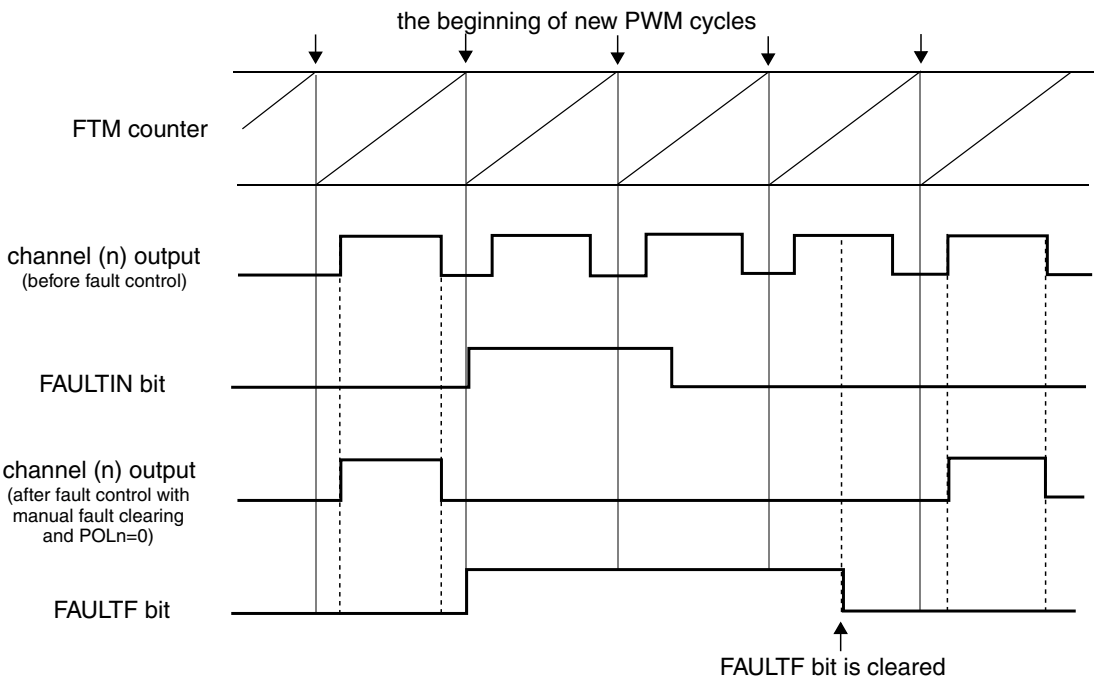


Figure 33-202. Fault control with manual fault clearing

33.4.15 Polarity control

The POLn bit selects the channel (n) output polarity:

- If (POLn = 0), the channel (n) output polarity is active-high: one is the active state; zero is the inactive state.
- If (POLn = 1), the channel (n) output polarity is active-low: zero is the active state; one is the inactive state.

Note

Polarity control is available only in combine mode.

33.4.16 Initialization

The initialization forces the CHnOI bit value to the channel (n) output when a one is written to the INIT bit.

Note

- It is recommended to use the initialization only when the FTM counter is disabled (CLKS[1:0] = 0:0).
- Initialization is available only in combine mode.

33.4.17 Features priority

The following figure shows the priority of the features that can be combined to generate channel (n) and (n+1) outputs.

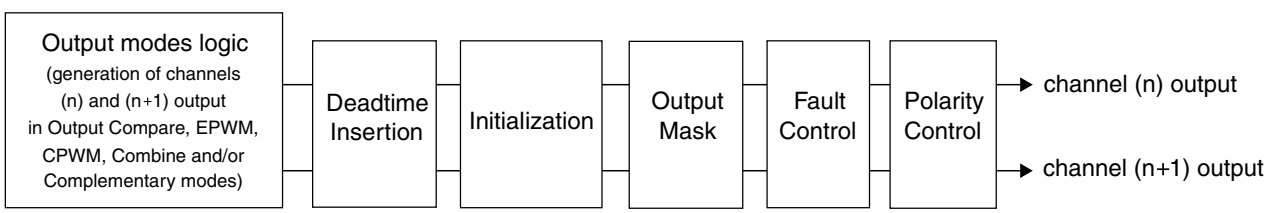


Figure 33-203. FTM features priority

33.4.18 Channel trigger output

The channel trigger output is generated if (FTMEN = 1) and one or more channels were selected by the CHjTRIG bit, where j = 0, 1, 2, 3, 4, or 5. The CHjTRIG bit defines if the channel (j) match (that is, FTM counter = C(j)VH:L) generates the trigger.

The channel trigger output provides a trigger signal that is used for on-chip modules.

The FTM is able to generate multiple triggers in one PWM period. Because each trigger is generated for a specific channel, several channels are required to implement this functionality. This behavior is described in the following figure.

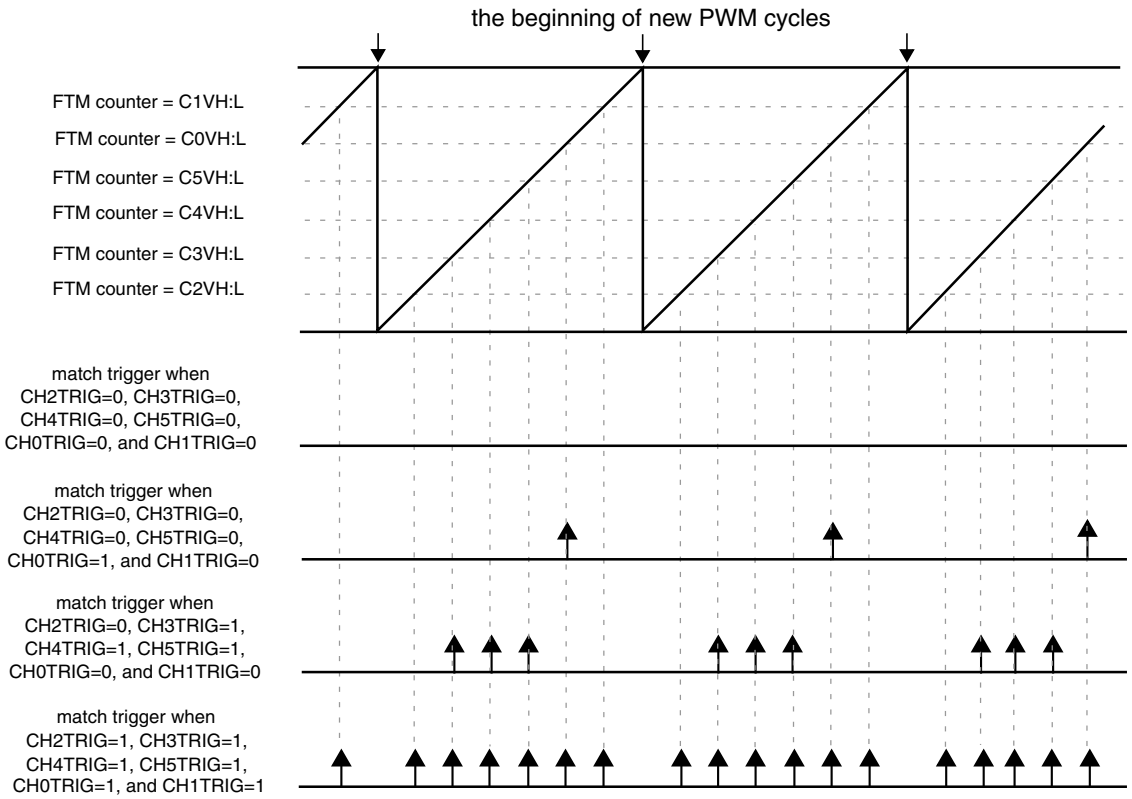


Figure 33-204. Match triggers

Note

Match trigger is available only in combine mode.

33.4.19 Initialization trigger

If INITTRIGEN = 1, the FTM generates a trigger when the FTM counter is updated with the CNTINH:L registers value in the following cases:

- The FTM counter is automatically updated with the CNTINH:L registers value by selected counting mode.

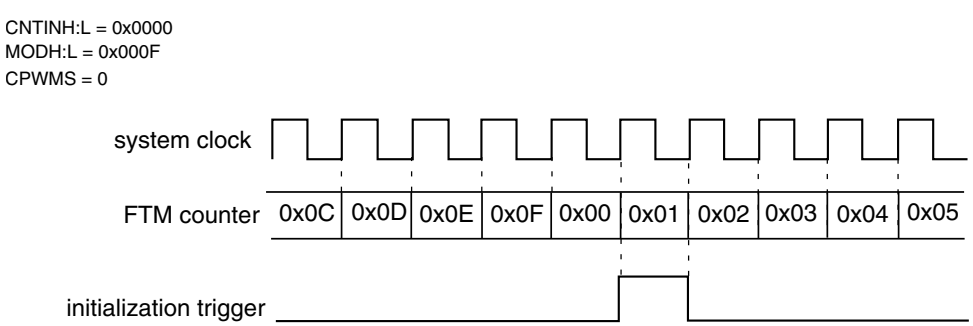


Figure 33-205. Initialization trigger is generated when the FTM counter achieves the value of CNTINH:L

- When there is a write to CNTH or CNTL register

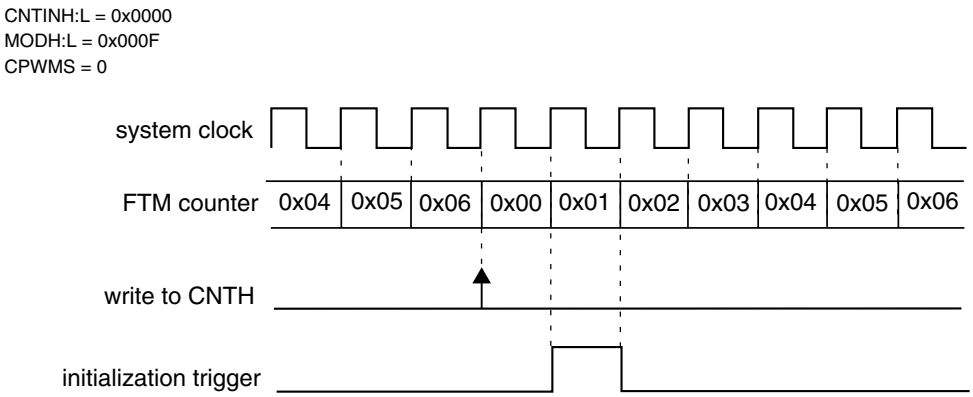


Figure 33-206. Initialization trigger is generated when there is a write to CNTH or CNTL

- When there is the FTM counter synchronization

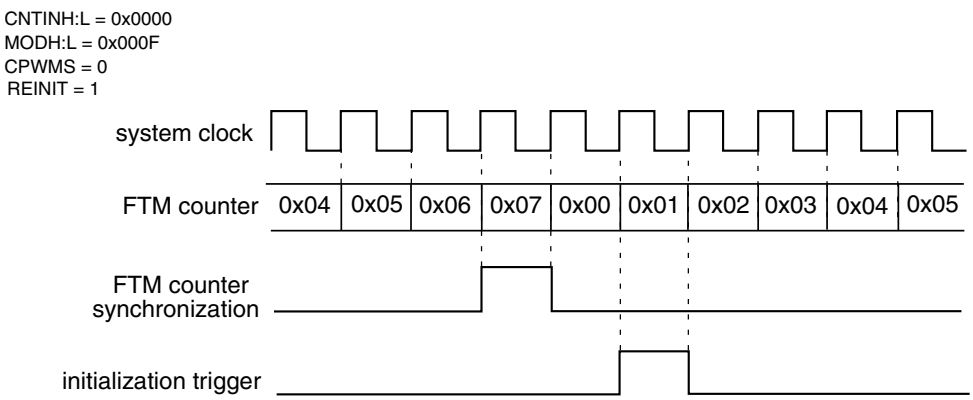


Figure 33-207. Initialization trigger is generated when there is the FTM counter synchronization

- If (CNTH:L = CNTINH:L), (CLKS[1:0] = 0:0), and a value different from zero is written to CLKS[1:0] bits

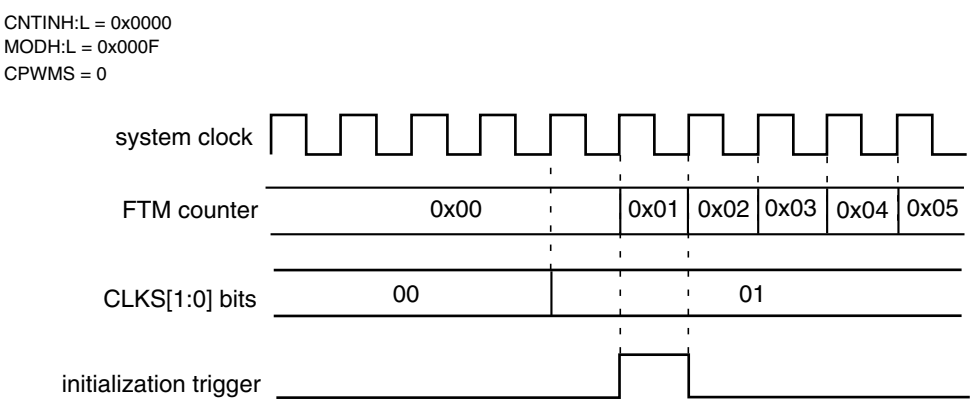


Figure 33-208. Initialization trigger is generated if (CNTH:L = CNTINH:L) and (CLKS[1:0] = 0:0) and a value different from zero is written to CLKS[1:0] bits

The initialization trigger output provides a trigger signal that is used for on-chip modules.

Note

Initialization trigger is available only in combine mode.

33.4.20 Capture test mode

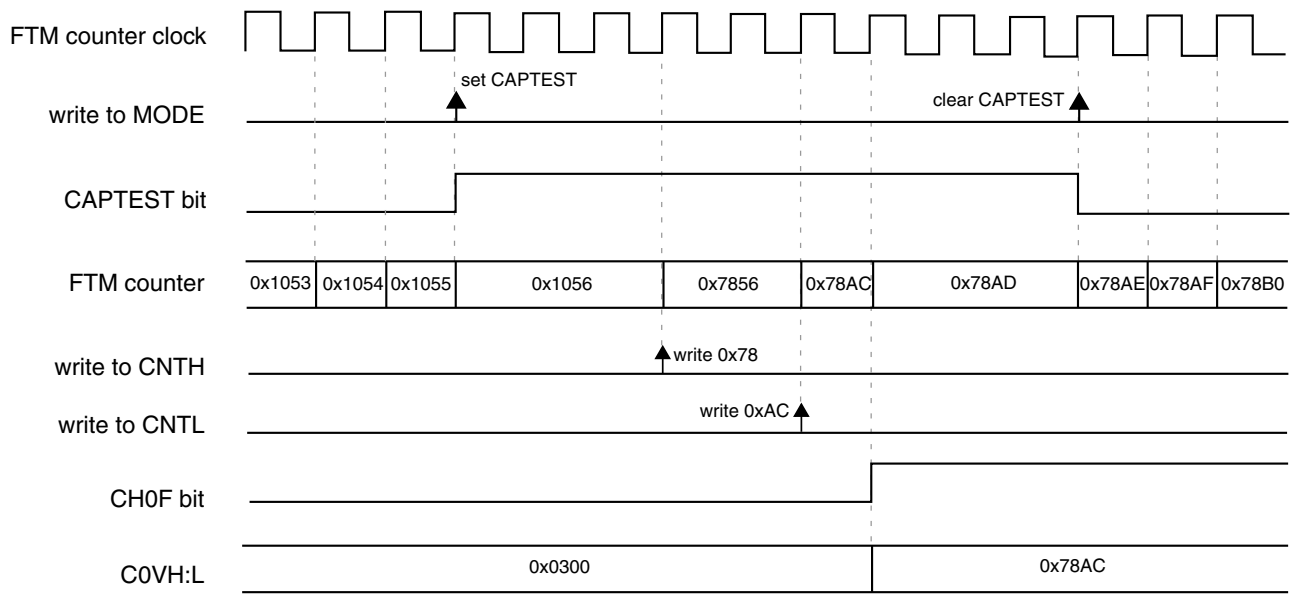
The capture test mode allows the testing of the CnVH:L registers, the FTM counter, and the interconnection logic between the FTM counter and CnVH:L registers.

In this test mode, all channels must be configured for input capture mode (see Input capture mode) and FTM counter must be configured for up-counting (see Up counting).

When the capture test mode is enabled (CAPTEST = 1), the FTM counter is frozen and any write to CNTH and CNTL updates directly the FTM counter; see the following figure. After both bytes were written, independent of the order, all CnVH:L registers are updated with the value that was written to CNTH:L registers and CHnF bits are set. Therefore, the FTM counter is updated with its next value according to its configuration. Its next value depends on CNTINH:L, MODH:L, and the value that was written to FTM counter.

The next reads of CnVH:L registers return the value that was written to FTM counter and the next reads of CNTH:L register return the next value of the FTM counter.

The read coherency mechanism of CNTH:L and CnVH:L registers remains enabled.



Notes
- FTM counter configuration: (FTMEN = 1), (QUADEN = 0) if the quadrature decoder feature is supported, (CAPTEST = 1), (CPWMS = 0), (CNTINH:L = 0x0000), and (MODH:L = 0xFFFF)

- FTM channel n configuration: input capture mode – (DECAPEN = 0), (COMBINE = 0), and (MSnB:MSnA = 0:0)

Figure 33-209. Capture test mode

33.4.21 DMA

The channel generates a DMA transfer request according to DMA and CHnIE bits (see the following table).

Table 33-190. Channel DMA Transfer Request

DMA	CHnIE	Channel DMA Transfer Request	Channel Interrupt
0	0	The channel DMA transfer request is not generated.	The channel interrupt is not generated.
0	1	The channel DMA transfer request is not generated.	The channel interrupt is generated if (CHnF = 1).
1	0	The channel DMA transfer request is not generated.	The channel interrupt is not generated.
1	1	The channel DMA transfer request is generated if (CHnF = 1).	The channel interrupt is not generated.

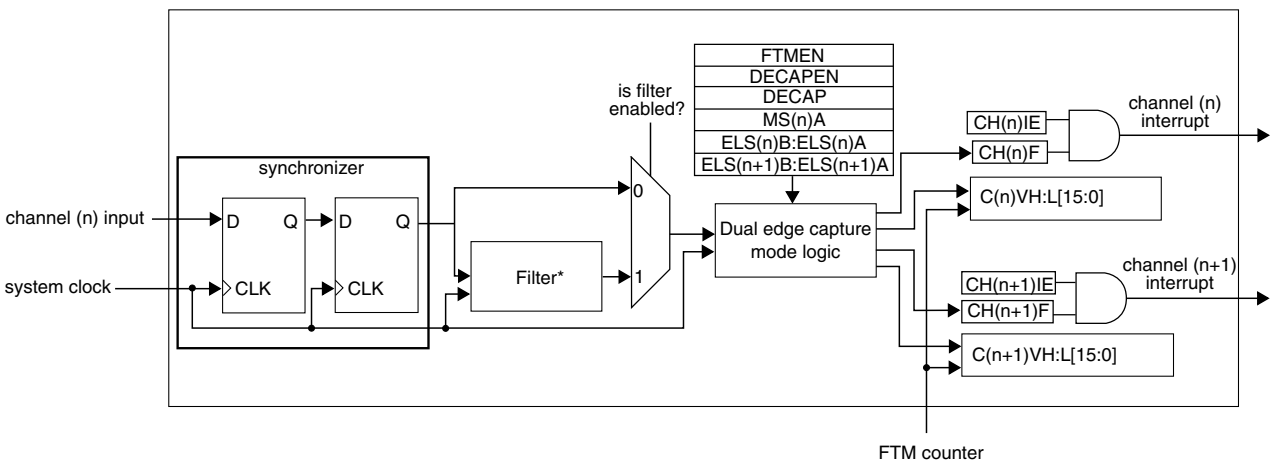
If DMA = 1, the CHnF bit is cleared either by channel DMA transfer done or reading CnSC while CHnF is set and then writing a logic 0 to CHnF bit according to CHnIE bit (see the following table).

Table 33-191. Clear CHnF Bit when DMA = 1

CHnIE	How CHnF Bit Can Be Cleared
0	CHnF bit is cleared either by the channel DMA transfer done or reading CnSC while CHnF is set and then writing a logic 0 to CHnF bit.
1	CHnF bit is cleared by the channel DMA transfer done.

33.4.22 Dual edge capture mode

The dual edge capture mode is selected if FTMEN = 1 and DECAPEN = 1. This mode allows software to measure a pulse width or period of the signal on the input of channel (n) of a channel pair. The channel (n) filter can be active in this mode when n is the channels 0 or 2.



* Filtering function for dual edge capture mode is only available in the channels 0 and 2

Figure 33-210. Dual edge capture mode block diagram

The MS(n)A bit defines if the dual edge capture mode is one-shot or continuous according to table "Mode, Edge, and Level Selection".

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) as described in table "Dual Edge Capture Mode — Edge Polarity Selection". If both ELS(n)B:ELS(n)A and ELS(n+1)B:ELS(n+1)A bits select the same edge, then it is the period measurement. If these bits select different edges, then it is a pulse width measurement.

In the dual edge capture mode, only channel (n) input is used and channel (n+1) input is ignored.

If the selected edge by channel (n) bits is detected at channel (n) input, then CH(n)F bit is set and the channel (n) interrupt is generated (if CH(n)IE = 1). If the selected edge by channel (n+1) bits is detected at channel (n) input and (CH(n)F = 1), then CH(n+1)F bit is set and the channel (n+1) interrupt is generated (if CH(n+1)IE = 1).

The C(n)VH:L registers store the value of FTM counter when the selected edge by channel (n) is detected at channel (n) input. The C(n+1)VH:L registers store the value of FTM counter when the selected edge by channel (n+1) is detected at channel (n) input.

In this mode, the coherency mechanism of the pair of channels ensures that data is coherent when the C(n)VH:L and C(n+1)VH:L registers are read. Note that the C(n)VH:L registers must be read first before reading the C(n+1)VH:L registers. C(n)VH:L registers must be read first than C(n+1)VH:L registers.

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.
- It is expected that the dual edge capture mode be used with ELS(n)B:ELS(n)A = 0:1 or 1:0, ELS(n+1)B:ELS(n+1)A = 0:1 or 1:0 and the FTM counter in free running counter mode. See Free running counter.

33.4.22.1 One-shot capture mode

The one-shot capture mode is selected when (FTMEN = 1), (DECAPEN = 1), and (MS(n)A = 0). In this capture mode, only one pair of edges at the channel (n) input is captured. The ELS(n)B:ELS(n)A bits select the first edge to be captured, and ELS(n +1)B:ELS(n+1)A bits select the second edge to be captured.

The edge captures are enabled while DECAP bit is set. For each new measurement in one-shot capture mode, first the CH(n)F and CH(n+1) bits must be cleared, and then the DECAP bit must be set.

In this mode, the DECAP bit is automatically cleared by FTM when the edge selected by channel (n+1) is captured. Therefore, while DECAP bit is set, the one-shot capture is in process. When this bit is cleared, both edges were captured and the captured values are ready for reading in the C(n)VH:L and C(n+1)VH:L registers.

Similarly, when the CH(n+1)F bit is set, both edges were captured and the captured values are ready for reading in the C(n)VH:L and C(n+1)VH:L registers.

33.4.22.2 Continuous capture mode

The continuous capture mode is selected when (FTMEN = 1), (DECAPEN = 1), and (MS(n)A = 1). In this capture mode, the edges at the channel (n) input are captured continuously. The ELS(n)B:ELS(n)A bits select the initial edge to be captured, and ELS(n+1)B:ELS(n+1)A bits select the final edge to be captured.

The edge captures are enabled while DECAP bit is set. For the initial use, first the CH(n)F and CH(n+1)F bits must be cleared, and then DECAP bit must be set to start the continuous measurements.

When the CH(n+1)F bit is set, both edges are captured and the captured values are ready for reading in the C(n)VH:L and C(n+1)VH:L registers. The latest captured values are always available in these registers even after the DECAP bit is cleared.

In this mode, it is possible to clear only the CH(n+1)F bit. Therefore, when the CH(n+1)F bit is set again, the latest captured values are available in C(n)VH:L and C(n+1)VH:L registers.

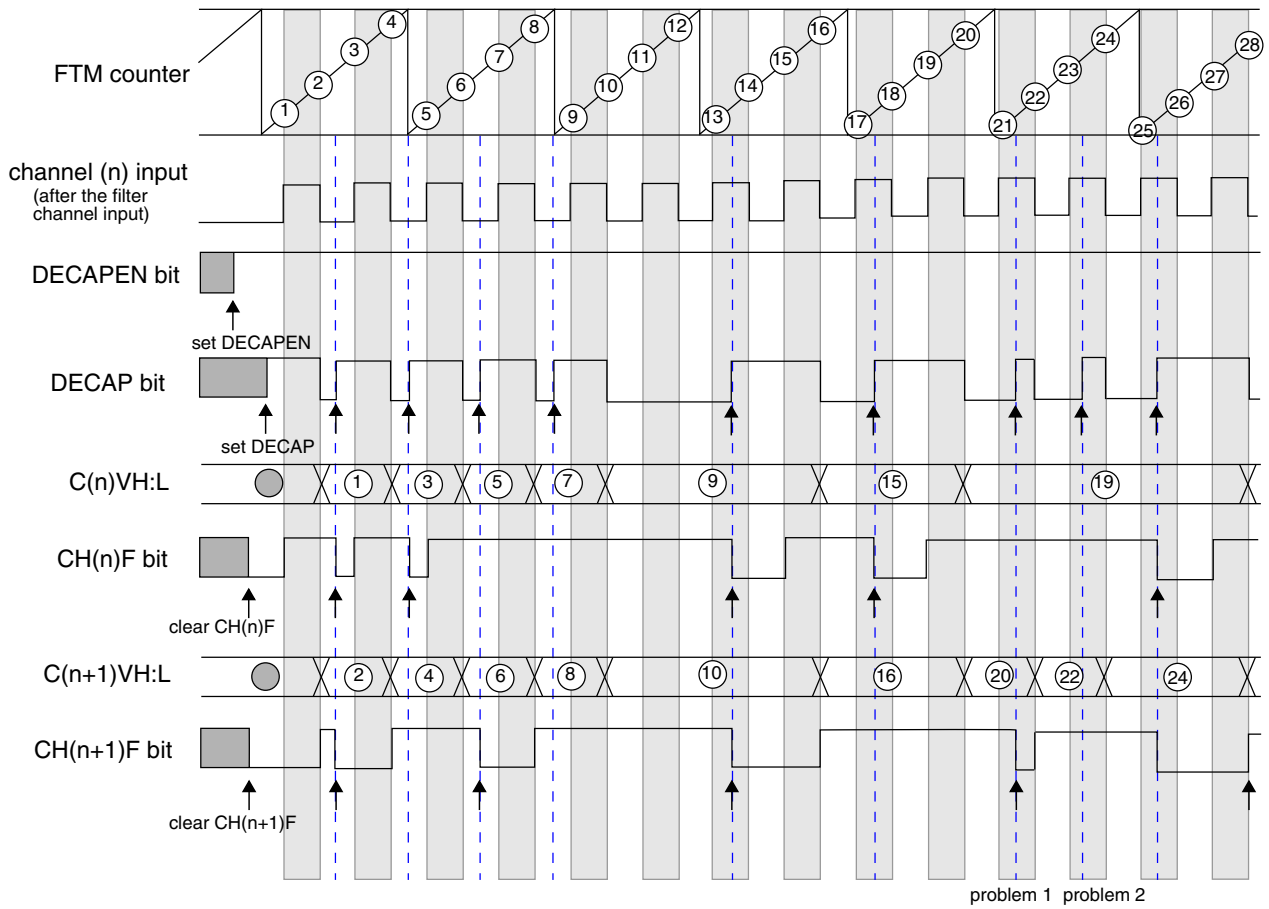
For a new sequence of the measurements in the dual edge capture – continuous mode, it is recommended to clear the CH(n)F and CH(n+1) bits to start new measurements.

33.4.22.3 Pulse width measurement

If the channel (n) is configured to capture rising edges (ELS(n)B:ELS(n)A = 0:1) and the channel (n+1) to capture falling edges (ELS(n+1)B:ELS(n+1)A = 1:0), then the positive polarity pulse width is measured. If the channel (n) is configured to capture falling edges (ELS(n)B:ELS(n)A = 1:0) and the channel (n+1) to capture rising edges (ELS(n +1)B:ELS(n+1)A = 0:1), then the negative polarity pulse width is measured.

The pulse width measurement can be made in one-shot capture mode (One-shot capture mode) or continuous capture mode (Continuous capture mode).

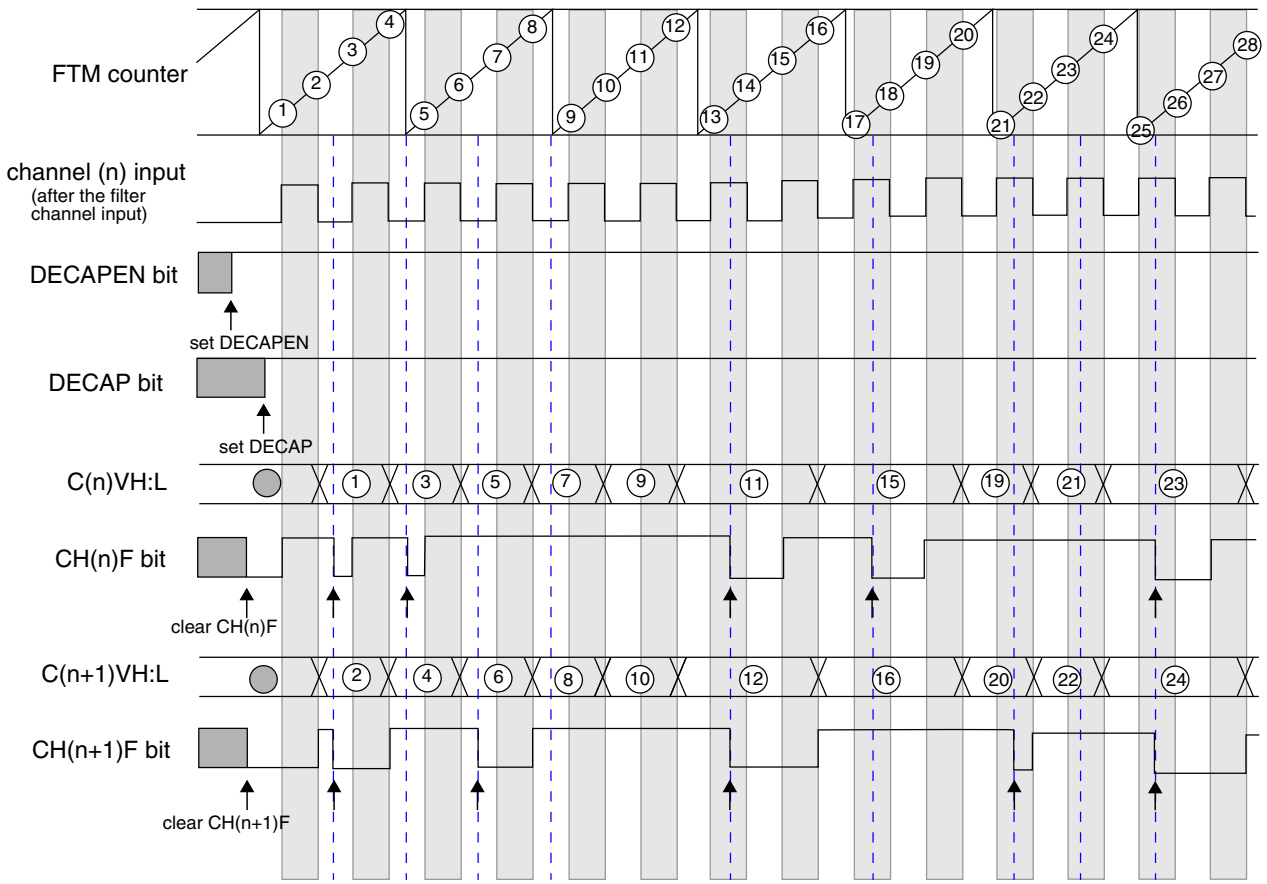
The following figure shows an example of the dual edge capture – one-shot mode used to measure the positive polarity pulse width. The DECAPEN bit selects the dual edge capture mode. The DECAP bit is set to enable the measurement of next positive polarity pulse width. The CH(n)F bit is set when the first edge of this pulse is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set and DECAP bit is cleared when the second edge of this pulse is detected, that is, the edge selected by ELS(n +1)B:ELS(n+1)A bits. Both DECAP and CH(n+1)F bits indicate when two edges of the pulse were captured and the C(n)VH:L and C(n+1)VH:L registers are ready for reading.



Note:
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.
- Problem 1: channel (n) input = 1, set DECAP, not clear CH(n)F, and clear CH(n+1)F.
- Problem 2: channel (n) input = 1, set DECAP, not clear CH(n)F, and not clear CH(n+1)F.

Figure 33-211. Dual edge capture – one-shot mode for positive polarity pulse width measurement

The following figure shows an example of the dual edge capture – continuous mode used to measure the positive polarity pulse width. The DECAPEN bit selects the dual edge capture mode, so it keeps set in all operation mode. While the DECAP bit is set the configured measurements are made. The CH(n)F bit is set when the first edge of the positive polarity pulse is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set when the second edge of this pulse is detected, that is, the edge selected by ELS(n+1)B:ELS(n+1)A bits. The CH(n+1)F bit indicates when two edges of the pulse were captured and the C(n)VH:L and C(n+1)VH:L registers are ready for reading.



Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.

Figure 33-212. Dual edge capture – continuous mode for positive polarity pulse width measurement

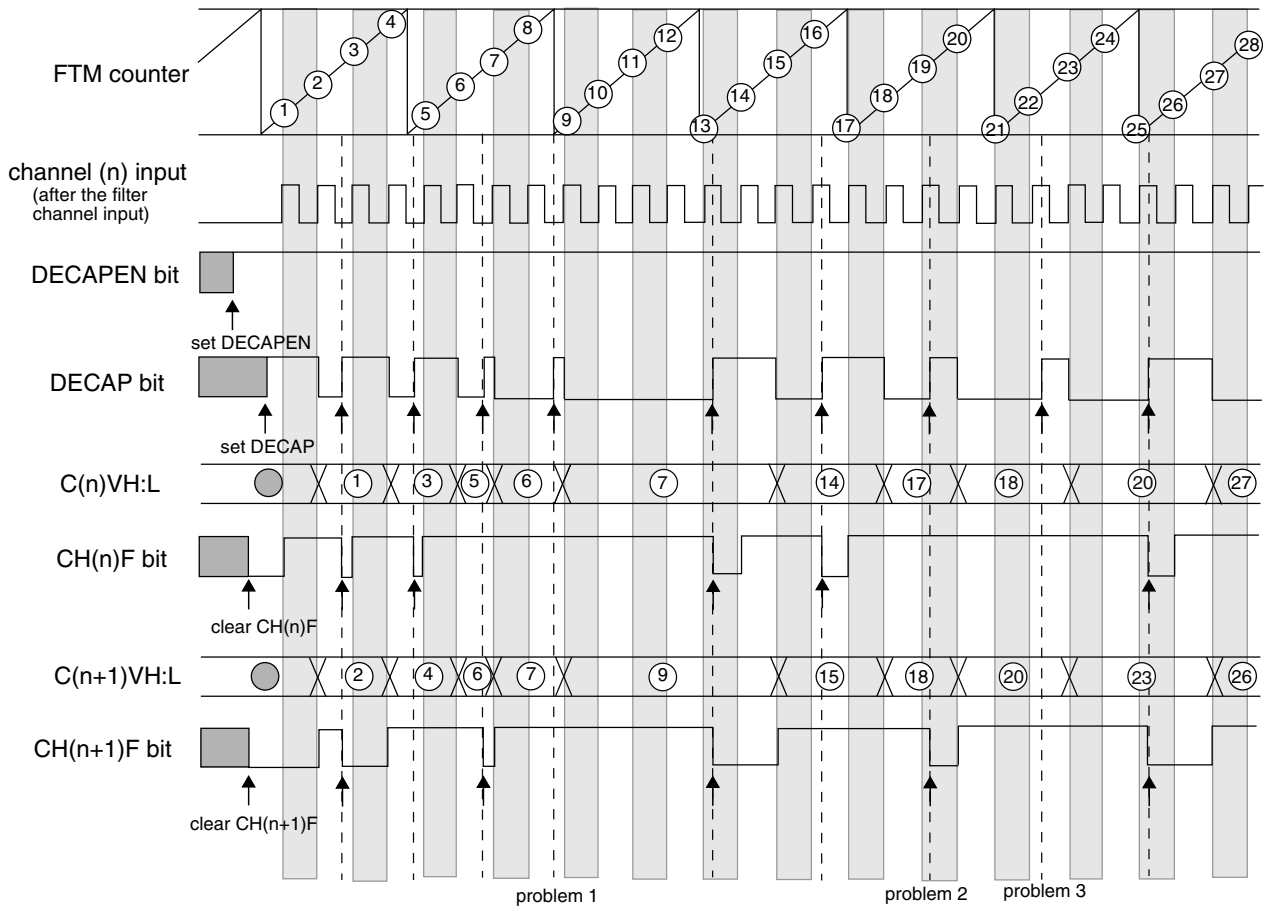
33.4.22.4 Period measurement

If the channels (n) and (n+1) are configured to capture consecutive edges of the same polarity, then the period of the channel (n) input signal is measured. If both channels (n) and (n+1) are configured to capture rising edges (ELS(n)B:ELS(n)A = 0:1 and ELS(n +1)B:ELS(n+1)A = 0:1), then the period between two consecutive rising edges is measured. If both channels (n) and (n+1) are configured to capture falling edges (ELS(n)B:ELS(n)A = 1:0 and ELS(n+1)B:ELS(n+1)A = 1:0), then the period between two consecutive rising edges is measured.

The period measurement can be made in one-shot capture mode (One-shot capture mode) or continuous capture mode (Continuous capture mode).

The following figure shows an example of the dual edge capture – one-shot mode used to measure the period between two consecutive rising edges. The DECAPEN bit selects the dual edge capture mode, so it keeps set in all operation mode. The DECAP bit is set to

enable the measurement of next period. The CH(n)F bit is set when the first rising edge is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set and DECAP bit is cleared when the second rising edge is detected, that is, the edge selected by ELS(n+1)B:ELS(n+1)A bits. Both DECAP and CH(n+1)F bits indicate when two selected edges were captured and the C(n)VH:L and C(n+1)VH:L registers are ready for reading.

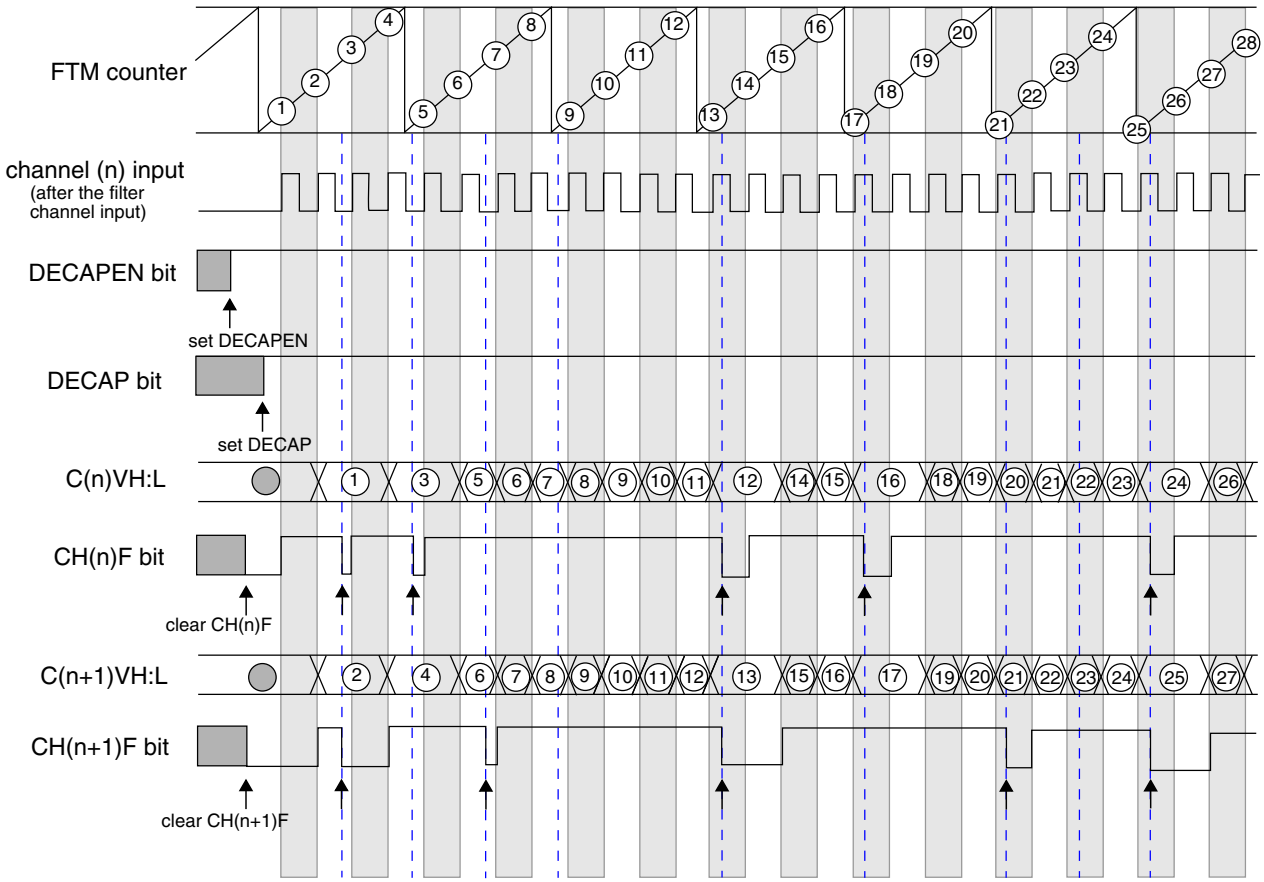


Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.
- Problem 1: channel (n) input = 0, set DECAP, not clear CH(n)F, and not clear CH(n+1)F.
- Problem 2: channel (n) input = 1, set DECAP, not clear CH(n)F, and clear CH(n+1)F.
- Problem 3: channel (n) input = 1, set DECAP, not clear CH(n)F, and not clear CH(n+1)F.

Figure 33-213. Dual edge capture – one-shot mode to measure of the period between two consecutive rising edges

The following figure shows an example of the dual edge capture – continuous mode used to measure the period between two consecutive rising edges. The DECAPEN bit selects the dual edge capture mode, so it keeps set in all operation mode. While the DECAP bit is set the configured measurements are made. The CH(n)F bit is set when the first rising edge is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit

is set when the second rising edge is detected, that is, the edge selected by ELS(n +1)B:ELS(n+1)A bits. The CH(n+1)F bit indicates when two edges of the period were captured and the C(n)VH:L and C(n+1)VH:L registers are ready for reading.



Note:
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.

Figure 33-214. Dual edge capture – continuous mode to measure of the period between two consecutive rising edges

33.4.22.5 Read coherency mechanism

The dual edge capture mode implements a read coherency mechanism between the FTM counter value captured in C(n)VH:L and C(n+1)VH:L registers. The read coherency mechanism is illustrated in the following figure. In this example, the channels (n) and (n +1) are in dual edge capture – continuous mode for positive polarity pulse width measurement. Thus, the channel (n) is configured to capture the FTM counter value when there is a rising edge at channel (n) input signal, and channel (n+1) to capture the FTM counter value when there is a falling edge at channel (n) input signal.

When a rising edge occurs in the channel (n) input signal, the FTM counter value is captured into channel (n) capture buffer. The channel (n) capture buffer value is transferred to C(n)VH:L registers when a falling edge occurs in the channel (n) input signal. C(n)VH:L registers have the FTM counter value when the previous rising edge occurred, and the channel (n) capture buffer has the FTM counter value when the last rising edge occurred.

When a negative edge occurs in the channel (n) input signal, the FTM counter value is captured into channel (n+1) capture buffer. The channel (n+1) capture buffer value is transferred to C(n+1)VH:L registers when the first byte of C(n)VH:L registers is read.

In the following figure, the read of C(n)VH returns the FTM counter high byte value when the event 1 occurred, and the read of C(n+1)VL returns the FTM counter low byte value when the event 1 occurred. The read of C(n+1)VL returns the FTM counter low byte value when the event 2 occurred, and the read of C(n+1)VH returns the FTM counter high byte value when the event 2 occurred.

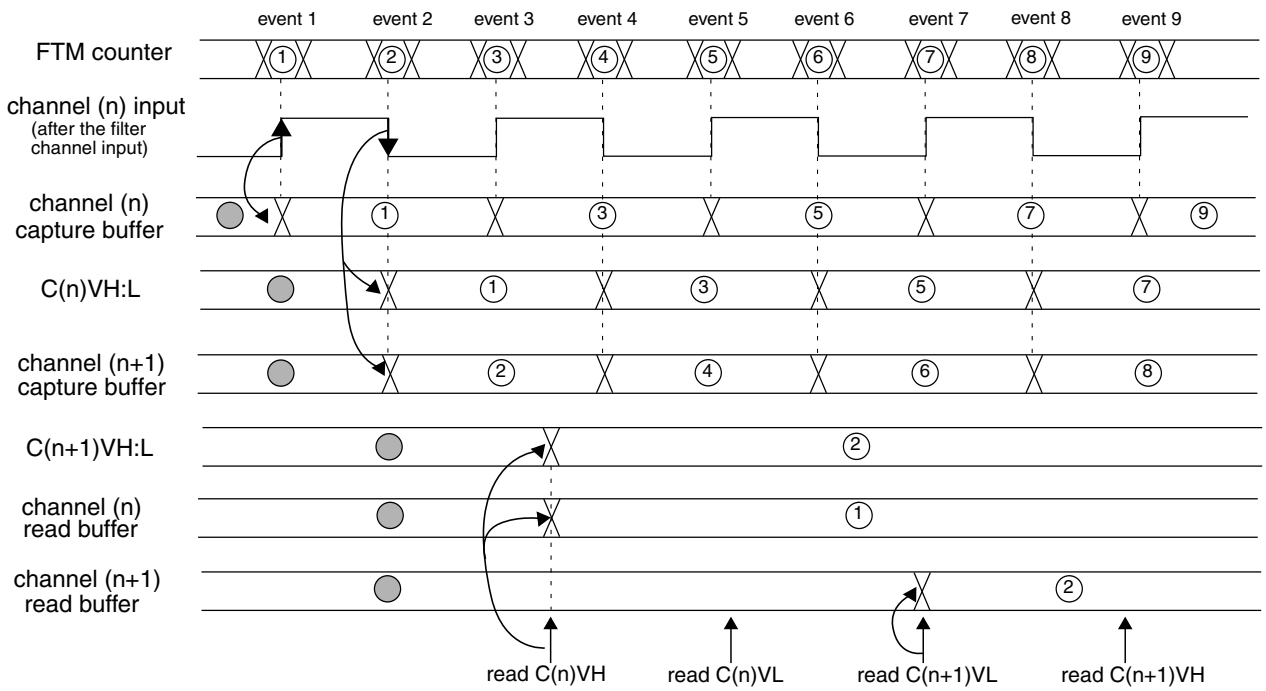


Figure 33-215. Dual edge capture mode read coherency mechanism

C(n)VH:L registers must be read prior to C(n+1)VH:L registers in dual edge capture one-shot and continuous modes for the read coherency mechanism works properly. Either the high or low bytes of C(n)VH:L and C(n+1)VH:L registers can be accessed first; however, the C(n)VH:L registers must be read prior to the C(n+1)VH:L registers in dual edge capture oneshot and continuous modes for the read coherency mechanism to work properly.

33.4.23 Quadrature Decoder Mode

The quadrature decoder mode is selected if FTMEN = 1 and QUADEN = 1 (when the quadrature decoder feature is supported). The quadrature decoder mode uses the input signals phase A and B to control the FTM counter increment and decrement. The following figure is the quadrature decoder block diagram.

Each one of input signals phase A and B has a filter that is equivalent to the filter used in the channels input (Filter for input capture mode). The phase A input filter is enabled by PHAFLTREN bit and this filter's value is defined by CH0FVAL[3:0] bits (CH(n)FVAL[3:0] bits of FILTER0 register). The phase B input filter is enabled by PHBFLTREN bit and this filter's value is defined by CH1FVAL[3:0] bits (CH(n +1)FVAL[3:0] bits of FILTER0 register).

Except for CH0FVAL[3:0] and CH1FVAL[3:0] bits, no channel logic is used in quadrature decoder mode.

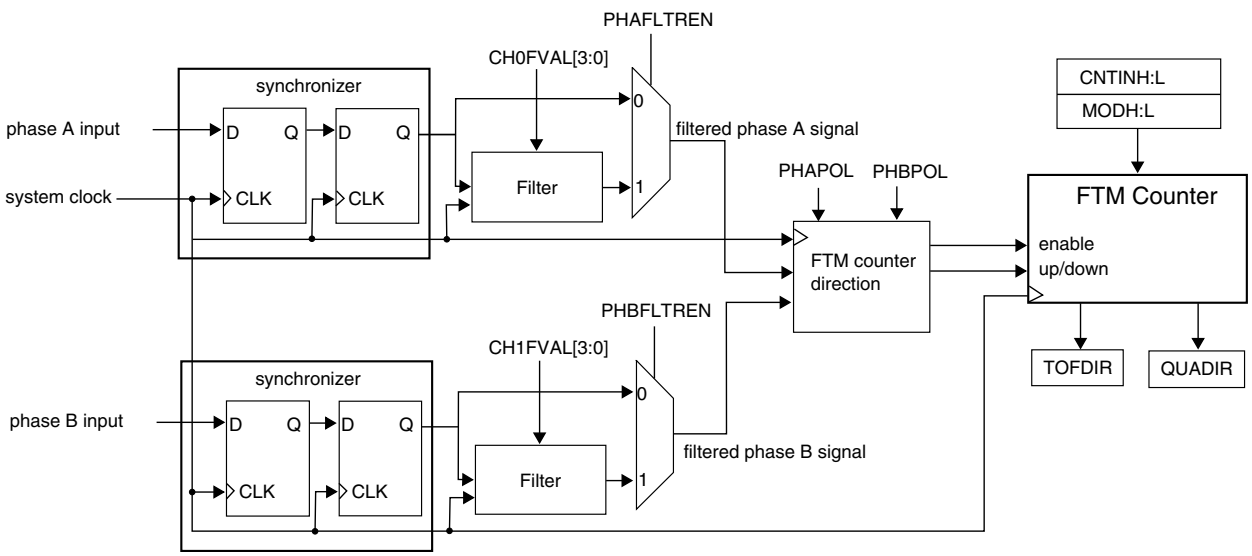


Figure 33-216. Quadrature Decoder Block Diagram

Note

It is important to notice that the FTM counter is clocked by the phase A and B input signals when quadrature decoder mode is selected. Therefore it is expected that the quadrature decoder be used only with the FTM channels in input capture or output compare modes.

The PHAPOL bit selects the polarity of the phase A input, and the PHBPOL bit selects the polarity of the phase B input.

The QUADMODE selects the encoding mode used in the quadrature decoder mode. If QUADMODE = 1, then the count and direction encoding mode (refer to the following figure) is enabled. In this mode, the phase B input value indicates the counting direction (FTM counter increment or decrement), and the phase A input defines the counting rate (FTM counter is updated when there is a rising edge at phase A input signal).

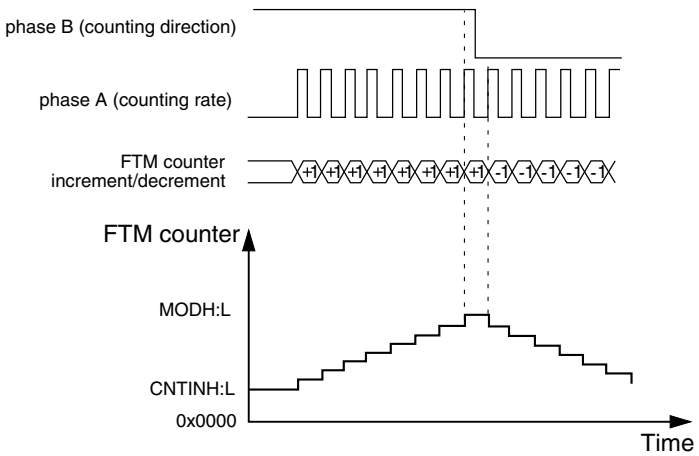


Figure 33-217. Quadrature Decoder – Count and Direction Encoding Mode

If QUADMODE = 0, then the phase A and phase B encoding mode (refer to the following figure) is enabled. In this mode, the relationship between phase A and B signals indicates the counting direction, and phase A and B signals define the counting rate (FTM counter is updated when there is an edge either at the phase A or phase B signals).

If PHAPOL = 0 and PHBPOL = 0, then the FTM counter increment happens when:

- there is a rising edge at phase A signal and phase B signal is at logic zero;
- there is a rising edge at phase B signal and phase A signal is at logic one;
- there is a falling edge at phase B signal and phase A signal is at logic zero;
- there is a falling edge at phase A signal and phase B signal is at logic one;

and the FTM counter decrement happens when:

- there is a falling edge at phase A signal and phase B signal is at logic zero;
- there is a falling edge at phase B signal and phase A signal is at logic one;
- there is a rising edge at phase B signal and phase A signal is at logic zero;
- there is a rising edge at phase A signal and phase B signal is at logic one.

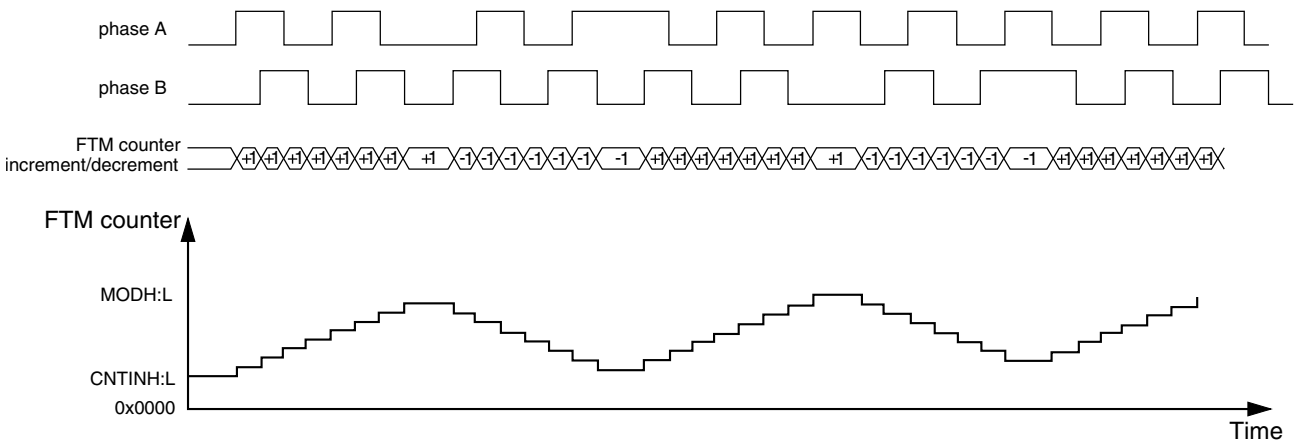


Figure 33-218. Quadrature Decoder – Phase A and Phase B Encoding Mode

The following figure shows the FTM counter overflow in up counting. In this case, when the FTM counter changes from MODH:L to CNTINH:L, the TOF and TOFDIR bits are set. The TOF bit indicates the FTM counter overflow occurred. TOFDIR indicates the counting was up when the FTM counter overflow occurred.

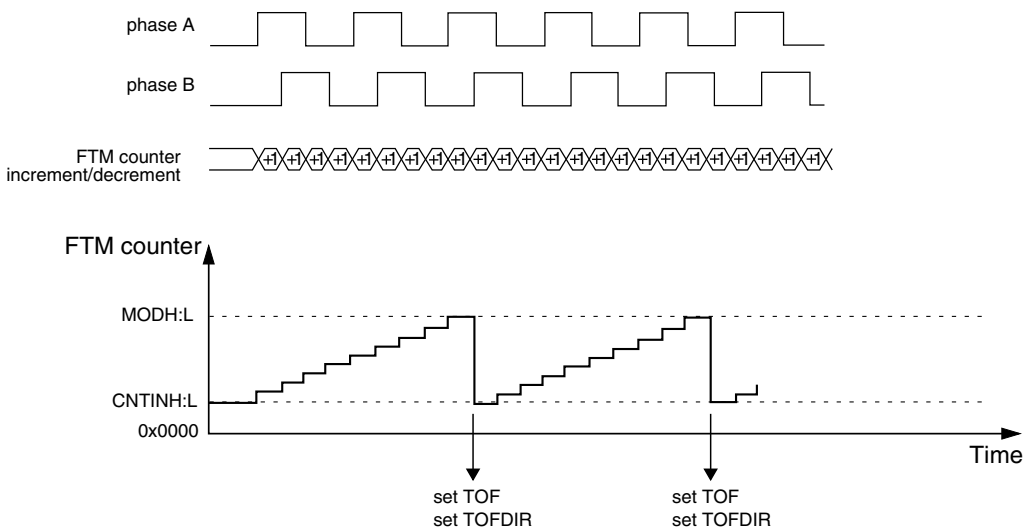


Figure 33-219. FTM Counter Overflow in Up Counting for Quadrature Decoder Mode

The following figure shows the FTM counter overflow in down counting. In this case, when the FTM counter changes from CNTINH:L to MODH:L, the TOF bit is set and the TOFDIR bit is cleared. The TOF bit indicates the FTM counter overflow occurred. TOFDIR indicates the counting was down when the FTM counter overflow occurred.

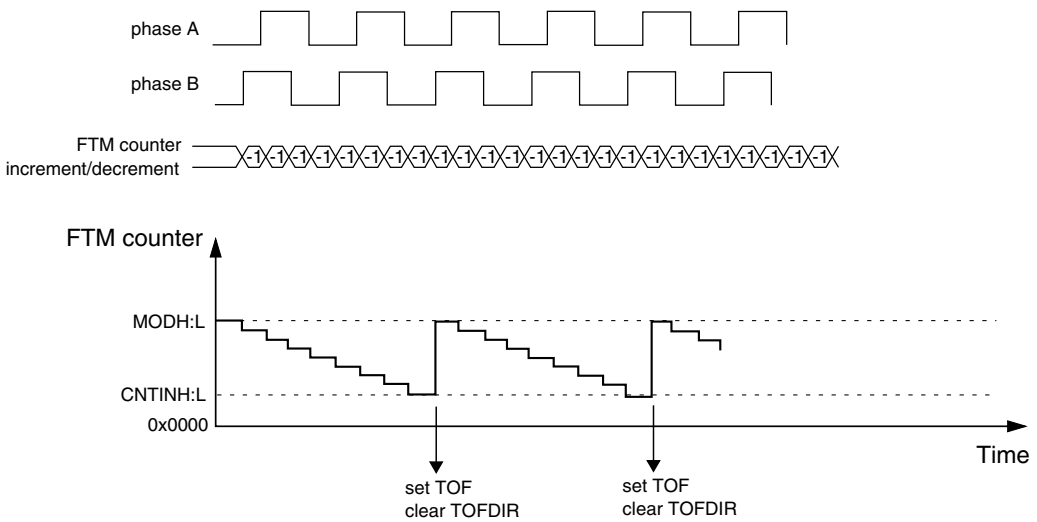


Figure 33-220. FTM Counter Overflow in Down Counting for Quadrature Decoder Mode

33.4.23.1 Quadrature Decoder Boundary Conditions

The following two figures illustrate examples of motor jittering that cause FTM counter transitions. Motor position control applications are expected to observe these behaviors.

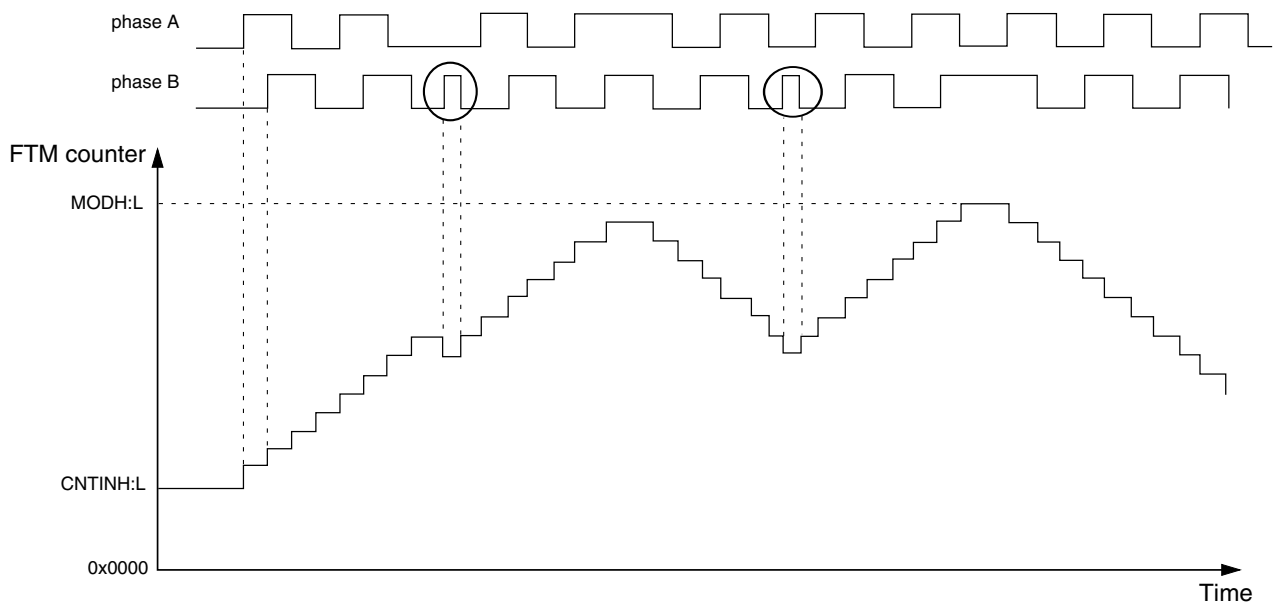


Figure 33-221. Motor Position Jittering in a Mid Count Value

The following figure shows motor jittering produced by the phase B and A pulses, respectively. The first highlighted transition causes a jitter on the FTM counter value near the maximum count value (MODH:L). The second indicated transition occurs on phase A and causes the FTM counter transition between the maximum and minimum count values, which are defined by the MODH:L and CNTINH:L registers.

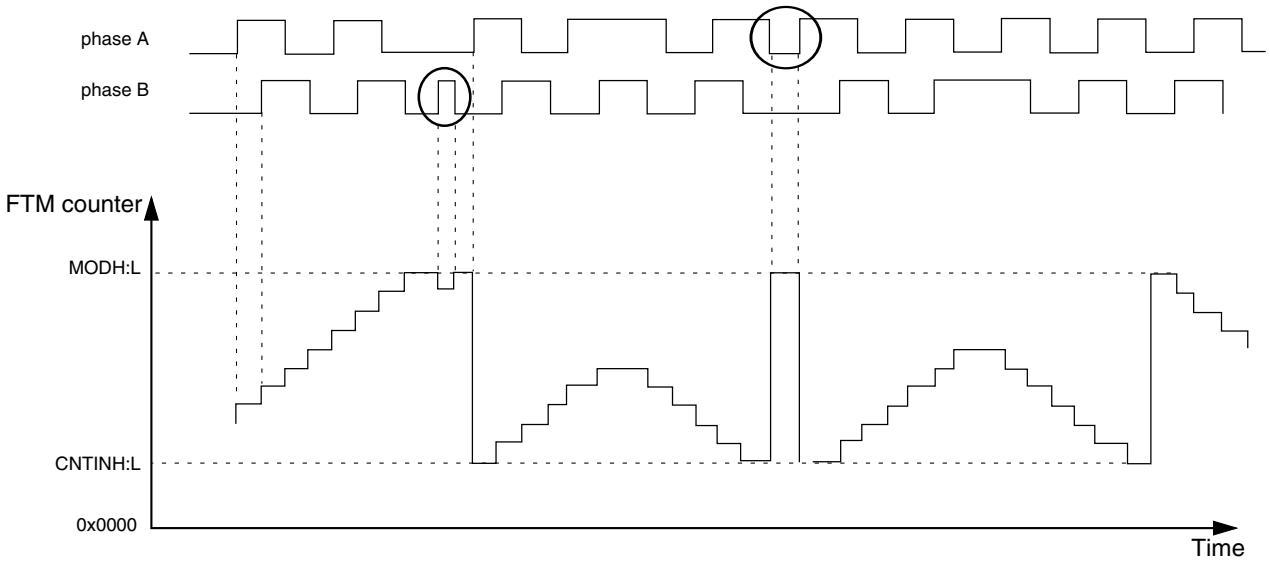


Figure 33-222. Motor Position Jittering Near Maximum and Minimum Count Value

The appropriate settings of the phase A and phase B input filters are important to avoid glitches that may cause oscillation on the FTM counter value. The two preceding figures show examples of oscillations that can be caused by poor input filter setup. To avoid these oscillations, guarantee a minimum pulse width.

33.4.24 TPM emulation

This section describe the FTM features that are selected according to the FTMEN bit.

33.4.24.1 MODH:L and CnVH:L synchronization

If (FTMEN = 0), then the MODH:L and CnVH:L registers are updated according to the Update of the registers with write buffers and they are not updated by PWM synchronization.

If (FTMEN = 1), then the MODH:L and CnVH:L registers are updated only by PWM synchronization (PWM synchronization).

33.4.24.2 Free running counter

If (FTMEN = 0), then the FTM counter is a free running counter when (MODH:L = 0x0000) or (MODH:L = 0xFFFF).

If (FTMEN = 1), then the FTM counter is a free running counter when (CPWMS = 0), (CNTINH:L = 0x0000), and (MODH:L = 0xFFFF).

33.4.24.3 Write to SC

If (FTMEN = 0), then a write to the SC register resets the write coherency mechanism of MODH:L registers.

If (FTMEN = 1), then a write to the SC register does not reset the write coherency mechanism of MODH:L registers.

33.4.24.4 Write to CnSC

If (FTMEN = 0), then a write to the CnSC register resets the write coherency mechanism of CnVH:L registers.

If (FTMEN = 1), then a write to the CnSC register does not reset the write coherency mechanism of CnVH:L registers.

33.4.25 BDM mode

When BDM mode is active, the FlexTimer counter and the channels output are frozen.

However, the value of FlexTimer counter or the channels output are modified in BDM mode when:

- A write of any value to the CNTH or CNTL registers (Counter reset) resets the FTM counter to the value of CNTINH:L and the channels output to their initial value, except for channels in output compare mode.
- The PWM synchronization with REINIT = 1 (see FTM counter synchronization) resets the FTM counter to the value of CNTINH:L registers and the channels output to their initial value, except for channels in output compare mode.
- The initialization (Initialization) forces the value of the CHnOI bit to the channel (n) output.

Note

Do not use the above cases together with fault control (Fault control). If fault control is enabled and the fault condition is at the enabled fault input, these cases reset the FTM counter to the CNTINH:L value and the channels output to their initial value.

33.5 Reset overview

The FTM is reset whenever any chip reset occurs.

When the FTM exits from reset:

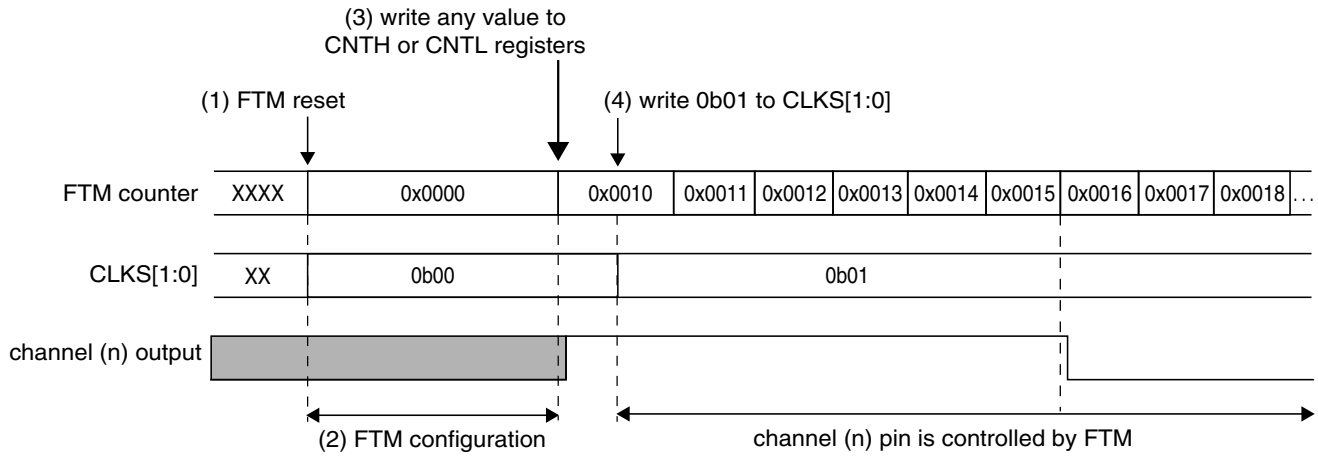
- The FTM counter and the prescaler counter are zero and are stopped (CLKS[1:0] = 0b00)
- The timer overflow interrupt is zero (Timer overflow interrupt)
- The channels interrupts are zero (Channel (n) interrupt)
- The fault interrupt is zero (Fault interrupt)
- The channels are in input capture mode (Input capture mode)
- The channels outputs are zero
- The channels pins are not controlled by FTM (ELS(n)B:ELS(n)A = 0b00). See table "Mode, Edge, and Level Selection"

The following figure shows the FTM behavior after the reset. At the reset (item 1), the FTM counter is disabled (see table "FTM Clock Source Selection"), its value is updated to zero and the pins are not controlled by FTM (table "Mode, Edge, and Level Selection").

After the reset, the FTM should be configured (item 2). It is necessary to define the FTM counter mode, the FTM counting limits (MODH:L and CNTINH:L registers value), the channels mode and CnVH:L registers value according to the channels mode.

Because of this, you should write any value to CNTH or CNTL registers (item 3). This write updates the FTM counter with the value of CNTINH:L and the channels output with its initial value (except for channels in output compare mode) (Counter reset).

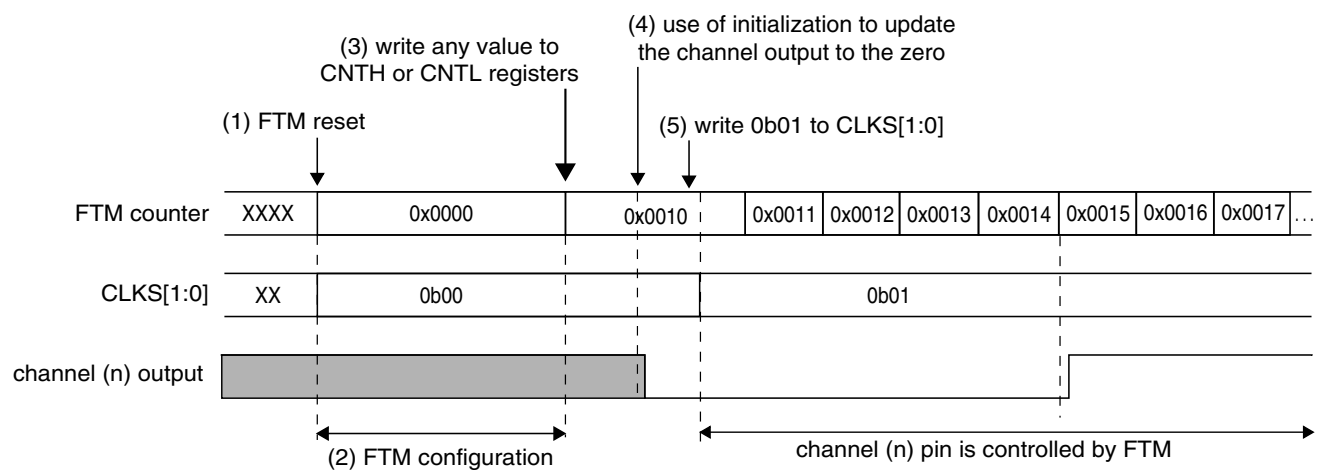
The next step is to select the FTM counter clock by the CLKS[1:0] bits (item 4). It is important to highlight that the pins are controlled only by FTM when CLKS[1:0] bits are different from zero (table "Mode, Edge, and Level Selection").



Note
– CNTINH:L = 0x0010
– Channel (n) is in low-true combine mode with CNTINH:L < C(n)VH:L < C(n+1)VH:L < MODH:L
– C(n)VH:L = 0x0015

Figure 33-223. FTM behavior after the reset when the channel (n) is in combine mode

The following figure shows an example when the channel (n) is in output compare mode and the channel (n) output is toggled when there is a match. In the output compare mode, the channel output is not updated to its initial value when there is a write to CNTH or CNTL registers (item 3). In this case, it is recommended to use the initialization (Initialization) to update the channel output to the selected value (item 4).



Note
– CNTINH:L = 0x0010
– Channel (n) is in output compare and the channel (n) output is toggled when there is a match
– C(n)VH:L = 0x0014

Figure 33-224. FTM behavior after the reset when the channel (n) is in output compare mode

33.6 FTM Interrupts

33.6.1 Timer overflow interrupt

The timer overflow interrupt is generated when (TOIE = 1) and (TOF = 1).

33.6.2 Channel (n) interrupt

The channel (n) interrupt is generated when (CHnIE = 1) and (CHnF = 1).

33.6.3 Fault interrupt

The fault interrupt is generated when (FAULTIE = 1) and (FAULTF = 1).

Chapter 34
Serial Peripheral Interface (SPI)

34.1 Introduction

The serial peripheral interface (SPI) module provides for full-duplex, synchronous, serial communication between the MCU and peripheral devices. These peripheral devices can include other microcontrollers, analog-to-digital converters, shift registers, sensors, and memories, among others.

The SPI runs at a baud rate up to the bus clock divided by two in master mode and up to the bus clock divided by four in slave mode. Software can poll the status flags, or SPI operation can be interrupt driven.

NOTE

For the actual maximum SPI baud rate, refer to the Chip Configuration details and to the device's Data Sheet.

The SPI also supports a data length of 8 or 16 bits and includes a hardware match feature for the receive data buffer.

The SPI includes an internal DMA interface to support continuous SPI transmission through an on-chip DMA controller instead of through the CPU. This feature decreases CPU loading, allowing CPU time to be used for other work.

34.1.1 Features

The SPI includes these distinctive features:

- Master mode or slave mode operation
- Full-duplex or single-wire bidirectional mode
- Programmable transmit bit rate
- Double-buffered transmit and receive data register

- Serial clock phase and polarity options
- Slave select output
- Mode fault error flag with CPU interrupt capability
- Control of SPI operation during wait mode
- Selectable MSB-first or LSB-first shifting
- Programmable 8- or 16-bit data transmission length
- Receive data buffer hardware match feature
- 64-bit FIFO mode for high speed/large amounts of data transfers
- Support transmission of both Transmit and Receive by DMA

34.1.2 Modes of Operation

The SPI functions in three modes, run, wait, and stop.

- Run Mode

  This is the basic mode of operation.

- Wait Mode

  SPI operation in wait mode is a configurable low power mode, controlled by the SPISWAI bit located in the SPIx_C2 register. In wait mode, if the SPISWAI bit is clear, the SPI operates like in Run Mode. If the SPISWAI bit is set, the SPI goes into a power conservative state, with the SPI clock generation turned off. If the SPI is configured as a master, any transmission in progress stops, but is resumed after CPU enters run mode. If the SPI is configured as a slave, reception and transmission of a byte continues, so that the slave stays synchronized to the master.

- Stop Mode

  To reduce power consumption, the SPI is inactive in stop modes where the peripheral bus clock is stopped but internal logic states are retained. If the SPI is configured as a master, any transmission in progress stops, but is resumed after the CPU enters run mode. If the SPI is configured as a slave, reception and transmission of a data continues, so that the slave stays synchronized to the master.

The SPI is completely disabled in stop modes where the peripheral bus clock is stopped and internal logic states are not retained. When the CPU wakes from these stop modes, all SPI register content is reset.

Detailed descriptions of operating modes appear in Low Power Mode Options.

34.1.3 Block Diagrams

This section includes block diagrams showing SPI system connections, the internal organization of the SPI module, and the SPI clock dividers that control the master mode bit rate.

34.1.3.1 SPI System Block Diagram

The following figure shows the SPI modules of two MCUs connected in a master-slave arrangement. The master device initiates all SPI data transfers. During a transfer, the master shifts data out (on the MOSI pin) to the slave while simultaneously shifting data in (on the MISO pin) from the slave. The transfer effectively exchanges the data that was in the SPI shift registers of the two SPI systems. The SPSCK signal is a clock output from the master and an input to the slave. The slave device must be selected by a low level on the slave select input (SS pin). In this system, the master device has configured its SS pin as an optional slave select output.

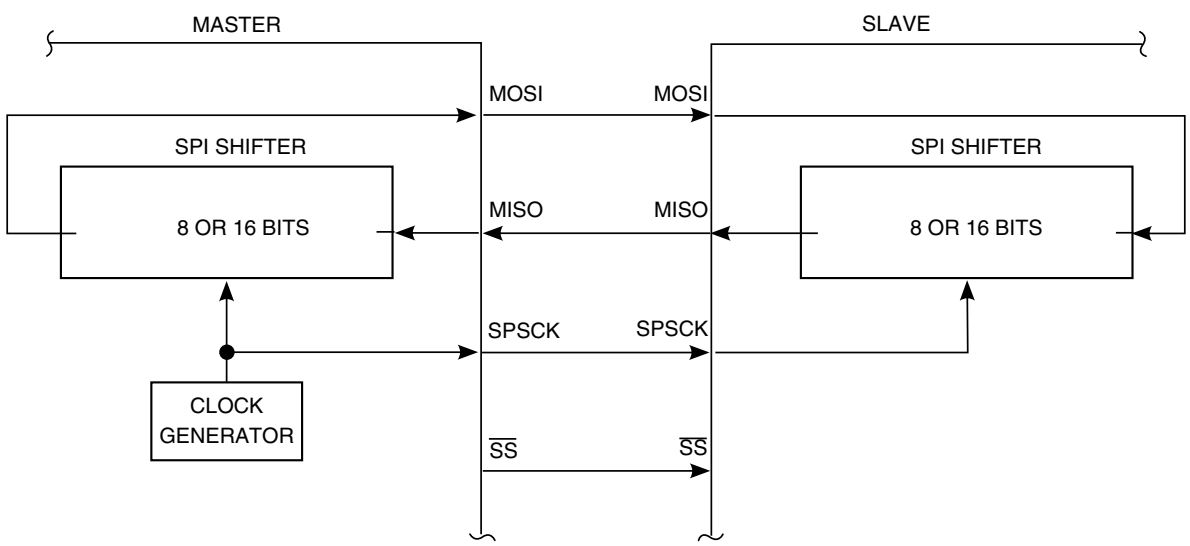


Figure 34-1. SPI System Connections

34.1.3.2 SPI Module Block Diagram

The following is a block diagram of the SPI module. The central element of the SPI is the SPI shift register. Data is written to the double-buffered transmitter (write to SPIx_DH:SPIx_DL) and gets transferred to the SPI shift register at the start of a data transfer. After shifting in 8 bits or 16 bits (as determined by the SPIMODE bit) of data, the data is transferred into the double-buffered receiver where it can be read from SPIx_DH:SPIx_DL. Pin multiplexing logic controls connections between MCU pins and the SPI module.

When the FIFO feature is supported: Additionally there is an 8-byte receive FIFO and an 8-byte transmit FIFO that (once enabled) provide features to allow fewer CPU interrupts to occur when transmitting/receiving high volume/high speed data. When FIFO mode is enabled, the SPI can still function in either 8-bit or 16-bit mode (as per SPIMODE bit) and three additional flags help monitor the FIFO status. Two of these flags can provide CPU interrupts.

When the SPI is configured as a master, the clock output is routed to the SPSCK pin, the shifter output is routed to MOSI, and the shifter input is routed from the MISO pin.

When the SPI is configured as a slave, the SPSCK pin is routed to the clock input of the SPI, the shifter output is routed to MISO, and the shifter input is routed from the MOSI pin.

In the external SPI system, simply connect all SPSCK pins to each other, all MISO pins together, and all MOSI pins together. Peripheral devices often use slightly different names for these pins.

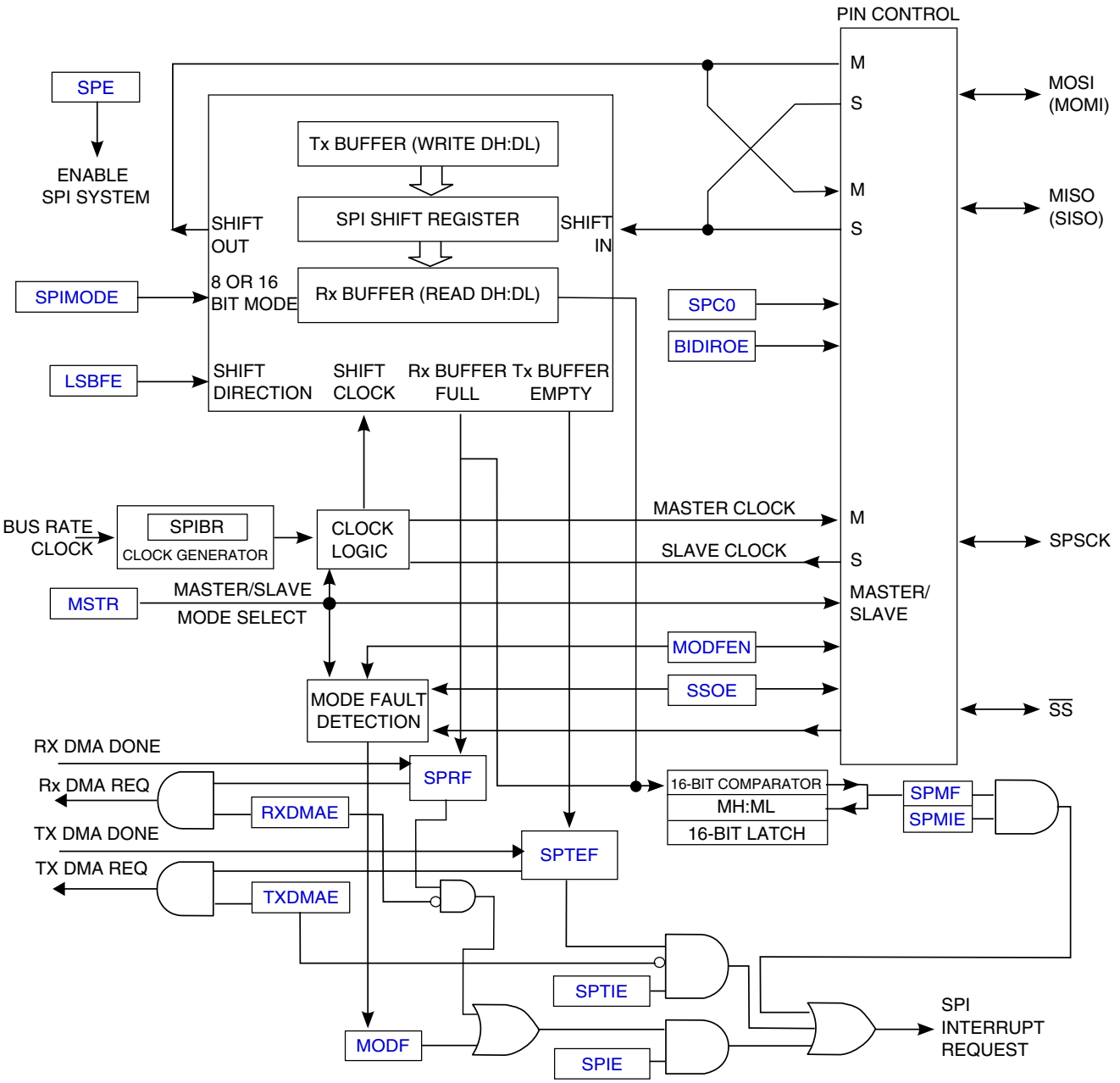


Figure 34-2. SPI Module Block Diagram without FIFO

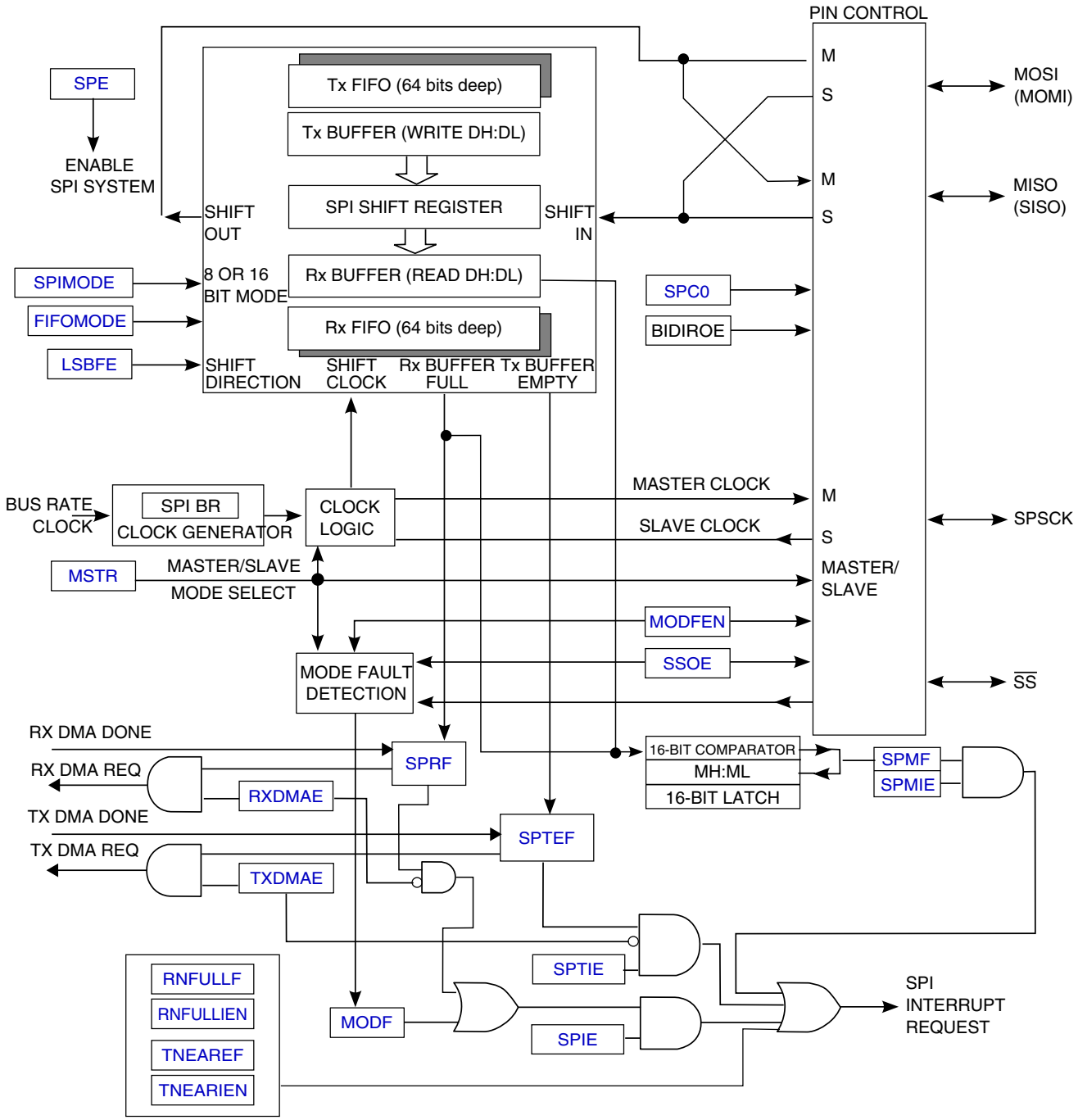


Figure 34-3. SPI Module Block Diagram with FIFO

34.2 External Signal Description

The SPI optionally shares four port pins. The function of these pins depends on the settings of SPI control bits. When the SPI is disabled (SPE = 0), these four pins revert to other functions that are not controlled by the SPI (based on chip configuration).

34.2.1 SPSCK — SPI Serial Clock

When the SPI is enabled as a slave, this pin is the serial clock input. When the SPI is enabled as a master, this pin is the serial clock output.

34.2.2 MOSI — Master Data Out, Slave Data In

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data output. When the SPI is enabled as a slave and SPC0 is 0, this pin is the serial data input. If SPC0 is 1 to select single-wire bidirectional mode, and master mode is selected, this pin becomes the bidirectional data I/O pin (MOMI). Also, the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE is 0) or an output (BIDIROE is 1). If SPC0 is 1 and slave mode is selected, this pin is not used by the SPI and reverts to other functions (based on chip configuration).

34.2.3 MISO — Master Data In, Slave Data Out

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data input. When the SPI is enabled as a slave and SPC0 is 0, this pin is the serial data output. If SPC0 is 1 to select single-wire bidirectional mode, and slave mode is selected, this pin becomes the bidirectional data I/O pin (SISO), and the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE is 0) or an output (BIDIROE is 1). If SPC0 is 1 and master mode is selected, this pin is not used by the SPI and reverts to other functions (based on chip configuration).

34.2.4 $\overline{SS}$ — Slave Select

When the SPI is enabled as a slave, this pin is the low-true slave select input. When the SPI is enabled as a master and mode fault enable is off (MODFEN is 0), this pin is not used by the SPI and reverts to other functions (based on chip configuration). When the SPI is enabled as a master and MODFEN is 1, the slave select output enable bit determines whether this pin acts as the mode fault input (SSOE is 0) or as the slave select output (SSOE is 1).

34.3 Memory Map and Register Descriptions

The SPI has 8-bit registers to select SPI options, to control baud rate, to report SPI status, to hold an SPI data match value, and for transmit/receive data.

SPI memory map

Address offset (hex)	Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	FFFF_8190	SPI control register 1 (SPI0_C1)	8	R/W	04h	34.3.1/715
1	FFFF_8191	SPI control register 2 (SPI0_C2)	8	R/W	00h	34.3.2/717
2	FFFF_8192	SPI baud rate register (SPI0_BR)	8	R/W	00h	34.3.3/718
3	FFFF_8193	SPI status register (SPI0_S)	8	R	20h	34.3.4/719
4	FFFF_8194	SPI data register high (SPI0_DH)	8	R/W	00h	34.3.5/723
5	FFFF_8195	SPI data register low (SPI0_DL)	8	R/W	00h	34.3.6/723
6	FFFF_8196	SPI match register high (SPI0_MH)	8	R/W	00h	34.3.7/724
7	FFFF_8197	SPI match register low (SPI0_ML)	8	R/W	00h	34.3.8/724
8	FFFF_8198	SPI control register 3 (SPI0_C3)	8	R/W	00h	34.3.9/725
9	FFFF_8199	SPI clear interrupt register (SPI0_CI)	8	R/W	00h	34.3.10/ 727
0	FFFF_81A0	SPI control register 1 (SPI1_C1)	8	R/W	04h	34.3.1/715
1	FFFF_81A1	SPI control register 2 (SPI1_C2)	8	R/W	00h	34.3.2/717
2	FFFF_81A2	SPI baud rate register (SPI1_BR)	8	R/W	00h	34.3.3/718
3	FFFF_81A3	SPI status register (SPI1_S)	8	R	20h	34.3.4/719
4	FFFF_81A4	SPI data register high (SPI1_DH)	8	R/W	00h	34.3.5/723
5	FFFF_81A5	SPI data register low (SPI1_DL)	8	R/W	00h	34.3.6/723
6	FFFF_81A6	SPI match register high (SPI1_MH)	8	R/W	00h	34.3.7/724
7	FFFF_81A7	SPI match register low (SPI1_ML)	8	R/W	00h	34.3.8/724
8	FFFF_81A8	SPI control register 3 (SPI1_C3)	8	R/W	00h	34.3.9/725
9	FFFF_81A9	SPI clear interrupt register (SPI1_CI)	8	R/W	00h	34.3.10/ 727
0	FFFF_81B0	SPI control register 1 (SPI2_C1)	8	R/W	04h	34.3.1/715
1	FFFF_81B1	SPI control register 2 (SPI2_C2)	8	R/W	00h	34.3.2/717
2	FFFF_81B2	SPI baud rate register (SPI2_BR)	8	R/W	00h	34.3.3/718
3	FFFF_81B3	SPI status register (SPI2_S)	8	R	20h	34.3.4/719
4	FFFF_81B4	SPI data register high (SPI2_DH)	8	R/W	00h	34.3.5/723
5	FFFF_81B5	SPI data register low (SPI2_DL)	8	R/W	00h	34.3.6/723
6	FFFF_81B6	SPI match register high (SPI2_MH)	8	R/W	00h	34.3.7/724
7	FFFF_81B7	SPI match register low (SPI2_ML)	8	R/W	00h	34.3.8/724

Table continues on the next page...

SPI memory map (continued)

Address offset (hex)	Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8	FFFF_81B8	SPI control register 3 (SPI2_C3)	8	R/W	00h	34.3.9/725
9	FFFF_81B9	SPI clear interrupt register (SPI2_CI)	8	R/W	00h	34.3.10/ 727

34.3.1 SPI control register 1 (SPIx_C1)

This read/write register includes the SPI enable control, interrupt enables, and configuration options.

Address: FFFF_8190h base + 0h offset = FFFF_8190h

Bit	7	6	5	4	3	2	1	0
Read Write	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
Reset	0	0	0	0	0	1	0	0

SPI0_C1 field descriptions

Field	Description
7 SPIE	SPI interrupt enable: for SPRF and MODF (when FIFO is not supported or not enabled) or for read FIFO (when FIFO is supported and enabled) When the FIFO is not supported or not enabled (FIFOMODE is not present or is 0): This bit enables the interrupt for SPI receive buffer full (SPRF) and mode fault (MODF) events. When the FIFO is supported and enabled (FIFOMODE is 1): This bit enables the SPI to interrupt the CPU when the receive FIFO is full. An interrupt occurs when the SPRF bit is set or the MODF bit is set. 0 Interrupts from SPRF and MODF are inhibited—use polling (when FIFOMODE is not present or is 0) or Read FIFO Full Interrupts are disabled (when FIFOMODE is 1) 1 Request a hardware interrupt when SPRF or MODF is 1 (when FIFOMODE is not present or is 0) or Read FIFO Full Interrupts are enabled (when FIFOMODE is 1)
6 SPE	SPI system enable This bit enables the SPI system and dedicates the SPI port pins to SPI system functions. If SPE is cleared, the SPI is disabled and forced into an idle state, and all status bits in the S register are reset. 0 SPI system inactive 1 SPI system enabled
5 SPTIE	SPI transmit interrupt enable When the FIFO is not supported or not enabled (FIFOMODE is not present or is 0): This is the interrupt enable bit for SPI transmit buffer empty (SPTEF). An interrupt occurs when the SPI transmit buffer is empty (SPTEF is set). When the FIFO is supported and enabled (FIFOMODE is 1): This is the interrupt enable bit for SPI transmit FIFO empty (SPTEF). An interrupt occurs when the SPI transmit FIFO is empty (SPTEF is set).

Table continues on the next page...

SPI0_C1 field descriptions (continued)

Field	Description
	0 Interrupts from SPTEF inhibited (use polling) 1 When SPTEF is 1, hardware interrupt requested
4 MSTR	Master/slave mode select This bit selects master or slave mode operation. 0 SPI module configured as a slave SPI device 1 SPI module configured as a master SPI device
3 CPOL	Clock polarity This bit selects an inverted or non-inverted SPI clock. To transmit data between SPI modules, the SPI modules must have identical CPOL values. This bit effectively places an inverter in series with the clock signal either from a master SPI device or to a slave SPI device. Refer to the description of “SPI Clock Formats” for details. 0 Active-high SPI clock (idles low) 1 Active-low SPI clock (idles high)
2 CPHA	Clock phase This bit selects one of two clock formats for different kinds of synchronous serial peripheral devices. Refer to the description of “SPI Clock Formats” for details. 0 First edge on SPSCK occurs at the middle of the first cycle of a data transfer 1 First edge on SPSCK occurs at the start of the first cycle of a data transfer
1 SSOE	Slave select output enable This bit is used in combination with the mode fault enable (MODFEN) bit in the C2 register and the master/slave (MSTR) control bit to determine the function of the $\overline{SS}$ pin. 0 When MODFEN is 0: In master mode, $\overline{SS}$ pin function is general-purpose I/O (not SPI). In slave mode, $\overline{SS}$ pin function is slave select input. When MODFEN is 1: In master mode, $\overline{SS}$ pin function is $\overline{SS}$ input for mode fault. In slave mode, $\overline{SS}$ pin function is slave select input. 1 When MODFEN is 0: In master mode, $\overline{SS}$ pin function is general-purpose I/O (not SPI). In slave mode, $\overline{SS}$ pin function is slave select input. When MODFEN is 1: In master mode, $\overline{SS}$ pin function is automatic $\overline{SS}$ output. In slave mode: $\overline{SS}$ pin function is slave select input.
0 LSBFE	LSB first (shifter direction) This bit does not affect the position of the MSB and LSB in the data register. Reads and writes of the data register always have the MSB in bit 7 (or bit 15 in 16-bit mode). 0 SPI serial data transfers start with most significant bit 1 SPI serial data transfers start with least significant bit

34.3.2 SPI control register 2 (SPIx_C2)

This read/write register is used to control optional features of the SPI system.

Address: FFFF_8190h base + 1h offset = FFFF_8191h

Bit	7	6	5	4	3	2	1	0
Read Write	SPMIE	SPIMODE	TXDMAE	MODFEN	BIDIROE	RXDMAE	SPISWAI	SPC0
Reset	0	0	0	0	0	0	0	0

SPI0_C2 field descriptions

Field	Description
7 SPMIE	SPI match interrupt enable This is the interrupt enable bit for the SPI receive data buffer hardware match (SPMF) function. 0 Interrupts from SPMF inhibited (use polling) 1 When SPMF is 1, requests a hardware interrupt
6 SPIMODE	SPI 8-bit or 16-bit mode This bit allows the user to select either an 8-bit or 16-bit SPI data transmission length. In master mode, a change of this bit aborts a transmission in progress, forces the SPI system into an idle state, and resets all status bits in the S register. Refer to the description of “Data Transmission Length” for details. 0 8-bit SPI shift register, match register, and buffers 1 16-bit SPI shift register, match register, and buffers
5 TXDMAE	Transmit DMA enable This bit enables a transmit DMA request. When this bit is set to 1, a transmit DMA request is asserted when both SPTEF and SPE are set, and the interrupt from SPTEF is disabled. 0 DMA request for transmit is disabled and interrupt from SPTEF is allowed 1 DMA request for transmit is enabled and interrupt from SPTEF is disabled
4 MODFEN	Master mode-fault function enable When the SPI is configured for slave mode, this bit has no meaning or effect. (The $\overline{SS}$ pin is the slave select input.) In master mode, this bit determines how the $\overline{SS}$ pin is used. For details, refer to the description of the SSOE bit in the C1 register. 0 Mode fault function disabled, master $\overline{SS}$ pin reverts to general-purpose I/O not controlled by SPI 1 Mode fault function enabled, master $\overline{SS}$ pin acts as the mode fault input or the slave select output
3 BIDIROE	Bidirectional mode output enable When bidirectional mode is enabled because SPI pin control 0 (SPC0) is set to 1, the BIDIROE bit determines whether the SPI data output driver is enabled to the single bidirectional SPI I/O pin. Depending on whether the SPI is configured as a master or a slave, it uses the MOSI (MOMI) or MISO (SISO) pin, respectively, as the single SPI data I/O pin. When SPC0 is 0, BIDIROE has no meaning or effect. 0 Output driver disabled so SPI data I/O pin acts as an input 1 SPI I/O pin enabled as an output

Table continues on the next page...

SPI0_C2 field descriptions (continued)

Field	Description
2 RXDMAE	Receive DMA enable This is the enable bit for a receive DMA request. When this bit is set to 1, a receive DMA request is asserted when both SPRF and SPE are set, and the interrupt from SPRF is disabled. 0 DMA request for receive is disabled and interrupt from SPRF is allowed 1 DMA request for receive is enabled and interrupt from SPRF is disabled
1 SPISWAI	SPI stop in wait mode This bit is used for power conservation while the device is in wait mode. 0 SPI clocks continue to operate in wait mode 1 SPI clocks stop when the MCU enters wait mode
0 SPC0	SPI pin control 0 This bit enables bidirectional pin configurations. 0 SPI uses separate pins for data input and data output (pin mode is normal). In master mode of operation: MISO is master in and MOSI is master out. In slave mode of operation: MISO is slave out and MOSI is slave in. 1 SPI configured for single-wire bidirectional operation (pin mode is bidirectional). In master mode of operation: MISO is not used by SPI; MOSI is master in when BIDIROE is 0 or master I/O when BIDIROE is 1. In slave mode of operation: MISO is slave in when BIDIROE is 0 or slave I/O when BIDIROE is 1; MOSI is not used by SPI.

34.3.3 SPI baud rate register (SPIx_BR)

Use this register to set the prescaler and bit rate divisor for an SPI master. This register may be read or written at any time.

Address: FFFF_8190h base + 2h offset = FFFF_8192h

Bit	7	6	5	4	3	2	1	0
Read	0	SPPR[2:0]			SPR[3:0]			
Write								
Reset	0	0	0	0	0	0	0	0

SPI0_BR field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–4 SPPR[2:0]	SPI baud rate prescale divisor

Table continues on the next page...

SPI0_BR field descriptions (continued)

Field	Description
	This 3-bit field selects one of eight divisors for the SPI baud rate prescaler. The input to this prescaler is the bus rate clock (BUSCLK). The output of this prescaler drives the input of the SPI baud rate divider. Refer to the description of “SPI Baud Rate Generation” for details. 000 Baud rate prescaler divisor is 1 001 Baud rate prescaler divisor is 2 010 Baud rate prescaler divisor is 3 011 Baud rate prescaler divisor is 4 100 Baud rate prescaler divisor is 5 101 Baud rate prescaler divisor is 6 110 Baud rate prescaler divisor is 7 111 Baud rate prescaler divisor is 8
3–0 SPR[3:0]	SPI baud rate divisor This 4-bit field selects one of nine divisors for the SPI baud rate divider. The input to this divider comes from the SPI baud rate prescaler. Refer to the description of “SPI Baud Rate Generation” for details. 0000 Baud rate divisor is 2 0001 Baud rate divisor is 4 0010 Baud rate divisor is 8 0011 Baud rate divisor is 16 0100 Baud rate divisor is 32 0101 Baud rate divisor is 64 0110 Baud rate divisor is 128 0111 Baud rate divisor is 256 1000 Baud rate divisor is 512 All others Reserved

34.3.4 SPI status register (SPIx_S)

This register contains read-only status bits. Writes have no meaning or effect.

NOTE

When the FIFO is not supported or not enabled (FIFOMODE is not present or is 0): Bits 3 through 0 are not implemented and always read 0.

When the FIFO is supported and enabled (FIFOMODE is 1): This register has four flags that provide mechanisms to support an 8-byte FIFO mode: RNFULLF, TNEARF, TXFULLF, and RFIFOEF. When the SPI is in 8-byte FIFO mode, the function of SPRF and SPTEF differs slightly from their function in the normal buffered modes, mainly regarding how these flags are cleared by the amount available in the transmit and receive FIFOs.

- The RNFULLF and TNEAREF help improve the efficiency of FIFO operation when transfering large amounts of data. These flags provide a "watermark" feature of the FIFOs to allow continuous transmissions of data when running at high speed.
- The RNFULLF can generate an interrupt if the RNFULLIEN bit in the C3 register is set, which allows the CPU to start emptying the receive FIFO without delaying the reception of subsequent bytes. The user can also determine if all data in the receive FIFO has been read by monitoring the RFIFOEF.
- The TNEAREF can generate an interrupt if the TNEARIEN bit in the C3 register is set, which allows the CPU to start filling the transmit FIFO before it is empty and thus to prevent breaks in SPI transmission.

NOTE

At an initial POR, the values of TNEAREF and RFIFOEF are 0. However, the status (S) register and both TX and RX FIFOs are reset due to a change of SPIMODE, FIFOMODE or SPE. If this type of reset occurs and FIFOMODE is 0, TNEAREF and RFIFOEF continue to reset to 0. If this type of reset occurs and FIFOMODE is 1, TNEAREF and RFIFOEF reset to 1.

Address: FFFF_8190h base + 3h offset = FFFF_8193h

Bit	7	6	5	4	3	2	1	0
Read	SPRF	SPMF	SPTEF	MODF	RNFULLF	TNEAREF	TXFULLF	RFIFOEF
Write								
Reset	0	0	1	0	0	0	0	0

SPI0_S field descriptions

Field	Description
7 SPRF	SPI read buffer full flag (when FIFO is not supported or not enabled) or SPI read FIFO FULL flag (when FIFO is supported and enabled) When the FIFO is not supported or not enabled (FIFOMODE is not present or is 0): SPRF is set at the completion of an SPI transfer to indicate that received data may be read from the SPI data (DH:DL) register. When the receive DMA request is disabled (RXDMAE is 0), SPRF is cleared by reading SPRF while it is set and then reading the SPI data register. When the receive DMA request is enabled (RXDMAE is 1), SPRF is automatically cleared when the DMA transfer for the receive DMA request is completed (RX DMA Done is asserted). When FIFOMODE is 1: This bit indicates the status of the read FIFO when FIFOMODE is enabled. The SPRF is set when the read FIFO has received 64 bits (4 words or 8 bytes) of data from the shifter and there have been no CPU reads of the SPI data (DH:DL) register. When the receive DMA request is disabled (RXDMAE is 0), SPRF is cleared by reading the SPI data register, which empties the FIFO assuming another SPI message is not received. When the receive DMA request is enabled (RXDMAE is 1), SPRF is automatically cleared when the DMA transfer for the receive DMA request is completed (RX DMA Done is asserted).

Table continues on the next page...

SPI0_S field descriptions (continued)

Field	Description
	0 No data available in the receive data buffer (when FIFOMODE is not present or is 0) or Read FIFO is not full (when FIFOMODE is 1) 1 Data available in the receive data buffer (when FIFOMODE is not present or is 0) or Read FIFO is full (when FIFOMODE is 1)
6 SPMF	SPI match flag SPMF is set after SPRF is 1 when the value in the receive data buffer matches the value in the MH:ML registers. To clear the flag, read SPMF when it is set and then write a 1 to it. 0 Value in the receive data buffer does not match the value in the MH:ML registers 1 Value in the receive data buffer matches the value in the MH:ML registers
5 SPTEF	SPI transmit buffer empty flag (when FIFO is not supported or not enabled) or SPI transmit FIFO empty flag (when FIFO is supported and enabled) When the FIFO is not supported or not enabled (FIFOMODE is not present or is 0): This bit is set when the transmit data buffer is empty. When the transmit DMA request is disabled (TXDMAE is 0), SPTEF is cleared by reading the S register with SPTEF set and then writing a data value to the transmit buffer at DH:DL. The S register must be read with SPTEF set to 1 before writing data to the DH:DL register; otherwise, the DH:DL write is ignored. When the transmit DMA request is enabled (TXDMAE is 1), SPTEF is automatically cleared when the DMA transfer for the transmit DMA request is completed (TX DMA Done is asserted). SPTEF is automatically set when all data from the transmit buffer transfers into the transmit shift register. For an idle SPI, data written to DH:DL is transferred to the shifter almost immediately so that SPTEF is set within two bus cycles, allowing a second set of data to be queued into the transmit buffer. After completion of the transfer of the data in the shift register, the queued data from the transmit buffer automatically moves to the shifter, and SPTEF is set to indicate that room exists for new data in the transmit buffer. If no new data is waiting in the transmit buffer, SPTEF simply remains set and no data moves from the buffer to the shifter. When the FIFO is not supported or not enabled (FIFOMODE is not present or is 0): If a transfer does not stop, the last data that was transmitted is sent out again. When the FIFO is supported and enabled (FIFOMODE is 1): This bit provides the status of the FIFO rather than of an 8-bit or a 16-bit buffer. This bit is set when the transmit FIFO is empty. When the transmit DMA request is disabled (TXDMAE is 0), SPTEF is cleared by writing a data value to the transmit FIFO at DH:DL. When the transmit DMA request is enabled (TXDMAE is 1), SPTEF is automatically cleared when the DMA transfer for the transmit DMA request is completed (TX DMA Done is asserted). SPTEF is automatically set when all data from the transmit FIFO transfers into the transmit shift register. For an idle SPI, data written to the DH:DL register is transferred to the shifter almost immediately, so that SPTEF is set within two bus cycles. A second write of data to the DH:DL register clears this SPTEF flag. After completion of the transfer of the data in the shift register, the queued data from the transmit FIFO automatically moves to the shifter, and SPTEF will be set only when all data written to the transmit FIFO has been transfered to the shifter. If no new data is waiting in the transmit FIFO, SPTEF simply remains set and no data moves from the buffer to the shifter. 0 SPI transmit buffer not empty (when FIFOMODE is not present or is 0) or SPI FIFO not empty (when FIFOMODE is 1) 1 SPI transmit buffer empty (when FIFOMODE is not present or is 0) or SPI FIFO empty (when FIFOMODE is 1)
4 MODF	Master mode fault flag MODF is set if the SPI is configured as a master and the slave select input goes low, indicating some other SPI device is also configured as a master. The $\overline{SS}$ pin acts as a mode fault error input only when MSTR is 1, MODFEN is 1, and SSOE is 0; otherwise, MODF will never be set. MODF is cleared by reading MODF while it is 1 and then writing to the SPI control register 1 (C1).

Table continues on the next page...

SPI0_S field descriptions (continued)

Field	Description
	0 No mode fault error 1 Mode fault error detected
3 RNFULLF	Receive FIFO nearly full flag This flag is set when more than three 16-bit words or six 8-bit bytes of data remain in the receive FIFO, provided C3[4] is 0, or when more than two 16-bit words or four 8-bit bytes of data remain in the receive FIFO, provided C3[4] is 1. It has no function if FIFOMODE is not present or is 0. 0 Receive FIFO has received less than 48 bits (when C3[4] is 0) or less than 32 bits (when C3[4] is 1) 1 Receive FIFO has received data of an amount equal to or greater than 48 bits (when C3[4] is 0) or 32 bits (when C3[4] is 1)
2 TNEAREF	Transmit FIFO nearly empty flag This flag is set when only one 16-bit word or two 8-bit bytes of data remain in the transmit FIFO, provided C3[5] is 0, or when only two 16-bit words or four 8-bit bytes of data remain in the transmit FIFO, provided C3[5] is 1. If FIFOMODE is not enabled, ignore this bit. **NOTE:** At an initial POR, the values of TNEAREF and RFIFOEF are 0. However, the status (S) register and both TX and RX FIFOs are reset due to a change of SPIMODE, FIFOMODE or SPE. If this type of reset occurs and FIFOMODE is 0, TNEAREF and RFIFOEF continue to reset to 0. If this type of reset occurs and FIFOMODE is 1, TNEAREF and RFIFOEF reset to 1. 0 Transmit FIFO has more than 16 bits (when C3[5] is 0) or more than 32 bits (when C3[5] is 1) remaining to transmit 1 Transmit FIFO has an amount of data equal to or less than 16 bits (when C3[5] is 0) or 32 bits (when C3[5] is 1) remaining to transmit
1 TXFULLF	Transmit FIFO full flag This bit indicates the status of the transmit FIFO when FIFOMODE is enabled. This flag is set when there are 8 bytes in the transmit FIFO. If FIFOMODE is not enabled, ignore this bit. 0 Transmit FIFO has less than 8 bytes 1 Transmit FIFO has 8 bytes of data
0 RFIFOEF	SPI read FIFO empty flag This bit indicates the status of the read FIFO when FIFOMODE is enabled. If FIFOMODE is not enabled, ignore this bit. **NOTE:** At an initial POR, the values of TNEAREF and RFIFOEF are 0. However, the status (S) register and both TX and RX FIFOs are reset due to a change of SPIMODE, FIFOMODE or SPE. If this type of reset occurs and FIFOMODE is 0, TNEAREF and RFIFOEF continue to reset to 0. If this type of reset occurs and FIFOMODE is 1, TNEAREF and RFIFOEF reset to 1. 0 Read FIFO has data. Reads of the DH:DL registers in 16-bit mode or the DL register in 8-bit mode will empty the read FIFO. 1 Read FIFO is empty.

34.3.5 SPI data register high (SPIx_DH)

Refer to the description of the DL register.

Address: FFFF_8190h base + 4h offset = FFFF_8194h

Bit	7	6	5	4	3	2	1	0
Read Write	Bits[15:8]							
Reset	0	0	0	0	0	0	0	0

SPI0_DH field descriptions

Field	Description
7–0 Bits[15:8]	Data (high byte)

34.3.6 SPI data register low (SPIx_DL)

This register, together with the DH register, is both the input and output register for SPI data. A write to the registers writes to the transmit data buffer, allowing data to be queued and transmitted.

When the SPI is configured as a master, data queued in the transmit data buffer is transmitted immediately after the previous transmission has completed.

The SPTEF bit in the S register indicates when the transmit data buffer is ready to accept new data. When the transmit DMA request is disabled (TXDMAE is 0): The S register must be read when SPTEF is set before writing to the SPI data registers; otherwise, the write is ignored. When the transmit DMA request is enabled (TXDMAE is 1) when SPTEF is set, the SPI data registers can be written automatically by DMA without reading the S register first.

Data may be read from the SPI data registers any time after SPRF is set and before another transfer is finished. Failure to read the data out of the receive data buffer before a new transfer ends causes a receive overrun condition, and the data from the new transfer is lost. The new data is lost because the receive buffer still held the previous character and was not ready to accept the new data. There is no indication for a receive overrun condition, so the application system designer must ensure that previous data has been read from the receive buffer before a new transfer is initiated.

In 8-bit mode, only the DL register is available. Reads of the DH register return all zeros. Writes to the DH register are ignored.

In 16-bit mode, reading either byte (the DH or DL register) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. Writing to either byte (the DH or DL register) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent 16-bit value into the transmit data buffer.

Address: FFFF_8190h base + 5h offset = FFFF_8195h

Bit	7	6	5	4	3	2	1	0
Read Write	Bits[7:0]							
Reset	0	0	0	0	0	0	0	0

SPI0_DL field descriptions

Field	Description
7–0 Bits[7:0]	Data (low byte)

34.3.7 SPI match register high (SPIx_MH)

Refer to the description of the ML register.

Address: FFFF_8190h base + 6h offset = FFFF_8196h

Bit	7	6	5	4	3	2	1	0
Read Write	Bits[15:8]							
Reset	0	0	0	0	0	0	0	0

SPI0_MH field descriptions

Field	Description
7–0 Bits[15:8]	Hardware compare value (high byte)

34.3.8 SPI match register low (SPIx_ML)

This register, together with the MH register, contains the hardware compare value. When the value received in the SPI receive data buffer equals this hardware compare value, the SPI match flag (SPMF) sets.

In 8-bit mode, only the ML register is available. Reads of the MH register return all zeros. Writes to the MH register are ignored.

In 16-bit mode, reading either byte (the MH or ML register) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. Writing to either byte (the MH or ML register) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent value into the SPI match registers.

Address: FFFF_8190h base + 7h offset = FFFF_8197h

Bit	7	6	5	4	3	2	1	0
Read Write	Bits[7:0]							
Reset	0	0	0	0	0	0	0	0

SPI0_ML field descriptions

Field	Description
7–0 Bits[7:0]	Hardware compare value (low byte)

34.3.9 SPI control register 3 (SPIx_C3)

This register introduces a 64-bit FIFO function on both transmit and receive buffers. It applies only for an instance of the SPI module that supports the FIFO feature.

FIFO mode is enabled by setting the FIFOMODE bit to 1. A write to this register occurs only when it sets the FIFOMODE bit to 1.

Using this FIFO feature allows the SPI to provide high speed transfers of large amounts of data without consuming large amounts of the CPU bandwidth.

Enabling this FIFO function affects the behavior of some of the read/write buffer flags in the S register as follows:

- The SPRF of the S register is 1 when the receive FIFO is filled. As a result:
  - If the RXDMAE bit in the C2 register is 1, SPRF generates a receive DMA request.
  - If the RXDMAE bit in the C2 register is 0 and the SPIE bit in the C1 register is 1, SPRF interrupts the CPU.
- The SPTEF of the S register is 1 when the transmit FIFO is empty. As a result:
  - If the TXDMAE bit in the C2 register is 1, SPTEF generates a transmit DMA request.
  - If the TXDMAE bit in the C2 register is 0 and the SPTIE bit in the C1 register is 1, SPTEF interrupts the CPU.

Two interrupt enable bits, TNEARIEN and RNFULLIEN, provide CPU interrupts based on the "watermark" feature of the TNEARF and RNFULLF flags of the S register.

Address: FFFF_8190h base + 8h offset = FFFF_8198h

Bit	7	6	5	4	3	2	1	0
Read	0		TNEAREF_MARK	RNFULLF_MARK	INTCLR	TNEARIEN	RNFULLIEN	FIFOMODE
Write								
Reset	0	0	0	0	0	0	0	0

SPI0_C3 field descriptions

Field	Description
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 TNEAREF_MARK	Transmit FIFO nearly empty watermark This bit selects the mark after which the TNEAREF flag is asserted. 0 TNEAREF is set when the transmit FIFO has 16 bits or less 1 TNEAREF is set when the transmit FIFO has 32 bits or less
4 RNFULLF_MARK	Receive FIFO nearly full watermark This bit selects the mark after which the RNFULLF flag is asserted. 0 RNFULLF is set when the receive FIFO has 48 bits or more 1 RNFULLF is set when the receive FIFO has 32 bits or more
3 INTCLR	Interrupt clearing mechanism select This bit selects the mechanism by which the SPRF, SPTEF, TNEAREF, and RNFULLF interrupts are cleared. 0 These interrupts are cleared when the corresponding flags are cleared depending on the state of the FIFOs 1 These interrupts are cleared by writing the corresponding bits in the CI register
2 TNEARIEN	Transmit FIFO nearly empty interrupt enable Writing 1 to this bit enables the SPI to interrupt the CPU when the TNEAREF flag is set. This bit is ignored and has no function if the FIFOMODE bit is 0. 0 No interrupt upon TNEAREF being set 1 Enable interrupts upon TNEAREF being set
1 RNFULLIEN	Receive FIFO nearly full interrupt enable Writing 1 to this bit enables the SPI to interrupt the CPU when the RNFULLF flag is set. This bit is ignored and has no function if the FIFOMODE bit is 0. 0 No interrupt upon RNFULLF being set 1 Enable interrupts upon RNFULLF being set
0 FIFOMODE	FIFO mode enable This bit enables the SPI to use a 64-bit FIFO (8 bytes or four 16-bit words) for both transmit and receive buffers. 0 Buffer mode disabled 1 Data available in the receive data buffer

34.3.10 SPI clear interrupt register (SPIx_CI)

This register applies only for an instance of the SPI module that supports the FIFO feature.

The register has four bits dedicated to clearing the interrupts. Writing 1 to these bits clears the corresponding interrupts if the INTCLR bit in the C3 register is 1. Reading these bits always returns 0.

This register also has two read-only bits to indicate the transmit FIFO and receive FIFO overrun conditions. When the receive FIFO is full and data is received, RXFOF is set. Similarily, when the transmit FIFO is full and a write to the data register occurs, TXFOF is set. These flags are cleared when the CI register is read while the flags are set.

The register has two more read-only bits to indicate the error flags. These flags are set when, due to some spurious reason, entries in the FIFO total more than 64 bits of data. At this point, all the flags in the status register are reset, and entries in the FIFO are flushed with the corresponding error flags set. These flags are cleared when the CI register is read while the flags are set.

Address: FFFF_8190h base + 9h offset = FFFF_8199h

Bit	7	6	5	4	3	2	1	0
Read	TXFERR	RXFERR	TXFOF	RXFOF	0	0	0	0
Write					TNEAREFCI	RNFULLFCI	SPTEFCI	SPRFCI
Reset	0	0	0	0	0	0	0	0

SPI0_CI field descriptions

Field	Description
7 TXFERR	Transmit FIFO error flag This flag indicates that a transmit FIFO error occurred because entries in the FIFO total more than 64 bits of data. 0 No transmit FIFO error occurred 1 A transmit FIFO error occurred
6 RXFERR	Receive FIFO error flag This flag indicates that a receive FIFO error occurred because entries in the FIFO total more than 64 bits of data. 0 No receive FIFO error occurred 1 A receive FIFO error occurred
5 TXFOF	Transmit FIFO overflow flag This flag indicates that a transmit FIFO overflow condition has occurred.

Table continues on the next page...

SPI0_CI field descriptions (continued)

Field	Description
	0 Transmit FIFO overflow condition has not occurred 1 Transmit FIFO overflow condition occurred
4 RXFOF	Receive FIFO overflow flag This flag indicates that a receive FIFO overflow condition has occurred. 0 Receive FIFO overflow condition has not occurred 1 Receive FIFO overflow condition occurred
3 TNEAREFCI	Transmit FIFO nearly empty flag clear interrupt Writing 1 to this bit clears the TNEAREF interrupt provided that C3[3] is set.
2 RNFULLFCI	Receive FIFO nearly full flag clear interrupt Writing 1 to this bit clears the RNFULLF interrupt provided that C3[3] is set.
1 SPTEFCI	Transmit FIFO empty flag clear interrupt Writing 1 to this bit clears the SPTEF interrupt provided that C3[3] is set.
0 SPRFCI	Receive FIFO full flag clear interrupt Writing 1 to this bit clears the SPRF interrupt provided that C3[3] is set.

34.4 Functional Description

This section provides the functional description of the module.

34.4.1 General

The SPI system is enabled by setting the SPI enable (SPE) bit in SPI Control Register 1. While the SPE bit is set, the four associated SPI port pins are dedicated to the SPI function as:

- Slave select (SS)
- Serial clock (SPSCK)
- Master out/slave in (MOSI)
- Master in/slave out (MISO)

An SPI transfer is initiated in the master SPI device by reading the SPI status register (SPIx_S) when SPTEF = 1 and then writing data to the transmit data buffer (write to SPIx_DH:SPIx_DL). When a transfer is complete, received data is moved into the receive data buffer. The SPIx_DH:SPIx_DL registers act as the SPI receive data buffer for reads and as the SPI transmit data buffer for writes.

The clock phase control bit (CPHA) and clock polarity control bit (CPOL) in the SPI Control Register 1 (SPIx_C1) select one of four possible clock formats to be used by the SPI system. The CPOL bit simply selects a non-inverted or inverted clock. The CPHA bit is used to accommodate two fundamentally different protocols by sampling data on odd numbered SPSCK edges or on even numbered SPSCK edges.

The SPI can be configured to operate as a master or as a slave. When the MSTR bit in SPI Control Register 1 is set, master mode is selected; when the MSTR bit is clear, slave mode is selected.

34.4.2 Master Mode

The SPI operates in master mode when the MSTR bit is set. Only a master SPI module can initiate transmissions. A transmission begins by reading the SPIx_S register while SPTEF = 1 and writing to the master SPI data registers. If the shift register is empty, the byte immediately transfers to the shift register. The data begins shifting out on the MOSI pin under the control of the serial clock.

- SPSCK
  - The SPR3, SPR2, SPR1, and SPR0 baud rate selection bits in conjunction with the SPPR2, SPPR1, and SPPR0 baud rate preselection bits in the SPI Baud Rate register control the baud rate generator and determine the speed of the transmission. The SPSCK pin is the SPI clock output. Through the SPSCK pin, the baud rate generator of the master controls the shift register of the slave peripheral.
- MOSI, MISO pin
  - In master mode, the function of the serial data output pin (MOSI) and the serial data input pin (MISO) is determined by the SPC0 and BIDIROE control bits.
- $\overline{SS}$ pin
  - If MODFEN and SSOE bit are set, the SS pin is configured as slave select output. The SS output becomes low during each transmission and is high when the SPI is in idle state. If MODFEN is set and SSOE is cleared, the $\overline{SS}$ pin is configured as input for detecting mode fault error. If the SS input becomes low this indicates a mode fault error where another master tries to drive the MOSI and SPSCK lines. In this case, the SPI immediately switches to slave mode by clearing the MSTR bit and also disables the slave output buffer MISO (or SISO in bidirectional mode). As a result, all outputs are disabled, and SPSCK, MOSI and MISO are inputs. If a transmission is in progress when the mode fault occurs, the transmission is aborted and the SPI is forced into idle state. This

mode fault error also sets the mode fault (MODF) flag in the SPI Status Register (SPIx_S). If the SPI interrupt enable bit (SPIE) is set when the MODF flag gets set, then an SPI interrupt sequence is also requested. When a write to the SPI Data Register in the master occurs, there is a half SPSCK-cycle delay. After the delay, SPSCK is started within the master. The rest of the transfer operation differs slightly, depending on the clock format specified by the SPI clock phase bit, CPHA, in SPI Control Register 1 (see SPI Clock Formats).

Note

A change of the bits CPOL, CPHA, SSOE, LSBFE, MODFEN, SPC0, BIDIROE with SPC0 set, SPIMODE, FIFOMODE, SPPR2-SPPR0 and SPR3-SPR0 in master mode abort a transmission in progress and force the SPI into idle state. The remote slave cannot detect this, therefore the master has to ensure that the remote slave is set back to idle state.

34.4.3 Slave Mode

The SPI operates in slave mode when the MSTR bit in SPI Control Register1 is clear.

- SPSCK

  In slave mode, SPSCK is the SPI clock input from the master.

- MISO, MOSI pin

  In slave mode, the function of the serial data output pin (MISO) and serial data input pin (MOSI) is determined by the SPC0 bit and BIDIROE bit in SPI Control Register 2.

- SS pin

  The SS pin is the slave select input. Before a data transmission occurs, the SS pin of the slave SPI must be low. SS must remain low until the transmission is complete. If SS goes high, the SPI is forced into an idle state.

  The SS input also controls the serial data output pin. If SS is high (not selected), the serial data output pin is high impedance. If SS is low, the first bit in the SPI Data Register is driven out of the serial data output pin. Also, if the slave is not selected (SS is high), then the SPSCK input is ignored and no internal shifting of the SPI shift register occurs.

Although the SPI is capable of duplex operation, some SPI peripherals are capable of only receiving SPI data in a slave mode. For these simpler devices, there is no serial data out pin.

Note

When peripherals with duplex capability are used, take care not to simultaneously enable two receivers whose serial outputs drive the same system slave's serial data output line.

As long as no more than one slave device drives the system slave's serial data output line, it is possible for several slaves to receive the same transmission from a master, although the master would not receive return information from all of the receiving slaves.

If the CPHA bit in SPI Control Register 1 is clear, odd numbered edges on the SPSCK input cause the data at the serial data input pin to be latched. Even numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on the LSBFE bit.

If the CPHA bit is set, even numbered edges on the SPSCK input cause the data at the serial data input pin to be latched. Odd numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on the LSBFE bit.

When CPHA is set, the first edge is used to get the first data bit onto the serial data output pin. When CPHA is clear and the SS input is low (slave selected), the first bit of the SPI data is driven out of the serial data output pin. After the eighth (SPIMODE = 0) or sixteenth (SPIMODE = 1) shift, the transfer is considered complete and the received data is transferred into the SPI Data register. To indicate transfer is complete, the SPRF flag in the SPI Status Register is set.

Note

A change of the bits FIFOMODE,SPIMODE, BIDIROE with SPC0 set, CPOL, CPHA, SSOE, LSBFE, MODFEN, and SPC0 in slave mode will corrupt a transmission in progress and must be avoided.

34.4.4 SPI FIFO Mode

When the FIFO feature is supported: The SPI works in FIFO mode when the C3[FIFOMODE] bit is set. When the module is in FIFO mode, the SPI RX buffer and SPI TX buffer are replaced by an 8-byte-deep FIFO, as the following figures show.

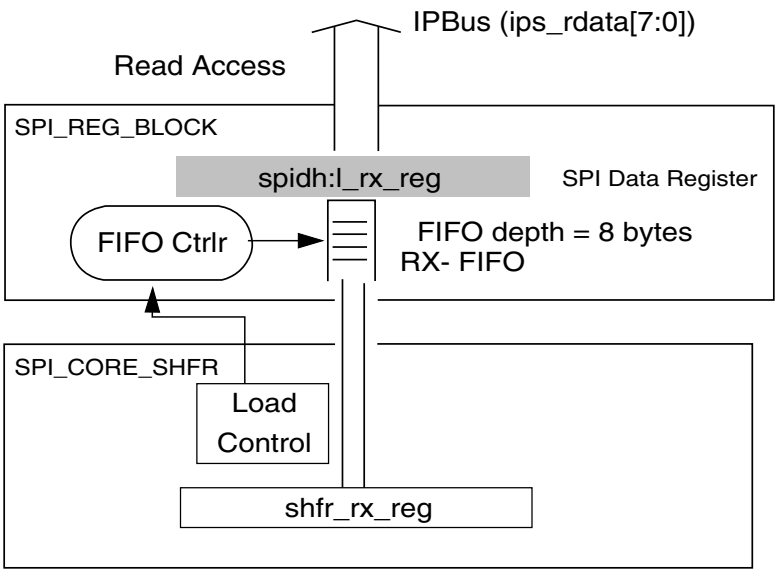


Figure 34-44. SPIH:L read side structural overview in FIFO mode

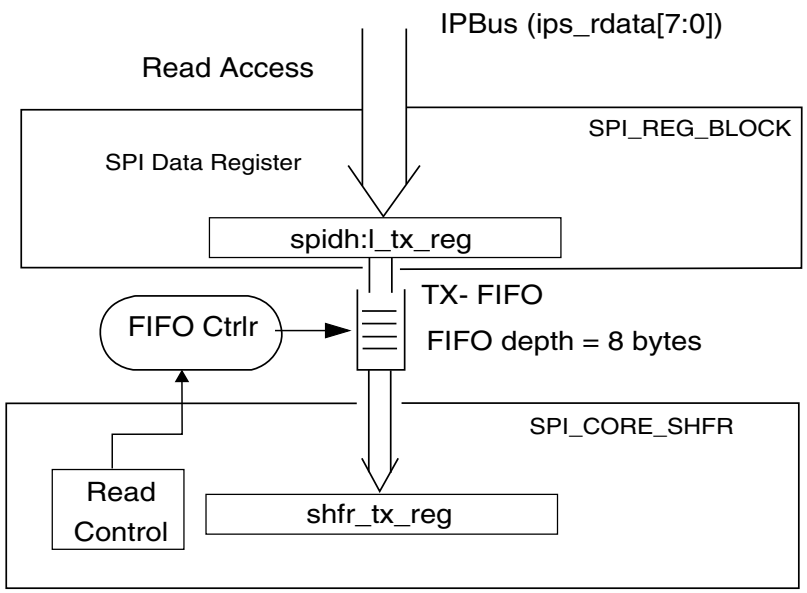


Figure 34-45. SPIH:L write side structural overview in FIFO mode

34.4.5 SPI Transmission by DMA

SPI supports both Transmit and Receive by DMA. The basic flow of SPI transmission by DMA is as below.

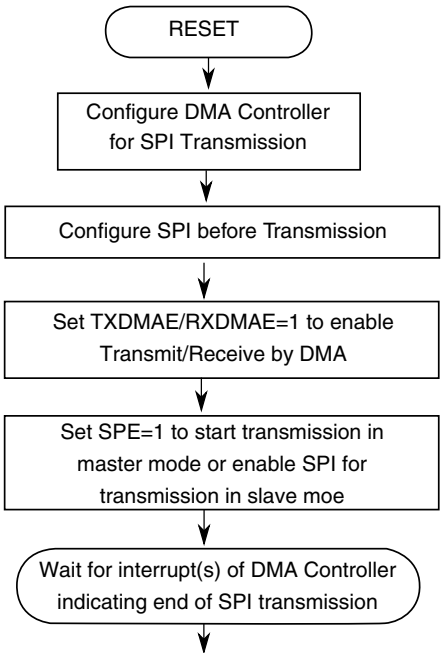


Figure 34-46. Basic Flow of SPI Transmission by DMA

34.4.5.1 Transmit by DMA

Transmit by DMA is supported only when TXDMAE is set. A transmit DMA request is asserted when both SPE and SPTEF are set. Then the on-chip DMA controller detects this request and transfers data from memory into the SPI data register. After that, TX DMA DONE is asserted to clear SPTEF automatically. This process repeats until all data for transmission (the number is decided by the configuration register[s] of the DMA controller) is sent.

When the FIFO feature is supported: In FIFO mode (FIFOMODE=1) and when a data length of 8 bits is selected (SPIMODE=0), the DMA transfer for one transmit DMA request can write more than 1 byte (up to 8 bytes) to the DL register because the TX FIFO can store 8 bytes of transmit data. In FIFO mode (FIFOMODE=1) and when a data length of 16 bits is selected (SPIMODE=1), the DMA transfer for one transmit DMA request can write more than 1 word (up to 4 words) to the DH:DL registers because the TX FIFO can store 4 words of transmit data. A larger number of bytes or words transferred from memory to the SPI data register for each transmit DMA request results in a lower total number of transmit DMA requests.

When the FIFO feature is supported and FIFOMODE is 0: After DMA transfers the first byte to the SPI data register, the SPI pushes this data into the shifter, thereby making SPTEF high again. This generates another DMA request immediately, but the CPU lacks enough time to service the first DMA interrupt service request (ISR). The subsequent

DMA request is paced at the SPI transfer rate. Manage this behavior during the first byte transfer through the DMA channel. Write the first byte to the SPI data register via the CPU. The other bytes are transmitted by the DMA.

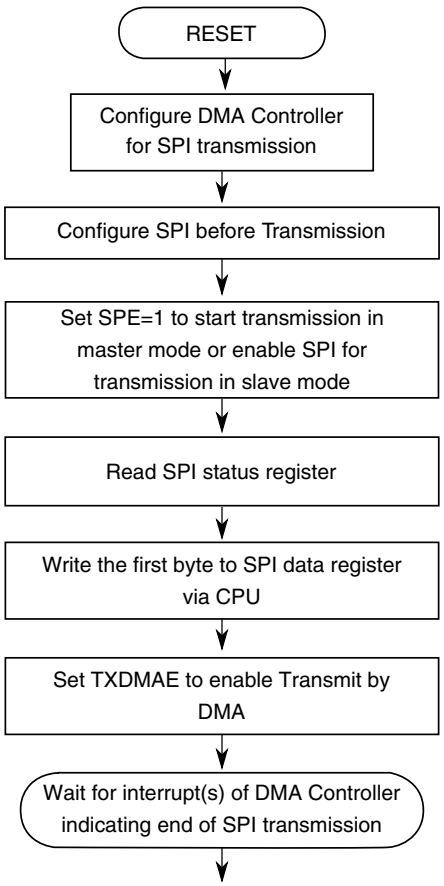


Figure 34-47. Recommended startup of SPI transmit by DMA

34.4.5.2 Receive by DMA

Receive by DMA is supported only when RXDMAE is set. A receive DMA request is asserted when both SPE and SPRF are set. Then the on-chip DMA controller detects this request and transfers data from the SPI data register into memory. After that, RX DMA DONE is asserted to clear SPRF automatically. This process repeats until all data to be received (the number is decided by configuration register[s] of the DMA controller) is received or no receive DMA request is generated again because the SPI transmission is finished.

When the FIFO feature is supported: In FIFO mode (FIFOMODE=1) and when a data length of 8 bits is selected (SPIMODE=0), the DMA transfer for one receive DMA request can read more than 1 byte (up to 8 bytes) from the SPI data register because the RX FIFO is full with 8 bytes. In FIFO mode (FIFOMODE=1) and when a data length of

16 bits is selected (SPIMODE=1), the DMA transfer for one receive DMA request can read more than 1 word (up to 4 words) from the DH:DL registers because the RX FIFO is full with 4 words. A larger number of bytes or words transferred from the SPI data register to memory for one receive DMA request results in a lower total number of receive DMA requests.

34.4.6 Data Transmission Length

The SPI can support data lengths of 8 or 16 bits. The length can be configured with the SPIMODE bit in the SPIx_C2 register.

In 8-bit mode (SPIMODE = 0), the SPI Data Register is comprised of one byte: SPIx_DL. The SPI Match Register is also comprised of only one byte: SPIx_ML. Reads of SPIx_DH and SPIx_MH will return zero. Writes to SPIx_DH and SPIx_MH will be ignored.

In 16-bit mode (SPIMODE = 1), the SPI Data Register is comprised of two bytes: SPIx_DH and SPIx_DL. Reading either byte (SPIx_DH or SPIx_DL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. Writing to either byte (SPIx_DH or SPIx_DL) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent 16-bit value into the transmit data buffer.

In 16-bit mode, the SPI Match Register is also comprised of two bytes: SPIx_MH and SPIx_ML. There is no buffer mechanism for the reading of SPIxMH and SPIxML since they can only be changed by writing at CPU side. Writing to either byte (SPIx_MH or SPIx_ML) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent 16-bit value into the SPI Match Register.

Any switching between 8- and 16-bit data transmission length (controlled by SPIMODE bit) in master mode will abort a transmission in progress, force the SPI system into idle state, and reset all status bits in the SPIx_S register. To initiate a transfer after writing to SPIMODE, the SPIx_S register must be read with SPTEF = 1, and data must be written to SPIx_DH:SPIx_DL in 16-bit mode (SPIMODE = 1) or SPIx_DL in 8-bit mode (SPIMODE = 0).

In slave mode, user software should write to SPIMODE only once to prevent corrupting a transmission in progress.

Note

Data can be lost if the data length is not the same for both master and slave devices.

34.4.7 SPI Clock Formats

To accommodate a wide variety of synchronous serial peripherals from different manufacturers, the SPI system has a clock polarity (CPOL) bit and a clock phase (CPHA) control bit to select one of four clock formats for data transfers. CPOL selectively inserts an inverter in series with the clock. CPHA chooses between two different clock phase relationships between the clock and data.

The following figure shows the clock formats when SPIMODE = 0 (8-bit mode) and CPHA = 1. At the top of the figure, the eight bit times are shown for reference with bit 1 starting at the first SPSCK edge and bit 8 ending one-half SPSCK cycle after the eighth SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The $\overline{SS}$ OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master $\overline{SS}$ output goes to active low one-half SPSCK cycle before the start of the transfer and goes back high at the end of the eighth bit time of the transfer. The $\overline{SS}$ IN waveform applies to the slave select input of a slave.

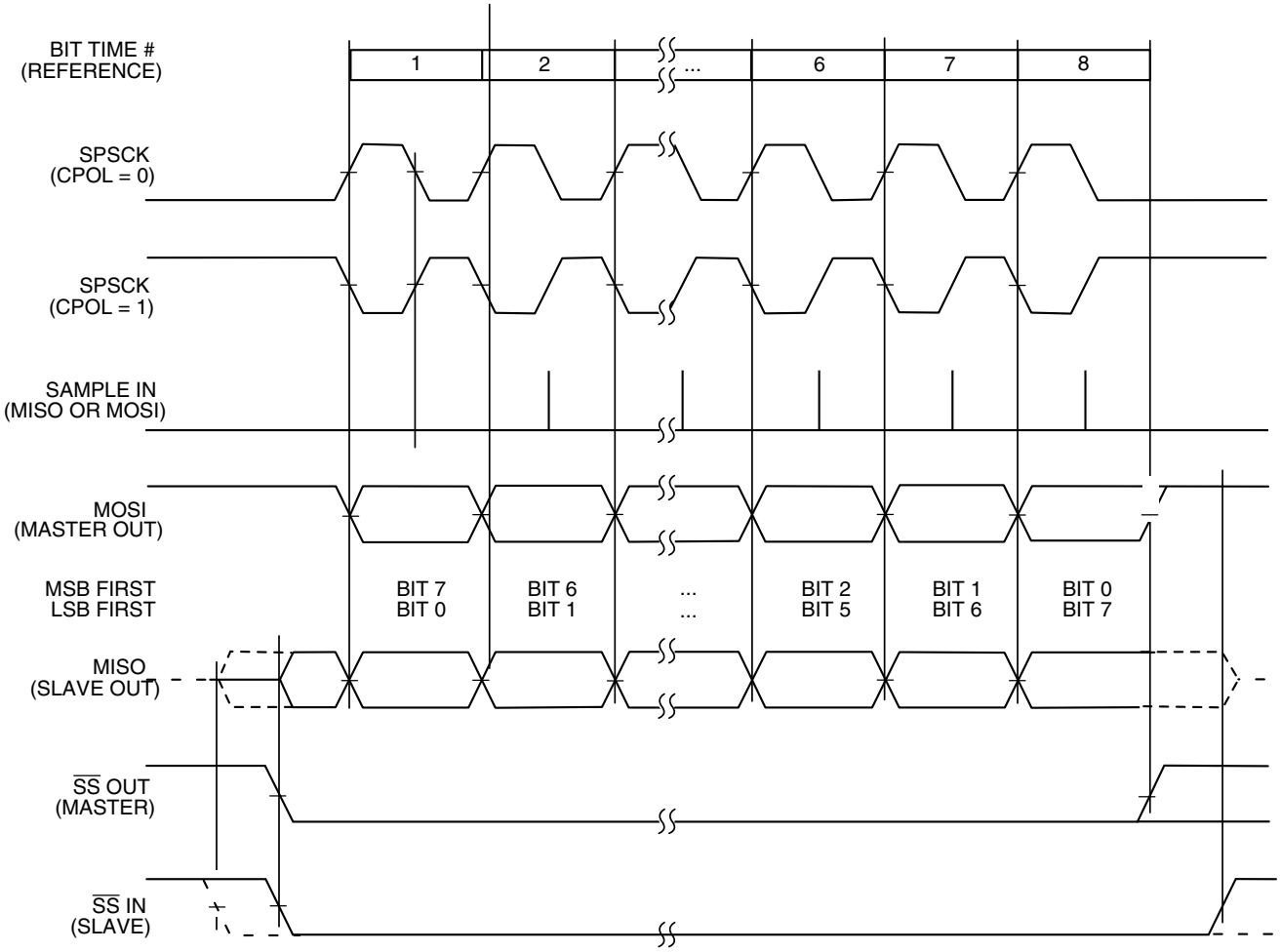


Figure 34-48. SPI Clock Formats (CPHA = 1)

When CPHA = 1, the slave begins to drive its MISO output when $\overline{SS}$ goes to active low, but the data is not defined until the first SPSCK edge. The first SPSCK edge shifts the first bit of data from the shifter onto the MOSI output of the master and the MISO output of the slave. The next SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the third SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled, and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively.

When CPHA = 1, the slave's $\overline{SS}$ input is not required to go to its inactive high level between transfers. In this clock format, a back-to-back transmission can occur, as follows:

1. A transmission is in progress.
2. A new data byte is written to the transmit buffer before the in-progress transmission is complete.
3. When the in-progress transmission is complete, the new, ready data byte is transmitted immediately.

Between these two successive transmissions, no pause is inserted; the $\overline{SS}$ pin remains low.

The following figure shows the clock formats when SPIMODE = 0 and CPHA = 0. At the top of the figure, the eight bit times are shown for reference with bit 1 starting as the slave is selected (SS IN goes low), and bit 8 ends at the last SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The SS OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master SS output goes to active low at the start of the first bit time of the transfer and goes back high one-half SPSCK cycle after the end of the eighth bit time of the transfer. The SS IN waveform applies to the slave select input of a slave.

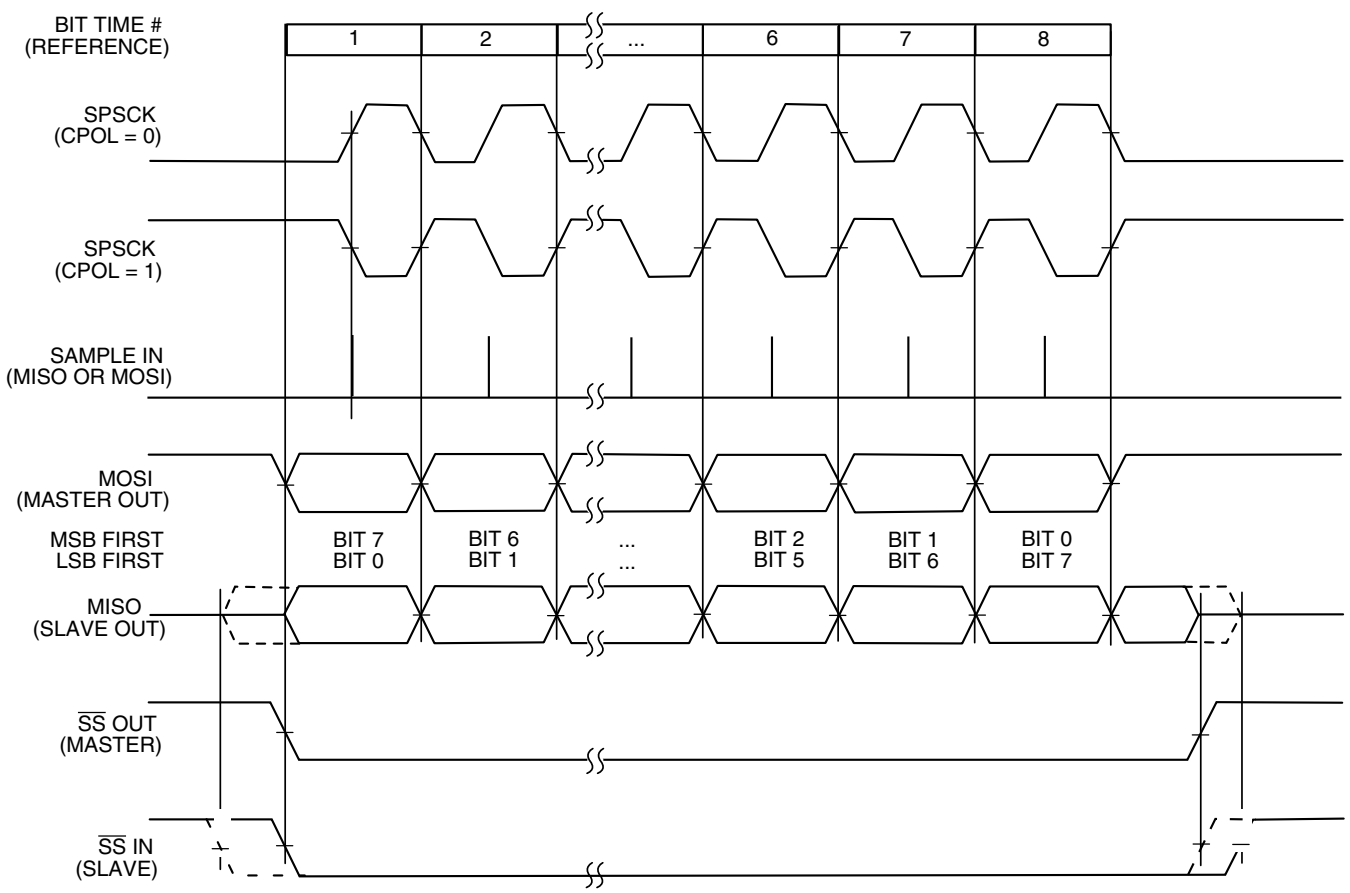


Figure 34-49. SPI Clock Formats (CPHA = 0)

When CPHA = 0, the slave begins to drive its MISO output with the first data bit value (MSB or LSB depending on LSBFE) when SS goes to active low. The first SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the second SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CPHA = 0, the slave's SS input must go to its inactive high level between transfers.

34.4.8 SPI Baud Rate Generation

As shown in the following figure, the clock source for the SPI baud rate generator is the bus clock. The three prescale bits (SPPR2:SPPR1:SPPR0) choose a prescale divisor of 1, 2, 3, 4, 5, 6, 7, or 8. The three rate select bits (SPR3:SPR2:SPR1:SPR0) divide the output of the prescaler stage by 2, 4, 8, 16, 32, 64, 128, 256, or 512 to get the internal SPI master mode bit-rate clock.

The baud rate generator is activated only when the SPI is in the master mode and a serial transfer is taking place. In the other cases, the divider is disabled to decrease I_{DD} current.

The baud rate divisor equation is as follows (except those reserved combinations in the SPI Baud Rate Divisor table).

$$BaudRateDivisor = (SPPR + 1) \times 2^{(SPR + 1)}$$

The baud rate can be calculated with the following equation:

$$BaudRate = BusClock / BaudRateDivisor$$

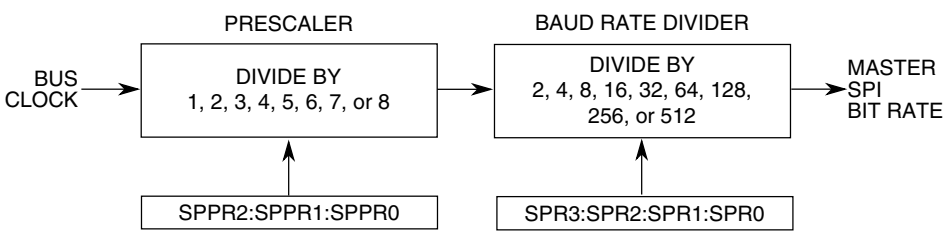


Figure 34-50. SPI Baud Rate Generation

34.4.9 Special Features

The following section shows the module special features.

34.4.9.1 $\overline{SS}$ Output

The $\overline{SS}$ output feature automatically drives the $\overline{SS}$ pin low during transmission to select external devices and drives the $\overline{SS}$ pin high during idle to deselect external devices. When the $\overline{SS}$ output is selected, the $\overline{SS}$ output pin is connected to the $\overline{SS}$ input pin of the external device.

The $\overline{SS}$ output is available only in master mode during normal SPI operation by asserting the SSOE and MODFEN bits as shown in the description of the C1[SSOE] bit.

The mode fault feature is disabled while $\overline{SS}$ output is enabled.

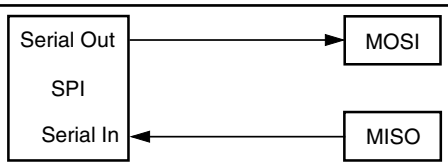
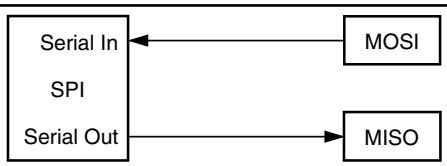
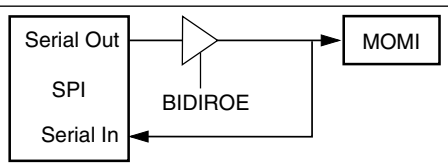
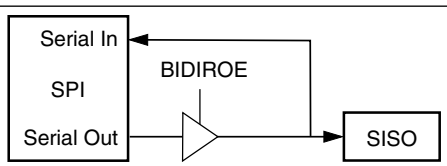
Note

Be careful when using the $\overline{SS}$ output feature in a multimaster system because the mode fault feature is not available for detecting system errors between masters.

34.4.9.2 Bidirectional Mode (MOMI or SISO)

The bidirectional mode is selected when the SPC0 bit is set in SPI Control Register 2 (see the following table). In this mode, the SPI uses only one serial data pin for the interface with one or more external devices. The MSTR bit decides which pin to use. The MOSI pin becomes the serial data I/O (MOMI) pin for the master mode, and the MISO pin becomes serial data I/O (SISO) pin for the slave mode. The MISO pin in master mode and MOSI pin in slave mode are not used by the SPI.

Table 34-45. Normal Mode and Bidirectional Mode

When SPE = 1	Master Mode MSTR = 1	Slave Mode MSTR = 0
Normal Mode SPC0 = 0		
Bidirectional Mode SPC0 = 1		

The direction of each serial I/O pin depends on the BIDIROE bit. If the pin is configured as an output, serial data from the shift register is driven out on the pin. The same pin is also the serial input to the shift register.

The SPSCK is an output for the master mode and an input for the slave mode.

$\overline{SS}$ is the input or output for the master mode, and it is always the input for the slave mode.

The bidirectional mode does not affect SPSCK and $\overline{SS}$ functions.

Note

In bidirectional master mode, with the mode fault feature enabled, both data pins MISO and MOSI can be occupied by the SPI, though MOSI is normally used for transmissions in bidirectional mode and MISO is not used by the SPI. If a mode fault occurs, the SPI is automatically switched to slave mode. In this case, MISO becomes occupied by the SPI and MOSI is not used. Consider this scenario if the MISO pin is used for another purpose.

34.4.10 Error Conditions

The SPI module has one error condition: the mode fault error.

34.4.10.1 Mode Fault Error

If the $\overline{SS}$ input becomes low while the SPI is configured as a master, it indicates a system error where more than one master may be trying to drive the MOSI and SPSCK lines simultaneously. This condition is not permitted in normal operation, and it sets the MODF bit in the SPI status register automatically provided that the MODFEN bit is set.

In the special case where the SPI is in master mode and the MODFEN bit is cleared, the $\overline{SS}$ pin is not used by the SPI. In this special case, the mode fault error function is inhibited and MODF remains cleared. If the SPI system is configured as a slave, the $\overline{SS}$ pin is a dedicated input pin. A mode fault error does not occur in slave mode.

If a mode fault error occurs, the SPI is switched to slave mode, with the exception that the slave output buffer is disabled. So the SPSCK, MISO and MOSI pins are forced to be high impedance inputs to avoid any possibility of conflict with another output driver. A transmission in progress is aborted and the SPI is forced into idle state.

If the mode fault error occurs in the bidirectional mode for an SPI system configured in master mode, the output enable of MOMI (MOSI in bidirectional mode) is cleared if it was set. No mode fault error occurs in the bidirectional mode for the SPI system configured in slave mode.

The mode fault flag is cleared automatically by a read of the SPI Status Register (with MODF set) followed by a write to SPI Control Register 1. If the mode fault flag is cleared, the SPI becomes a normal master or slave again.

34.4.11 Low Power Mode Options

This section describes the low power mode options.

34.4.11.1 SPI in Run Mode

In run mode, with the SPI system enable (SPE) bit in the SPI control register clear, the SPI system is in a low-power, disabled state. SPI registers can still be accessed, but clocks to the core of this module are disabled.

34.4.11.2 SPI in Wait Mode

SPI operation in wait mode depends upon the state of the SPISWAI bit in SPI Control Register 2.

- If SPISWAI is clear, the SPI operates normally when the CPU is in wait mode.
- If SPISWAI is set, SPI clock generation ceases and the SPI module enters a power conservation state when the CPU is in wait mode.
  - If SPISWAI is set and the SPI is configured for master, any transmission and reception in progress stops at wait mode entry. The transmission and reception resumes when the SPI exits wait mode.
  - If SPISWAI is set and the SPI is configured as a slave, any transmission and reception in progress continues if the SPSCK continues to be driven from the master. This keeps the slave synchronized to the master and the SPSCK.

    If the master transmits data while the slave is in wait mode, the slave continues to send data consistent with the operation mode at the start of wait mode (that is, if the slave is currently sending its SPIx_DH:SPIx_DL to the master, it continues to send the same byte. Otherwise, if the slave is currently sending the last data received byte from the master, it continues to send each previously received data from the master byte).

Note

Care must be taken when expecting data from a master while the slave is in a wait mode or a stop mode where the peripheral bus clock is stopped but internal logic states are retained. Even though the shift register continues to operate, the rest of the SPI is shut down (that is, an SPRF interrupt is not generated until an exit from stop or wait mode). Also, the data from the shift register is not copied into the SPIx_DH:SPIx_DL registers until after the slave SPI has exited wait or stop mode. An SPRF flag and SPIx_DH:SPIx_DL copy is only generated if wait mode is entered or exited during a transmission. If the slave enters wait mode in idle mode and exits wait mode in idle mode, neither an SPRF nor a SPIx_DH:SPIx_DL copy occurs.

34.4.11.3 SPI in Stop Mode

Operation in a stop mode where the peripheral bus clock is stopped but internal logic states are retained depends on the SPI system. The stop mode does not depend on the SPISWAI bit. Upon entry to this type of stop mode, the SPI module clock is disabled (held high or low).

- If the SPI is in master mode and exchanging data when the CPU enters the stop mode, the transmission is frozen until the CPU exits stop mode. After the exit from stop mode, data to and from the external SPI is exchanged correctly.
- In slave mode, the SPI remains synchronized with the master.

The SPI is completely disabled in a stop mode where the peripheral bus clock is stopped and internal logic states are not retained. After an exit from this type of stop mode, all registers are reset to their default values, and the SPI module must be re-initialized.

34.4.12 Reset

The reset values of registers and signals are described in the Memory Map and Register Descriptions content, which details the registers and their bitfields.

- If a data transmission occurs in slave mode after a reset without a write to SPIx_DH:SPIx_DL, the transmission consists of "garbage" or the data last received from the master before the reset.
- Reading from SPIx_DH:SPIx_DL after reset always returns zeros.

34.4.13 Interrupts

The SPI originates interrupt requests only when the SPI is enabled (the SPE bit in the SPIx_C1 register is set). The following is a description of how the SPI makes a request and how the MCU should acknowledge that request. The interrupt vector offset and interrupt priority are chip dependent.

Four flag bits, three interrupt mask bits, and one interrupt vector are associated with the SPI system. The SPI interrupt enable mask (SPIE) enables interrupts from the SPI receiver full flag (SPRF) and mode fault flag (MODF). The SPI transmit interrupt enable mask (SPTIE) enables interrupts from the SPI transmit buffer empty flag (SPTEF). The SPI match interrupt enable mask bit (SPIMIE) enables interrupts from the SPI match flag (SPMF). When one of the flag bits is set, and the associated interrupt mask bit is set, a hardware interrupt request is sent to the CPU. If the interrupt mask bits are cleared, software can poll the associated flag bits instead of using interrupts. The SPI interrupt service routine (ISR) should check the flag bits to determine which event caused the interrupt. The service routine should also clear the flag bit(s) before returning from the ISR (usually near the beginning of the ISR).

34.4.13.1 MODF

MODF occurs when the master detects an error on the $\overline{SS}$ pin. The master SPI must be configured for the MODF feature (see the description of the C1[SSOE] bit). Once MODF is set, the current transfer is aborted and the master (MSTR) bit in the SPIx_C1 register resets to 0.

The MODF interrupt is reflected in the status register's MODF flag. Clearing the flag also clears the interrupt. This interrupt stays active while the MODF flag is set. MODF has an automatic clearing process that is described in the SPI Status Register.

34.4.13.2 SPRF

SPRF occurs when new data has been received and copied to the SPI receive data buffer. In 8-bit mode, SPRF is set only after all 8 bits have been shifted out of the shift register and into SPIx_DL. In 16-bit mode, SPRF is set only after all 16 bits have been shifted out of the shift register and into SPIx_DH:SPIx_DL.

After SPRF is set, it does not clear until it is serviced. SPRF has an automatic clearing process that is described in the SPI Status Register details. If the SPRF is not serviced before the end of the next transfer (that is, SPRF remains active throughout another transfer), the subsequent transfers are ignored and no new data is copied into the Data register.

34.4.13.3 SPTEF

SPTEF occurs when the SPI transmit buffer is ready to accept new data. In 8-bit mode, SPTEF is set only after all 8 bits have been moved from SPIx_DL into the shifter. In 16-bit mode, SPTEF is set only after all 16 bits have been moved from SPIx_DH:SPIx_DL into the shifter.

After SPTEF is set, it does not clear until it is serviced. SPTEF has an automatic clearing process that is described in the SPI Status Register details.

34.4.13.4 SPMF

SPMF occurs when the data in the receive data buffer is equal to the data in the SPI match register. In 8-bit mode, SPMF is set only after bits 7–0 in the receive data buffer are determined to be equivalent to the value in SPIx_ML. In 16-bit mode, SPMF is set after bits 15–0 in the receive data buffer are determined to be equivalent to the value in SPIx_MH:SPIx_ML.

34.4.13.5 TNEAREF

The TNEAREF bit applies when the FIFO feature is supported.

The TNEAREF flag is set when only one 16-bit word or two 8-bit bytes of data remain in the transmit FIFO provided C3[5] = 0 or when only two 16-bit words or four 8-bit bytes of data remain in the transmit FIFO provided C3[5] =1. If FIFOMODE is not enabled, ignore this bit.

Clearing this interrupt depends on the state of C3[3] and the status of TNEAREF. Refer to the description of the SPI status (S) register.

34.4.13.6 RNFULLF

The RNFULLF bit applies when the FIFO feature is supported.

RNFULLF is set when more than three 16-bit words or six 8-bit bytes of data remain in the receive FIFO provided C3[4] = 0 or when more than two 16-bit words or four 8-bit bytes of data remain in the receive FIFO provided C3[4] = 1.

Clearing this interrupt depends on the state of C3[3] and the status of RNFULLF. Refer to the description of the SPI status (S) register.

34.5 Initialization/Application Information

This section discusses an example of how to initialize and use the SPI.

34.5.1 Initialization Sequence

Before the SPI module can be used for communication, an initialization procedure must be carried out, as follows:

1. Update control register 1 (SPIx_C1) to enable the SPI and to control interrupt enables. This register also sets the SPI as master or slave, determines clock phase and polarity, and configures the main SPI options.
2. Update control register 2 (SPIx_C2) to enable additional SPI functions such as the SPI match interrupt feature, the master mode-fault function, and bidirectional mode output as well as to control 8- or 16-bit mode selection and other optional features.
3. Update the baud rate register (SPIx_BR) to set the prescaler and bit rate divisor for an SPI master.
4. Update the hardware match register (SPIx_MH:SPIx_ML) with the value to be compared to the receive data register for triggering an interrupt if hardware match interrupts are enabled.
5. In the master, read SPIx_S while SPTEF = 1, and then write to the transmit data register (SPIx_DH:SPIx_DL) to begin transfer.

34.5.2 Pseudo-Code Example

In this example, the SPI module is set up for master mode with only hardware match interrupts enabled. The SPI runs in 16-bit mode at a maximum baud rate of bus clock divided by 2. Clock phase and polarity are set for an active-high SPI clock where the first edge on SPSCK occurs at the start of the first cycle of a data transfer.

SPIx_C1=0x54(%01010100)					
	Bit 7	SPIE	=	0	Disables receive and mode fault interrupts
	Bit 6	SPE	=	1	Enables the SPI system
	Bit 5	SPTIE	=	0	Disables SPI transmit interrupts
	Bit 4	MSTR	=	1	Sets the SPI module as a master SPI device
	Bit 3	CPOL	=	0	Configures SPI clock as active-high
	Bit 2	CPHA	=	1	First edge on SPSCK at start of first data transfer cycle
	Bit 1	SSOE	=	0	Determines SS pin function when mode fault enabled
	Bit 0	LSBFE	=	0	SPI serial data transfers start with most significant bit

SPIx_C2 = 0xC0(%11000000)					
	Bit 7	SPMIE	=	1	SPI hardware match interrupt enabled
	Bit 6	SPIMODE	=	1	Configures SPI for 16-bit mode
	Bit 5	TXDMAE	=	0	DMA request disabled
	Bit 4	MODFEN	=	0	Disables mode fault function
	Bit 3	BIDIROE	=	0	SPI data I/O pin acts as input
	Bit 2	RXDMAE	=	0	DMA request disabled
	Bit 1	SPISWAI	=	0	SPI clocks operate in wait mode
	Bit 0	SPC0	=	0	uses separate pins for data input and output

SPIx_BR = 0x00(%00000000)					
	Bit 7		=	0	Reserved
	Bit 6:4		=	000	Sets prescale divisor to 1
	Bit 3:0		=	0000	Sets baud rate divisor to 2

SPIx_S = 0x00(%00000000)					
	Bit 7	SPRF	=	0	Flag is set when receive data buffer is full
	Bit 6	SPMF	=	0	Flag is set when SPIx_MH/ML = receive data buffer
	Bit 5	SPTEF	=	0	Flag is set when transmit data buffer is empty
	Bit 4	MODF	=	0	Mode fault flag for master mode
	Bit 3:0	RNFULLF, TNEARF, TXFULLF, and RFIFOEF	=	0	Reserved (when FIFOMODE is not present or is 0) or FIFO flags (when FIFOMODE is 1)
	Bit 3:0		=	0	FIFOMODE is not enabled

SPIx_MH = 0xXX	
	In 16-bit mode, this register holds bits 8–15 of the hardware match buffer. In 8-bit mode, writes to this register will be ignored.

SPIx_ML = 0xXX	
	Holds bits 0–7 of the hardware match buffer.

SPIx_DH = 0xxx	
	In 16-bit mode, this register holds bits 8–15 of the data to be transmitted by the transmit buffer and received by the receive buffer.

SPIx_DL = 0xxx	
	Holds bits 0–7 of the data to be transmitted by the transmit buffer and received by the receive buffer.

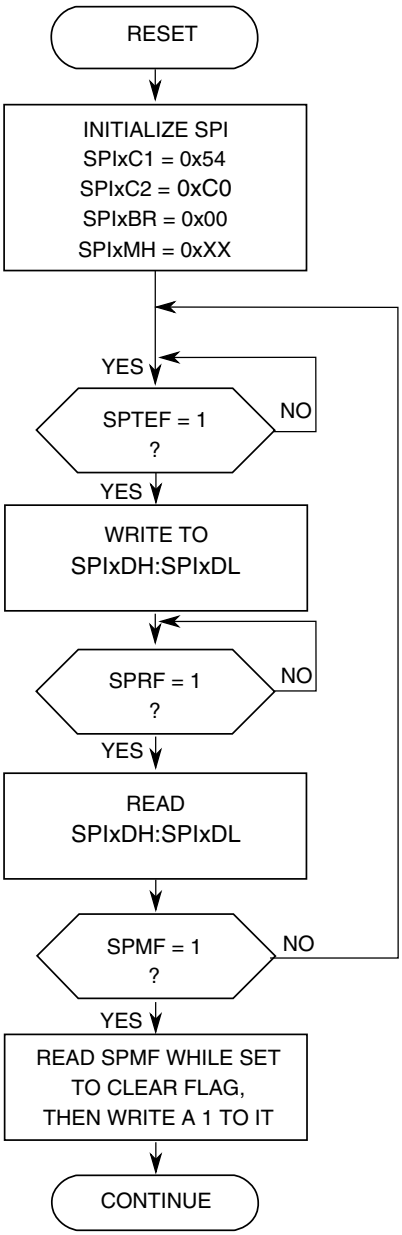


Figure 34-51. Initialization Flowchart Example for SPI Master Device in 16-bit Mode for FIFOMODE = 0

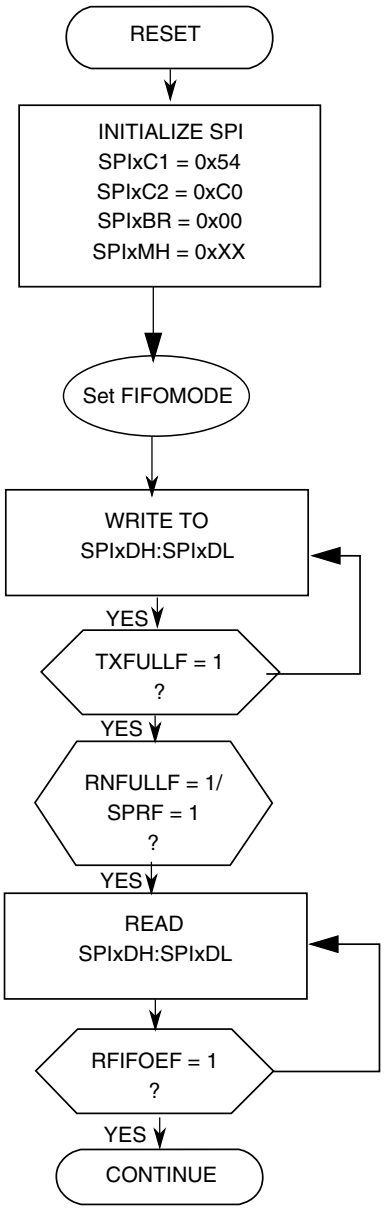


Figure 34-52. Initialization Flowchart Example for SPI Master Device in 16-bit Mode for FIFOMODE = 1

Chapter 35
Inter-Integrated Circuit (I2C)

35.1 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 100 kbit/s with maximum bus loading and timing. The I2C device is capable of operating at higher baud rates, up to a maximum of clock/20, with reduced bus loading. The maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF. The I2C module also complies with the *System Management Bus (SMBus) Specification, version 2*.

35.1.1 Features

The I2C module has the following features:

- Compatible with *The I^2C-Bus Specification*
- Multimaster operation
- Software programmable for one of 64 different serial clock frequencies
- Software-selectable acknowledge bit
- Interrupt-driven byte-by-byte data transfer
- Arbitration-lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- START and STOP signal generation and detection
- Repeated START signal generation and detection
- Acknowledge bit generation and detection
- Bus busy detection
- General call recognition
- 10-bit address extension
- Support for *System Management Bus (SMBus) Specification, version 2*
- Programmable glitch input filter
- Low power mode wakeup on slave address match

- Range slave address support
- DMA support

35.1.2 Modes of operation

The I2C module's operation in various low power modes is as follows:

- Run mode: This is the basic mode of operation. To conserve power in this mode, disable the module.
- Wait mode: The module continues to operate when the core is in Wait mode and can provide a wakeup interrupt.
- Stop mode: The module is inactive in Stop mode for reduced power consumption, except that address matching is enabled in Stop mode. The STOP instruction does not affect the I2C module's register states.

35.1.3 Block diagram

The following figure is a functional block diagram of the I2C module.

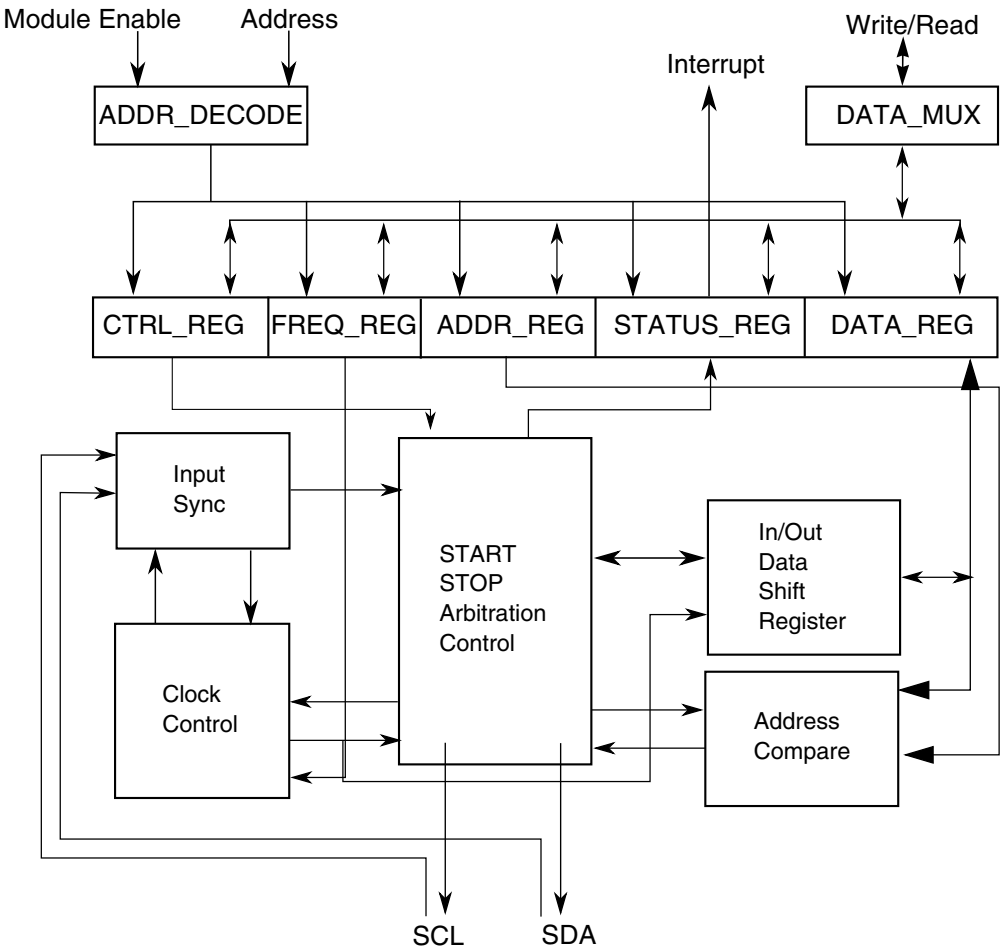


Figure 35-1. I2C Functional block diagram

35.2 I²C signal descriptions

The signal properties of I²C are shown in the following table.

Table 35-1. 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

35.3 Memory map and register descriptions

This section describes in detail all I2C registers accessible to the end user.

I2C memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_81C0	I2C Address Register 1 (I2C0_A1)	8	R/W	00h	35.3.1/754
FFFF_81C1	I2C Frequency Divider register (I2C0_F)	8	R/W	00h	35.3.2/755
FFFF_81C2	I2C Control Register 1 (I2C0_C1)	8	R/W	00h	35.3.3/756
FFFF_81C3	I2C Status register (I2C0_S)	8	R/W	80h	35.3.4/758
FFFF_81C4	I2C Data I/O register (I2C0_D)	8	R/W	00h	35.3.5/759
FFFF_81C5	I2C Control Register 2 (I2C0_C2)	8	R/W	00h	35.3.6/760
FFFF_81C6	I2C Programmable Input Glitch Filter register (I2C0_FLT)	8	R/W	00h	35.3.7/761
FFFF_81C7	I2C Range Address register (I2C0_RA)	8	R/W	00h	35.3.8/762
FFFF_81C8	I2C SMBus Control and Status register (I2C0_SMB)	8	R/W	00h	35.3.9/762
FFFF_81C9	I2C Address Register 2 (I2C0_A2)	8	R/W	C2h	35.3.10/ 764
FFFF_81CA	I2C SCL Low Timeout Register High (I2C0_SLTH)	8	R/W	00h	35.3.11/ 764
FFFF_81CB	I2C SCL Low Timeout Register Low (I2C0_SLTL)	8	R/W	00h	35.3.12/ 765
FFFF_81D0	I2C Address Register 1 (I2C1_A1)	8	R/W	00h	35.3.1/754
FFFF_81D1	I2C Frequency Divider register (I2C1_F)	8	R/W	00h	35.3.2/755
FFFF_81D2	I2C Control Register 1 (I2C1_C1)	8	R/W	00h	35.3.3/756
FFFF_81D3	I2C Status register (I2C1_S)	8	R/W	80h	35.3.4/758
FFFF_81D4	I2C Data I/O register (I2C1_D)	8	R/W	00h	35.3.5/759
FFFF_81D5	I2C Control Register 2 (I2C1_C2)	8	R/W	00h	35.3.6/760
FFFF_81D6	I2C Programmable Input Glitch Filter register (I2C1_FLT)	8	R/W	00h	35.3.7/761
FFFF_81D7	I2C Range Address register (I2C1_RA)	8	R/W	00h	35.3.8/762
FFFF_81D8	I2C SMBus Control and Status register (I2C1_SMB)	8	R/W	00h	35.3.9/762
FFFF_81D9	I2C Address Register 2 (I2C1_A2)	8	R/W	C2h	35.3.10/ 764
FFFF_81DA	I2C SCL Low Timeout Register High (I2C1_SLTH)	8	R/W	00h	35.3.11/ 764
FFFF_81DB	I2C SCL Low Timeout Register Low (I2C1_SLTL)	8	R/W	00h	35.3.12/ 765

35.3.1 I2C Address Register 1 (I2Cx_A1)

This register contains the slave address to be used by the I2C module.

Address: Base address + 0h offset

Bit	7	6	5	4	3	2	1	0
Read	AD[7:1]							0
Write								
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.

35.3.2 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	The MULT bits define 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

5–0
ICR
	ClockRate
	
	Prescales the bus clock for bit rate selection. This field and the MULT field determine the I2C baud rate, the SDA hold time, the SCL start hold time, and the SCL stop hold time. For a list of values corresponding to each ICR setting, see I2C divider and hold values.
	
	The SCL divider multiplied by multiplier factor (mul) determines the I2C baud rate.
	
	`I2C baud rate = bus speed (Hz)/(mul × SCL divider)`
	
	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 = bus 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 = bus 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 = bus period (s) × mul × SCL stop hold value`
	
	For example, if the bus speed is 8 MHz, the following table shows the possible hold time values with different ICR and MULT selections to achieve an I^2C baud rate of 100 kbit/s.

Table continues on the next page...

I2Cx_F field descriptions (continued)

Field	Description
	<table><tr><th rowspan="2">MULT</th><th rowspan="2">ICR</th><th colspan="3">Hold times (µs)</th></tr><tr><th>SDA</th><th>SCL Start</th><th>SCL Stop</th></tr><tr><td>2h</td><td>00h</td><td>3.500</td><td>3.000</td><td>5.500</td></tr><tr><td>1h</td><td>07h</td><td>2.500</td><td>4.000</td><td>5.250</td></tr><tr><td>1h</td><td>0Bh</td><td>2.250</td><td>4.000</td><td>5.250</td></tr><tr><td>0h</td><td>14h</td><td>2.125</td><td>4.250</td><td>5.125</td></tr><tr><td>0h</td><td>18h</td><td>1.125</td><td>4.750</td><td>5.125</td></tr></table>

35.3.3 I2C Control Register 1 (I2Cx_C1)

Address: Base address + 2h offset

Bit	7	6	5	4	3	2	1	0
Read	IICEN	IICIE	MST	TX	TXAK	0	WUEN	DMAEN
Write						RSTA		
Reset	0	0	0	0	0	0	0	0

I2Cx_C1 field descriptions

Field	Description
7 IICEN	I2C Enable Enables I2C module operation. 0 Disabled 1 Enabled
6 IICIE	I2C Interrupt Enable Enables I2C interrupt requests. 0 Disabled 1 Enabled
5 MST	Master Mode Select When the MST bit is changed from a 0 to a 1, a START signal is generated on the bus and master mode is selected. When this bit changes from a 1 to a 0, a STOP signal is generated and the mode of operation changes from master to slave. 0 Slave mode 1 Master mode
4 TX	Transmit Mode Select

Table continues on the next page...

I2Cx_C1 field descriptions (continued)

Field	Description
	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 the FACK bit 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 a one to this bit generates a repeated START condition provided it is the current master. This bit will always be read as zero. Attempting a repeat at the wrong time results in loss of arbitration.
1 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 The DMAEN bit enables or disables the DMA function. 0 All DMA signalling disabled. 1 DMA transfer is enabled and the following conditions trigger the DMA request: • While FACK = 0, a data byte is received, either address or data is transmitted. (ACK/NACK automatic) • While FACK = 0, the first byte received matches the A1 register or is general call address. If any address matching occurs, IAAS and TCF are set. If the direction of transfer is known from master to slave, then it is not required to check the SRW. With this assumption, DMA can also be used in this case. In other cases, if the master reads data from the slave, then it is required to rewrite the C1 register operation. With this assumption, DMA cannot be used. When FACK = 1, an address or a data byte is transmitted.

35.3.4 I2C Status register (I2Cx_S)

Address: Base address + 3h offset

Bit	7	6	5	4	3	2	1	0
Read	TCF	IAAS	BUSY	ARBL	RAM	SRW	IICIF	RXAK
Write				w1c			w1c	
Reset	1	0	0	0	0	0	0	0

I2Cx_S field descriptions

Field	Description
7 TCF	Transfer Complete Flag This bit sets on the completion of a byte and acknowledge bit transfer. This bit is valid only during or immediately following a transfer to or from the I2C module. The TCF bit is cleared by reading the I2C data register in receive mode or by writing to the I2C data register in transmit mode. 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 slave primary address in the A1 register or range address in the RA register (which must be set to a nonzero value). • GCAEN is set and a general call is received. • 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. This bit sets before the ACK bit. The CPU must check the SRW bit and set TX/RX accordingly. Writing the C1 register with any value clears this bit. 0 Not addressed 1 Addressed as a slave
5 BUSY	Bus Busy 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 a one 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:

Table continues on the next page...

I2Cx_S field descriptions (continued)

Field	Description
	• Any nonzero calling address is received that matches the address in the RA register. • The RMEN bit is set and 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 a 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 0 No interrupt pending 1 Interrupt pending
0 RXAK	Receive Acknowledge 0 Acknowledge signal was received after the completion of one byte of data transmission on the bus 1 No acknowledge signal detected

35.3.5 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
7–0 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).

35.3.6 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.

Table continues on the next page...

I2Cx_C2 field descriptions (continued)

Field	Description
	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 slave address matching for addresses between the values of the A1 and RA registers. When this bit is set, a slave address match occurs for any address greater than the value of the A1 register and less than or equal to the value of the RA register. 0 Range mode disabled. No address match 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.
2–0 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.

35.3.7 I2C Programmable Input Glitch Filter register (I2Cx_FLT)

Address: Base address + 6h offset

Bit	7	6	5	4	3	2	1	0
Read	Reserved	0		0	FLT			
Write								
Reset	0	0	0	0	0	0	0	0

I2Cx_FLT field descriptions

Field	Description
7 Reserved	This field is reserved. Writing this bit has no effect.
6–5 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.
3–0 FLT	I2C Programmable Filter Factor Controls the width of the glitch, in terms of bus clock cycles, that the filter must absorb. For any glitch whose size is less than or equal to this width setting, the filter does not allow the glitch to pass. 0h No filter/bypass 1-Fh Filter glitches up to width of *n* bus clock cycles, where *n*=1-15d

35.3.8 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. Any nonzero write enables this register. This register's use is similar to that of the A1 register, but in addition this register can be considered a maximum boundary in range matching mode.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

35.3.9 I2C SMBus Control and Status register (I2Cx_SMB)

NOTE

When the SCL and SDA signals are held high for a length of time greater than the high timeout period, the SHTF1 flag sets. Before reaching this threshold, while the system is detecting how long these signals are being held high, a master assumes that the bus is free. However, the SHTF1 bit rises in the bus transmission process with the idle bus state.

NOTE

When the TCKSEL bit is set, there is no need to monitor the SHTF1 bit because the bus speed is too high to match the protocol of SMBus.

Address: Base address + 8h offset

Bit	7	6	5	4	3	2	1	0
Read	FACK	ALERTEN	SIICAEN	TCKSEL	SLTF	SHTF1	SHTF2	SHTF2IE
Write					w1c		w1c	
Reset	0	0	0	0	0	0	0	0

I2Cx_SMB field descriptions

Field	Description
7 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.
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 bus clock / 64 1 Timeout counter counts at the frequency of the bus 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 zero. 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. 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 a 1 to it. 0 No SCL high and SDA low timeout occurs 1 SCL high and SDA low timeout occurs

Table continues on the next page...

I2Cx_SMB field descriptions (continued)

Field	Description
0 SHTF2IE	SHTF2 Interrupt Enable Enables SCL high and SDA low timeout interrupt. 0 SHTF2 interrupt is disabled 1 SHTF2 interrupt is enabled

35.3.10 I2C Address Register 2 (I2Cx_A2)

Address: Base address + 9h offset

Bit	7	6	5	4	3	2	1	0
Read Write	SAD							0
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.

35.3.11 I2C SCL Low Timeout Register High (I2Cx_SLTH)

Address: Base address + Ah offset

Bit	7	6	5	4	3	2	1	0
Read Write	SSLT[15:8]							
Reset	0	0	0	0	0	0	0	0

I2Cx_SLTH field descriptions

Field	Description
7–0 SSLT[15:8]	Most significant byte of SCL low timeout value that determines the timeout period of SCL low.

35.3.12 I2C SCL Low Timeout Register Low (I2Cx_SLTL)

Address: Base address + Bh offset

Bit	7	6	5	4	3	2	1	0
Read Write	SSLT[7:0]							
Reset	0	0	0	0	0	0	0	0

I2Cx_SLTL field descriptions

Field	Description
7–0 SSLT[7:0]	Least significant byte of SCL low timeout value that determines the timeout period of SCL low.

35.4 Functional description

This section provides a comprehensive functional description of the I2C module.

35.4.1 I2C protocol

The I2C bus system uses a serial data line (SDA) and a serial clock line (SCL) for data transfers. All devices connected to it must have open drain or open collector outputs. A logic AND function is exercised on both lines with external pull-up resistors. The value of these resistors depends on the system.

Normally, a standard instance of communication is composed of four parts:

1. START signal
2. Slave address transmission
3. Data transfer
4. STOP signal

The STOP signal should not be confused with the CPU STOP instruction. The following figure illustrates I2C bus system communication.

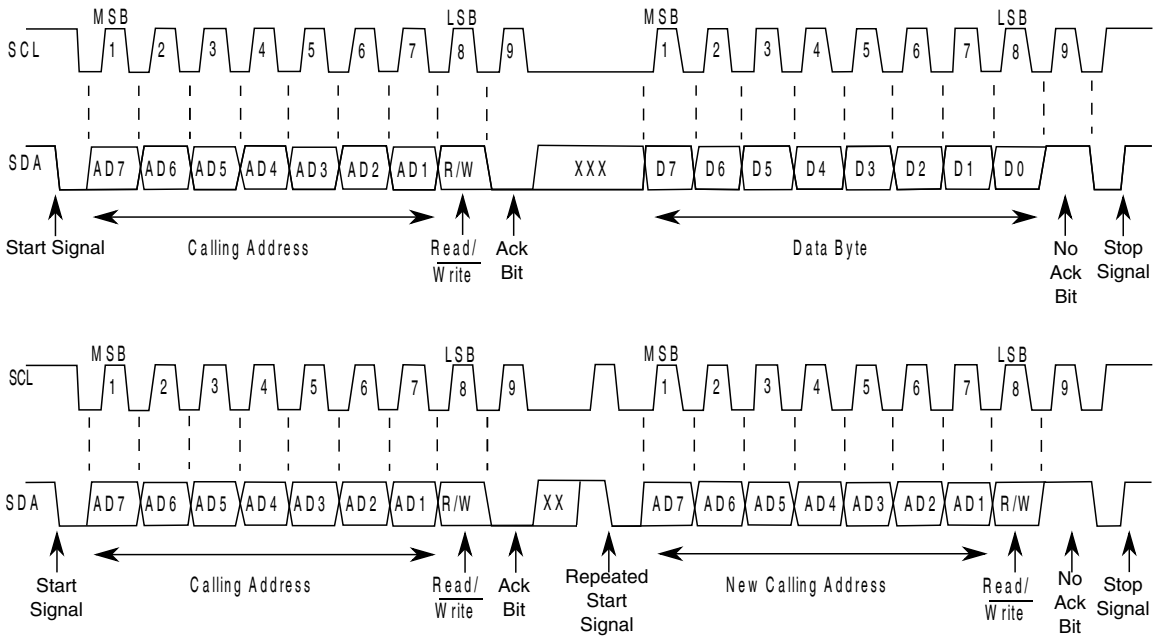


Figure 35-38. I2C bus transmission signals

35.4.1.1 START signal

The bus is free when no master device is engaging the bus (both SCL and SDA are high). When the bus is free, a master may initiate communication by sending a START signal. A START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer—each data transfer might contain several bytes of data—and brings all slaves out of their idle states.

35.4.1.2 Slave address transmission

Immediately after the START signal, the first byte of a data transfer is the slave address transmitted by the master. This address is a 7-bit calling address followed by an R/$\overline{W}$ bit. The R/$\overline{W}$ bit tells the slave the desired direction of data transfer.

- 1 = Read transfer: The slave transmits data to the master
- 0 = Write transfer: The master transmits data to the slave

Only the slave with a calling address that matches the one transmitted by the master responds by sending an acknowledge bit. The slave sends the acknowledge bit by pulling SDA low at the ninth clock.

No two slaves in the system can have the same address. If the I2C module is the master, it must not transmit an address that is equal to its own slave address. The I2C module cannot be master and slave at the same time. However, if arbitration is lost during an address cycle, the I2C module reverts to slave mode and operates correctly even if it is being addressed by another master.

35.4.1.3 Data transfers

When successful slave addressing is achieved, data transfer can proceed on a byte-by-byte basis in the direction specified by the R/$\overline{W}$ bit sent by the calling master.

All transfers that follow an address cycle are referred to as data transfers, even if they carry subaddress information for the slave device.

Each data byte is 8 bits long. Data may be changed only while SCL is low. Data must be held stable while SCL is high. There is one clock pulse on SCL for each data bit, and the MSB is transferred first. Each data byte is followed by a ninth (acknowledge) bit, which is signaled from the receiving device by pulling SDA low at the ninth clock. In summary, one complete data transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master in the ninth bit, the slave must leave SDA high. The master interprets the failed acknowledgement as an unsuccessful data transfer.

If the master receiver does not acknowledge the slave transmitter after a data byte transmission, the slave interprets it as an end to data transfer and releases the SDA line.

In the case of a failed acknowledgement by either the slave or master, the data transfer is aborted and the master does one of two things:

- Relinquishes the bus by generating a STOP signal.
- Commences a new call by generating a repeated START signal.

35.4.1.4 STOP signal

The master can terminate the communication by generating a STOP signal to free the bus. A STOP signal is defined as a low-to-high transition of SDA while SCL is asserted.

The master can generate a STOP signal even if the slave has generated an acknowledgement, at which point the slave must release the bus.

35.4.1.5 Repeated START signal

The master may generate a START signal followed by a calling command without generating a STOP signal first. This action is called a repeated START. The master uses a repeated START to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.

35.4.1.6 Arbitration procedure

The I2C bus is a true multimaster bus that allows more than one master to be connected on it.

If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock. The bus clock's low period is equal to the longest clock low period, and the high period is equal to the shortest one among the masters.

The relative priority of the contending masters is determined by a data arbitration procedure. A bus master loses arbitration if it transmits logic level 1 while another master transmits logic level 0. The losing masters immediately switch to slave receive mode and stop driving SDA output. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, hardware sets a status bit to indicate the loss of arbitration.

35.4.1.7 Clock synchronization

Because wire AND logic is performed on SCL, a high-to-low transition on SCL affects all devices connected on the bus. The devices start counting their low period and, after a device's clock has gone low, that device holds SCL low until the clock reaches its high state. However, the change of low to high in this device clock might not change the state of SCL if another device clock is still within its low period. Therefore, the synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time; see the following diagram. When all applicable devices have counted off their low period, the synchronized clock SCL is released and pulled high. Afterward there is no difference between the device clocks and the state of SCL, and all devices start counting their high periods. The first device to complete its high period pulls SCL low again.

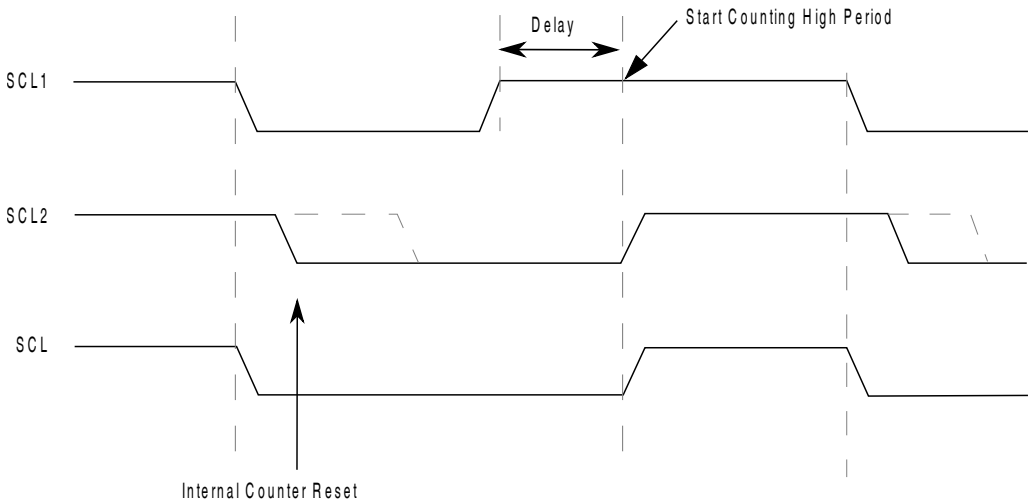


Figure 35-39. I2C clock synchronization

35.4.1.8 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfers. A slave device may hold SCL low after completing a single byte transfer (9 bits). In this case, it halts the bus clock and forces the master clock into wait states until the slave releases SCL.

35.4.1.9 Clock stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master drives SCL low, a slave can drive SCL low for the required period and then release it. If the slave's SCL low period is greater than the master's SCL low period, the resulting SCL bus signal's low period is stretched. In other words, the SCL bus signal's low period is increased to be the same length as the slave's SCL low period.

35.4.1.10 I2C divider and hold values

NOTE

For some cases on some devices, the SCL divider value may vary by +/-2 or +/-4 when ICR’s value ranges from 00h to 0Fh. These potentially varying SCL divider values are highlighted in the following table. For the actual SCL divider values for your device, see the chip-specific details about the I2C module.

Table 35-41. 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

35.4.2 10-bit address

For 10-bit addressing, 0x11110 is used for the first 5 bits of the first address byte. Various combinations of read/write formats are possible within a transfer that includes 10-bit addressing.

35.4.2.1 Master-transmitter addresses a slave-receiver

The transfer direction is not changed. When a 10-bit address follows a START condition, each slave compares the first 7 bits of the first byte of the slave address (11110XX) with its own address and tests whether the eighth bit (R/$\overline{W}$ direction bit) is 0. It is possible that more than one device finds a match and generates an acknowledge (A1). Each slave that finds a match compares the 8 bits of the second byte of the slave address with its own address, but only one slave finds a match and generates an acknowledge (A2). The matching slave remains addressed by the master until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different slave address.

Table 35-42. 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.

35.4.2.2 Master-receiver addresses a slave-transmitter

The transfer direction is changed after the second R/$\overline{W}$ bit. Up to and including acknowledge bit A2, the procedure is the same as that described for a master-transmitter addressing a slave-receiver. After the repeated START condition (Sr), a matching slave remembers that it was addressed before. This slave then checks whether the first seven bits of the first byte of the slave address following Sr are the same as they were after the START condition (S), and it tests whether the eighth (R/$\overline{W}$) bit is 1. If there is a match, the slave considers that it has been addressed as a transmitter and generates acknowledge A3. The slave-transmitter remains addressed until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different slave address.

After a repeated START condition (Sr), all other slave devices also compare the first seven bits of the first byte of the slave address with their own addresses and test the eighth (R/$\overline{W}$) bit. However, none of them are addressed because R/$\overline{W}$ = 1 (for 10-bit devices), or the 11110XX slave address (for 7-bit devices) does not match.

Table 35-43. 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.

35.4.3 Address matching

All received addresses can be requested in 7-bit or 10-bit address format.

- AD[7:1] in Address Register 1, which contains the I2C primary slave address, always participates in the address matching process. It provides a 7-bit address.
- If the ADEXT bit is set, AD[10:8] in Control Register 2 participates in the address matching process. It extends the I2C primary slave address to a 10-bit address.

Additional conditions that affect address matching include:

- If the GCAEN bit is set, general call participates the address matching process.
- If the ALERTEN bit is set, alert response participates the address matching process.
- If the SIICAEN bit is set, Address Register 2 participates in the address matching process.
- If the Range Address register is programmed to a nonzero value, the range address itself participates in the address matching process.
- If the RMEN bit is set, any address within the range of values of Address Register 1 and the Range Address register participates in the address matching process. The Range Address register must be programmed to a value greater than the value of Address Register 1.

When the I2C module responds to one of these addresses, it acts as a slave-receiver and the IAAS bit is set after the address cycle. Software must read the Data register after the first byte transfer to determine that the address is matched.

35.4.4 System management bus specification

SMBus provides a control bus for system and power management related tasks. A system can use SMBus to pass messages to and from devices instead of tripping individual control lines. Removing the individual control lines reduces pin count. Accepting messages ensures future expandability. With the system management bus, a device can provide manufacturer information, tell the system what its model/part number is, save its state for a suspend event, report different types of errors, accept control parameters, and return its status.

35.4.4.1 Timeouts

The $T_{TIMEOUT,MIN}$ parameter allows a master or slave to conclude that a defective device is holding the clock low indefinitely or a master is intentionally trying to drive devices off the bus. The slave device must release the bus (stop driving the bus and let SCL and SDA float high) when it detects any single clock held low longer than $T_{TIMEOUT,MIN}$. Devices that have detected this condition must reset their communication and be able to receive a new START condition within the timeframe of $T_{TIMEOUT,MAX}$.

SMBus defines a clock low timeout, $T_{TIMEOUT}$, of 35 ms, specifies $T_{LOW:SEXT}$ as the cumulative clock low extend time for a slave device, and specifies $T_{LOW:MEXT}$ as the cumulative clock low extend time for a master device.

35.4.4.1.1 SCL low timeout

If the SCL line is held low by a slave device on the bus, no further communication is possible. Furthermore, the master cannot force the SCL line high to correct the error condition. To solve this problem, the SMBus protocol specifies that devices participating in a transfer must detect any clock cycle held low longer than a timeout value condition. Devices that have detected the timeout condition must reset the communication. When the I2C module is an active master, if it detects that SMBCLK low has exceeded the value of $T_{TIMEOUT,MIN}$, it must generate a stop condition within or after the current data byte in the transfer process. When the I2C module is a slave, if it detects the $T_{TIMEOUT,MIN}$ condition, it resets its communication and is then able to receive a new START condition.

35.4.4.1.2 SCL high timeout

When the I2C module has determined that the SMBCLK and SMBDAT signals have been high for at least $T_{HIGH:MAX}$, it assumes that the bus is idle.

A HIGH timeout occurs after a START condition appears on the bus but before a STOP condition appears on the bus. Any master detecting this scenario can assume the bus is free when either of the following occurs:

- SHTF1 rises.
- The BUSY bit is high and SHTF1 is high.

When the SMBDAT signal is low and the SMBCLK signal is high for a period of time, another kind of timeout occurs. The time period must be defined in software. SHTF2 is used as the flag when the time limit is reached. This flag is also an interrupt resource, so it triggers IICIF.

35.4.4.1.3 CSMBCLK TIMEOUT MEXT and CSMBCLK TIMEOUT SEXT

The following figure illustrates the definition of the timeout intervals $T_{LOW:SEXT}$ and $T_{LOW:MEXT}$. When in master mode, the I2C module must not cumulatively extend its clock cycles for a period greater than $T_{LOW:MEXT}$ within a byte, where each byte is defined as START-to-ACK, ACK-to-ACK, or ACK-to-STOP. When CSMBCLK TIMEOUT MEXT occurs, SMBus MEXT rises and also triggers the SLTF.

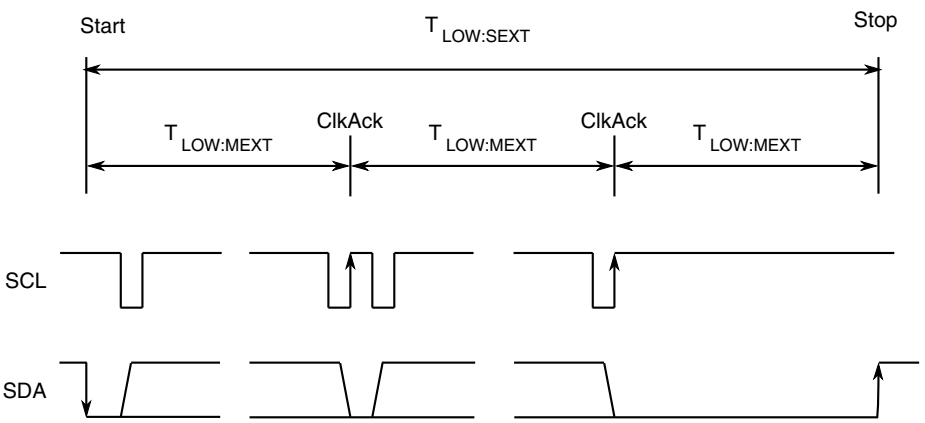


Figure 35-40. Timeout measurement intervals

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.

35.4.4.2 FAST ACK and NACK

To improve reliability and communication robustness, implementation of packet error checking (PEC) by SMBus devices is optional for SMBus devices but required for devices participating in and only during the address resolution protocol (ARP) process. The PEC is a CRC-8 error checking byte, calculated on all the message bytes. The PEC is appended to the message by the device that supplied the last data byte. If the PEC is present but not correct, a NACK is issued by the receiver. Otherwise an ACK is issued. To calculate the CRC-8 by software, this module can hold the SCL line low after receiving the eighth SCL (8th bit) if this byte is a data byte. So software can determine whether an ACK or NACK should be sent to the bus by setting or clearing the TXAK bit if the FACK (fast ACK/NACK enable) bit is enabled.

SMBus requires a device always to acknowledge its own address, as a mechanism to detect the presence of a removable device (such as a battery or docking station) on the bus. In addition to indicating a slave device busy condition, SMBus uses the NACK mechanism to indicate the reception of an invalid command or invalid data. Because such a condition may occur on the last byte of the transfer, SMBus devices are required to have the ability to generate the not acknowledge after the transfer of each byte and before the completion of the transaction. This requirement is important because SMBus does not provide any other resend signaling. This difference in the use of the NACK signaling has implications on the specific implementation of the SMBus port, especially in devices that handle critical system data such as the SMBus host and the SBS components.

NOTE

In the last byte of master receive slave transmit mode, the master must send a NACK to the bus, so FACK must be switched off before the last byte transmits.

35.4.5 Resets

The I2C module is disabled after a reset. The I2C module cannot cause a core reset.

35.4.6 Interrupts

The I2C module generates an interrupt when any of the events in the following table occur, provided that the IICIE bit is set. The interrupt is driven by the IICIF bit (of the I2C Status Register) and masked with the IICIE bit (of the I2C Control Register 1). The IICIF bit must be cleared (by software) by writing 1 to it in the interrupt routine. The

SMBus timeouts interrupt is driven by SLTF and masked with the IICIE bit. The SLTF bit must be cleared by software by writing 1 to it in the interrupt routine. You can determine the interrupt type by reading the Status Register.

NOTE

In master receive mode, the FACK bit must be set to zero before the last byte transfer.

Table 35-44. 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
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

35.4.6.1 Byte transfer interrupt

The Transfer Complete Flag (TCF) bit is set at the falling edge of the ninth clock to indicate the completion of a byte and acknowledgement transfer. When FACK is enabled, TCF is then set at the falling edge of eighth clock to indicate the completion of byte.

35.4.6.2 Address detect interrupt

When the calling address matches the programmed slave address (I2C Address Register) or when the GCAEN bit is set and a general call is received, the IAAS bit in the Status Register is set. The CPU is interrupted, provided the IICIE bit is set. The CPU must check the SRW bit and set its Tx mode accordingly.

35.4.6.3 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.

35.4.6.4 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.

35.4.6.5 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.

35.4.7 Programmable input glitch filter

An I2C glitch filter has been added outside legacy I2C modules but within the I2C package. This filter can absorb glitches on the I2C clock and data lines for the I2C module. The width of the glitch to absorb can be specified in terms of the number of (half) bus clock cycles. A single Programmable Input Glitch Filter control register is provided. Effectively, any down-up-down or up-down-up transition on the data line that

occurs within the number of clock cycles programmed in this register is ignored by the I2C module. The programmer must specify the size of the glitch (in terms of bus clock cycles) for the filter to absorb and not pass.

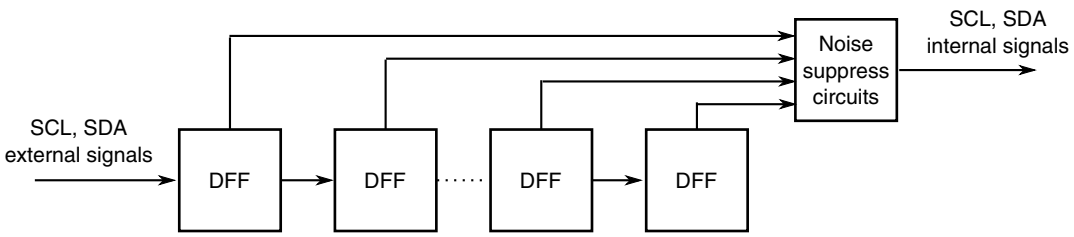


Figure 35-41. Programmable input glitch filter diagram

35.4.8 Address matching wakeup

When a primary, range, or general call address match occurs when the I2C module is in slave receive mode, the MCU wakes from a low power mode with no peripheral bus 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. The SCL line is not held low until the I2C module resets after address matching. The main purpose of this feature is to wake the MCU from a low power mode where no peripheral bus is running. When the MCU is in such a mode: addressing as a slave, slave read/write, and sending an acknowledge bit are not fully supported. To avoid I2C transfer problems resulting from this situation, firmware should prevent the MCU execution of a STOP instruction when the I2C module is in the middle of a transfer.

35.4.9 DMA support

If the DMAEN bit is cleared and the IICIE bit is set, an interrupt condition generates an interrupt request. If the DMAEN bit is set and the IICIE bit is set, an interrupt condition generates a DMA request instead. DMA requests are generated by the transfer complete flag (TCF).

If the DMAEN bit is set, the only arbitration lost is to another I2C module (error), and SCL low timeouts (error) generate CPU interrupts. All other events initiate a DMA transfer.

NOTE

Before the last byte of master receive mode, TXAK must be set to send a NACK after the last byte's transfer. Therefore, the DMA must be disabled before the last byte's transfer.

NOTE

In 10-bit address mode transmission, the addresses to send occupy 2-3 bytes. During this transfer period, the DMA must be disabled because the C1 register is written to send a repeat start or to change the transfer direction.

35.5 Initialization/application information

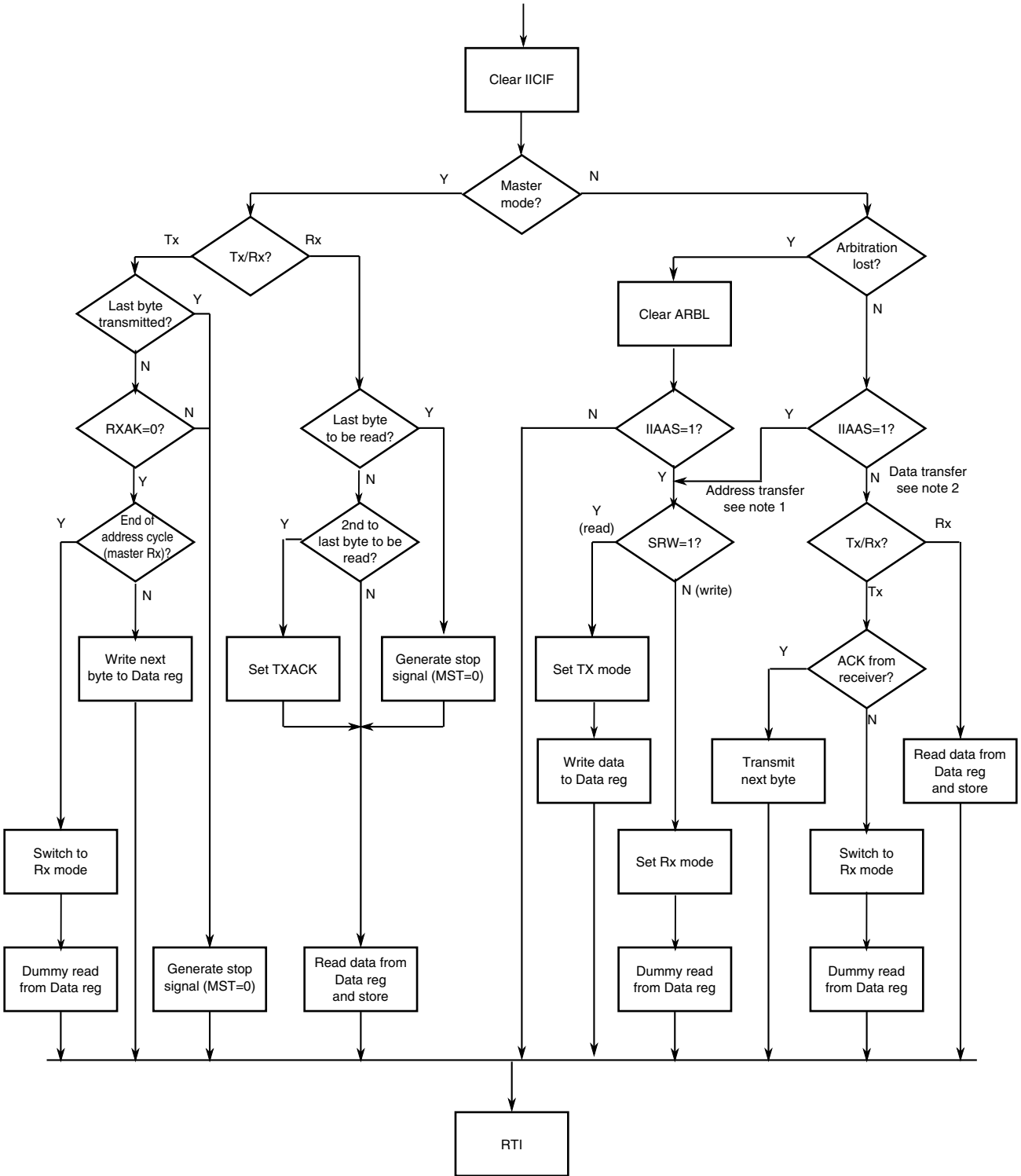
Module Initialization (Slave)

1. Write: Control Register 2
   - to enable or disable general call
   - to select 10-bit or 7-bit addressing mode
2. Write: Address Register 1 to set the slave address
3. Write: Control Register 1 to enable the I2C module and interrupts
4. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
5. Initialize RAM variables used to achieve the routine shown in the following figure

Module Initialization (Master)

1. Write: Frequency Divider register to set the I2C baud rate (see example in description of ICR)
2. Write: Control Register 1 to enable the I2C module and interrupts
3. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
4. Initialize RAM variables used to achieve the routine shown in the following figure
5. Write: Control Register 1 to enable TX
6. Write: Control Register 1 to enable MST (master mode)
7. Write: Data register with the address of the target slave (the LSB of this byte determines whether the communication is master receive or transmit)

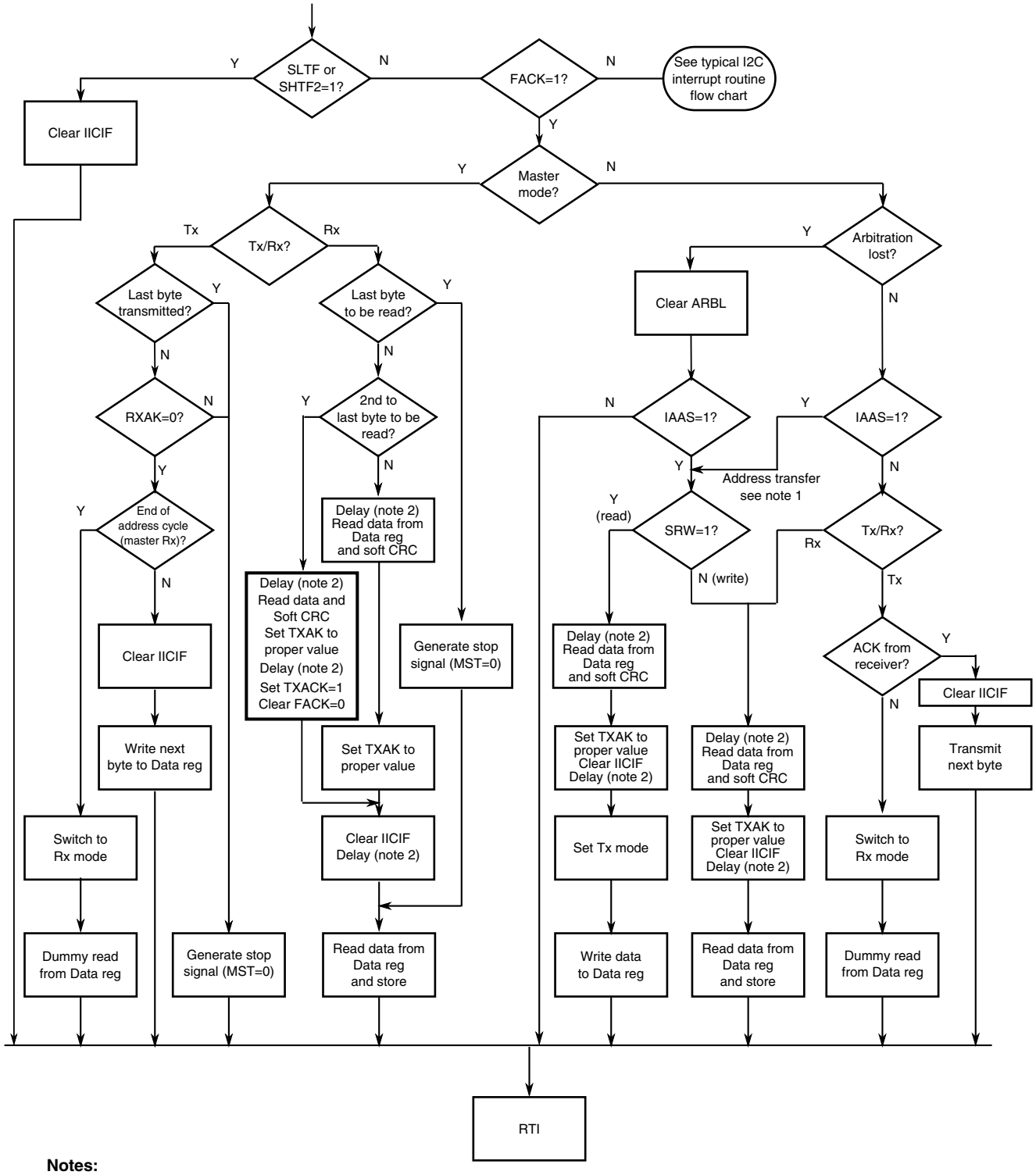
The routine shown in the following figure encompasses both master and slave I2C operations. For slave operation, an incoming I2C message that contains the proper address begins I2C communication. For master operation, communication must be initiated by writing the Data register.



Notes:

1. If general call is enabled, check to determine if the received address is a general call address (0x00). If the received address is a general call address, the general call must be handled by user software.
2. When 10-bit addressing addresses a slave, the slave sees an interrupt following the first byte of the extended address. Ensure that for this interrupt, the contents of the Data register are ignored and not treated as a valid data transfer.

Figure 35-42. Typical I2C interrupt routine



Notes:

1. If general call or SIICAEN is enabled, check to determine if the received address is a general call address (0x00) or an SMBus device default address. In either case, they must be handled by user software.
2. In receive mode, one bit time delay may be needed before the first and second data reading.

Figure 35-43. Typical I2C SMBus interrupt routine

Chapter 36
Serial Communication Interface (SCI) / Universal Asynchronous Receiver/Transmitter (UART)

36.1 Introduction

The UART allows asynchronous serial communication with peripheral devices and CPUs.

36.1.1 Features

The UART includes the following features:

- Full-duplex operation
- Standard mark/space non-return-to-zero (NRZ) format
- 13-bit baud rate selection with /32 fractional divide, based on the module clock frequency
- Programmable 8-bit or 9-bit data format
- Separately enabled transmitter and receiver
- Programmable transmitter output polarity
- Programmable receive input polarity
- 13-bit break character option
- 11-bit break character detection option
- Independent FIFO structure for transmit and receive
- Two receiver wakeup methods:

- Idle line wakeup
- Address mark wakeup

- Address match feature in the receiver to reduce address mark wakeup ISR overhead
- Ability to select MSB or LSB to be first bit on wire
- Hardware flow control support for request to send (RTS) and clear to send (CTS) signals
- Support for ISO 7816 protocol to interface with SIM cards and smart cards
  - Support for T=0 and T=1 protocols
  - Automatic retransmission of NACK'd packets with programmable retry threshold
  - Support for 11 and 12 ETU transfers
  - Detection of initial packet and automated transfer parameter programming
  - Interrupt-driven operation with seven ISO-7816 specific interrupts:
    - Wait time violated
    - Character wait time violated
    - Block wait time violated
    - Initial frame detected
    - Transmit error threshold exceeded
    - Receive error threshold exceeded
    - Guard time violated
- Interrupt-driven operation with 12 flags, not specific to ISO-7816 support
  - Transmitter data buffer at or below watermark
  - Transmission complete
  - Receiver data buffer at or above watermark
  - Idle receiver input
  - Receiver data buffer overrun
  - Receiver data buffer underflow
  - Transmit data buffer overflow

  - Noise error
  - Framing error
  - Parity error
  - Active edge on receive pin
  - LIN break detect
- Receiver framing error detection
- Hardware parity generation and checking
- 1/16 bit-time noise detection
- DMA interface

36.1.2 Modes of operation

The UART functions in the same way in all the normal modes.

It has the following low power modes:

- Wait mode
- Stop mode

36.1.2.1 Run mode

This is the normal mode of operation.

36.1.2.2 Wait mode

UART operation in the Wait mode depends on the state of the C1[UARTSWAI] field.

- If C1[UARTSWAI] is cleared, and the CPU is in Wait mode, the UART operates normally.
- If C1[UARTSWAI] is set, and the CPU is in Wait mode, the UART clock generation ceases and the UART module enters a power conservation state.

C1[UARTSWAI] does not initiate any power down or power up procedures for the ISO-7816 smartcard interface.

Setting C1[UARTSWAI] does not affect the state of the C2[RE] or C2[TE].

If C1[UARTSWAI] is set, any ongoing transmission or reception stops at the Wait mode entry. The transmission or reception resumes when either an internal or external interrupt brings the CPU out of Wait mode. Bringing the CPU out of Wait mode by reset aborts any ongoing transmission or reception and resets the UART.

36.1.2.3 Stop mode

The UART is inactive during Stop mode for reduced power consumption. The STOP instruction does not affect the UART register states, but the UART module clock is disabled. The UART operation resumes after an external interrupt brings the CPU out of Stop mode. Bringing the CPU out of Stop mode by reset aborts any ongoing transmission or reception and resets the UART. Entering or leaving Stop mode does not initiate any power down or power up procedures for the ISO-7816 smartcard interface.

36.2 UART signal descriptions

The UART signals are shown in the following table.

Table 36-1. UART signal descriptions

Signal	Description	I/O
CTS	Clear to send	I
$\overline{\text{RTS}}$	Request to send	O
RXD	Receive data	I
TXD	Transmit data	O

36.2.1 Detailed signal descriptions

The detailed signal descriptions of the UART are shown in the following table.

Table 36-2. UART—Detailed signal descriptions

Signal	I/O	Description	
CTS	I	Clear to send. Indicates whether the UART can start transmitting data when flow control is enabled.	
		State meaning	Asserted—Data transmission can start. Negated—Data transmission cannot start.
		Timing	Assertion—When transmitting device's RTS asserts. Negation—When transmitting device's RTS deasserts.
RTS	O	Request to send. When driven by the receiver, indicates whether the UART is ready to receive data. When driven by the transmitter, can enable an external transceiver during transmission.	
		State meaning	Asserted—When driven by the receiver, ready to receive data. When driven by the transmitter, enable the external transmitter. Negated—When driven by the receiver, not ready to receive data. When driven by the transmitter, disable the external transmitter.
		Timing	Assertion—Can occur at any time; can assert asynchronously to the other input signals. Negation—Can occur at any time; can deassert asynchronously to the other input signals.
RXD	I	Receive data. Serial data input to receiver.	
		State meaning	Whether RXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.
		Timing	Sampled at a frequency determined by the module clock divided by the baud rate.
TXD	O	Transmit data. Serial data output from transmitter.	
		State meaning	Whether TXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.
		Timing	Driven at the beginning or within a bit time according to the bit encoding method along with other configuration settings. Otherwise, transmissions are independent of reception timing.

36.3 Memory map and registers

This section provides a detailed description of all memory and registers.

Accessing reserved addresses within the memory map results in a transfer error. None of the contents of the implemented addresses are modified as a result of that access.

Only byte accesses are supported.

UART memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_8120	UART Baud Rate Registers: High (UART0_BDH)	8	R/W	00h	36.3.1/791
FFFF_8121	UART Baud Rate Registers: Low (UART0_BDL)	8	R/W	04h	36.3.2/792
FFFF_8122	UART Control Register 1 (UART0_C1)	8	R/W	00h	36.3.3/793
FFFF_8123	UART Control Register 2 (UART0_C2)	8	R/W	00h	36.3.4/794
FFFF_8124	UART Status Register 1 (UART0_S1)	8	R	C0h	36.3.5/796
FFFF_8125	UART Status Register 2 (UART0_S2)	8	R/W	00h	36.3.6/799
FFFF_8126	UART Control Register 3 (UART0_C3)	8	R/W	00h	36.3.7/801
FFFF_8127	UART Data Register (UART0_D)	8	R/W	00h	36.3.8/802
FFFF_8128	UART Match Address Registers 1 (UART0_MA1)	8	R/W	00h	36.3.9/804
FFFF_8129	UART Match Address Registers 2 (UART0_MA2)	8	R/W	00h	36.3.10/ 804
FFFF_812A	UART Control Register 4 (UART0_C4)	8	R/W	00h	36.3.11/ 804
FFFF_812B	UART Control Register 5 (UART0_C5)	8	R/W	00h	36.3.12/ 805
FFFF_812C	UART Extended Data Register (UART0_ED)	8	R	00h	36.3.13/ 806
FFFF_812D	UART Modem Register (UART0_MODEM)	8	R/W	00h	36.3.14/ 807
FFFF_8130	UART FIFO Parameters (UART0_PFIFO)	8	R/W	See section	36.3.15/ 809
FFFF_8131	UART FIFO Control Register (UART0_CFIFO)	8	R/W	00h	36.3.16/ 810
FFFF_8132	UART FIFO Status Register (UART0_SFIFO)	8	R/W	C0h	36.3.17/ 811
FFFF_8133	UART FIFO Transmit Watermark (UART0_TWFIFO)	8	R/W	00h	36.3.18/ 812
FFFF_8134	UART FIFO Transmit Count (UART0_TCFIFO)	8	R	00h	36.3.19/ 813
FFFF_8135	UART FIFO Receive Watermark (UART0_RWFIFO)	8	R/W	01h	36.3.20/ 813
FFFF_8136	UART FIFO Receive Count (UART0_RCFIFO)	8	R	00h	36.3.21/ 814
FFFF_8138	UART 7816 Control Register (UART0_C7816)	8	R/W	00h	36.3.22/ 814
FFFF_8139	UART 7816 Interrupt Enable Register (UART0_IE7816)	8	R/W	00h	36.3.23/ 816
FFFF_813A	UART 7816 Interrupt Status Register (UART0_IS7816)	8	R/W	00h	36.3.24/ 817
FFFF_813B	UART 7816 Wait Parameter Register (UART0_WP7816T0)	8	R/W	0Ah	36.3.25/ 818
FFFF_813B	UART 7816 Wait Parameter Register (UART0_WP7816T1)	8	R/W	0Ah	36.3.26/ 819

Table continues on the next page...

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_813C	UART 7816 Wait N Register (UART0_WN7816)	8	R/W	00h	36.3.27/ 819
FFFF_813D	UART 7816 Wait FD Register (UART0_WF7816)	8	R/W	01h	36.3.28/ 820
FFFF_813E	UART 7816 Error Threshold Register (UART0_ET7816)	8	R/W	00h	36.3.29/ 820
FFFF_813F	UART 7816 Transmit Length Register (UART0_TL7816)	8	R/W	00h	36.3.30/ 821
FFFF_8140	UART Baud Rate Registers: High (UART1_BDH)	8	R/W	00h	36.3.1/791
FFFF_8141	UART Baud Rate Registers: Low (UART1_BDL)	8	R/W	04h	36.3.2/792
FFFF_8142	UART Control Register 1 (UART1_C1)	8	R/W	00h	36.3.3/793
FFFF_8143	UART Control Register 2 (UART1_C2)	8	R/W	00h	36.3.4/794
FFFF_8144	UART Status Register 1 (UART1_S1)	8	R	C0h	36.3.5/796
FFFF_8145	UART Status Register 2 (UART1_S2)	8	R/W	00h	36.3.6/799
FFFF_8146	UART Control Register 3 (UART1_C3)	8	R/W	00h	36.3.7/801
FFFF_8147	UART Data Register (UART1_D)	8	R/W	00h	36.3.8/802
FFFF_8148	UART Match Address Registers 1 (UART1_MA1)	8	R/W	00h	36.3.9/804
FFFF_8149	UART Match Address Registers 2 (UART1_MA2)	8	R/W	00h	36.3.10/ 804
FFFF_814A	UART Control Register 4 (UART1_C4)	8	R/W	00h	36.3.11/ 804
FFFF_814B	UART Control Register 5 (UART1_C5)	8	R/W	00h	36.3.12/ 805
FFFF_814C	UART Extended Data Register (UART1_ED)	8	R	00h	36.3.13/ 806
FFFF_814D	UART Modem Register (UART1_MODEM)	8	R/W	00h	36.3.14/ 807
FFFF_8150	UART FIFO Parameters (UART1_PFIFO)	8	R/W	See section	36.3.15/ 809
FFFF_8151	UART FIFO Control Register (UART1_CFIFO)	8	R/W	00h	36.3.16/ 810
FFFF_8152	UART FIFO Status Register (UART1_SFIFO)	8	R/W	C0h	36.3.17/ 811
FFFF_8153	UART FIFO Transmit Watermark (UART1_TWFIFO)	8	R/W	00h	36.3.18/ 812
FFFF_8154	UART FIFO Transmit Count (UART1_TCFIFO)	8	R	00h	36.3.19/ 813
FFFF_8155	UART FIFO Receive Watermark (UART1_RWFIFO)	8	R/W	01h	36.3.20/ 813
FFFF_8156	UART FIFO Receive Count (UART1_RCFIFO)	8	R	00h	36.3.21/ 814
FFFF_8158	UART 7816 Control Register (UART1_C7816)	8	R/W	00h	36.3.22/ 814

Table continues on the next page...

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_8159	UART 7816 Interrupt Enable Register (UART1_IE7816)	8	R/W	00h	36.3.23/ 816
FFFF_815A	UART 7816 Interrupt Status Register (UART1_IS7816)	8	R/W	00h	36.3.24/ 817
FFFF_815B	UART 7816 Wait Parameter Register (UART1_WP7816T0)	8	R/W	0Ah	36.3.25/ 818
FFFF_815B	UART 7816 Wait Parameter Register (UART1_WP7816T1)	8	R/W	0Ah	36.3.26/ 819
FFFF_815C	UART 7816 Wait N Register (UART1_WN7816)	8	R/W	00h	36.3.27/ 819
FFFF_815D	UART 7816 Wait FD Register (UART1_WF7816)	8	R/W	01h	36.3.28/ 820
FFFF_815E	UART 7816 Error Threshold Register (UART1_ET7816)	8	R/W	00h	36.3.29/ 820
FFFF_815F	UART 7816 Transmit Length Register (UART1_TL7816)	8	R/W	00h	36.3.30/ 821
FFFF_8160	UART Baud Rate Registers: High (UART2_BDH)	8	R/W	00h	36.3.1/791
FFFF_8161	UART Baud Rate Registers: Low (UART2_BDL)	8	R/W	04h	36.3.2/792
FFFF_8162	UART Control Register 1 (UART2_C1)	8	R/W	00h	36.3.3/793
FFFF_8163	UART Control Register 2 (UART2_C2)	8	R/W	00h	36.3.4/794
FFFF_8164	UART Status Register 1 (UART2_S1)	8	R	C0h	36.3.5/796
FFFF_8165	UART Status Register 2 (UART2_S2)	8	R/W	00h	36.3.6/799
FFFF_8166	UART Control Register 3 (UART2_C3)	8	R/W	00h	36.3.7/801
FFFF_8167	UART Data Register (UART2_D)	8	R/W	00h	36.3.8/802
FFFF_8168	UART Match Address Registers 1 (UART2_MA1)	8	R/W	00h	36.3.9/804
FFFF_8169	UART Match Address Registers 2 (UART2_MA2)	8	R/W	00h	36.3.10/ 804
FFFF_816A	UART Control Register 4 (UART2_C4)	8	R/W	00h	36.3.11/ 804
FFFF_816B	UART Control Register 5 (UART2_C5)	8	R/W	00h	36.3.12/ 805
FFFF_816C	UART Extended Data Register (UART2_ED)	8	R	00h	36.3.13/ 806
FFFF_816D	UART Modem Register (UART2_MODEM)	8	R/W	00h	36.3.14/ 807
FFFF_8170	UART FIFO Parameters (UART2_PFIFO)	8	R/W	See section	36.3.15/ 809
FFFF_8171	UART FIFO Control Register (UART2_CFIFO)	8	R/W	00h	36.3.16/ 810
FFFF_8172	UART FIFO Status Register (UART2_SFIFO)	8	R/W	C0h	36.3.17/ 811
FFFF_8173	UART FIFO Transmit Watermark (UART2_TWFIFO)	8	R/W	00h	36.3.18/ 812

Table continues on the next page...

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_8174	UART FIFO Transmit Count (UART2_TCFIFO)	8	R	00h	36.3.19/ 813
FFFF_8175	UART FIFO Receive Watermark (UART2_RWFIFO)	8	R/W	01h	36.3.20/ 813
FFFF_8176	UART FIFO Receive Count (UART2_RCFIFO)	8	R	00h	36.3.21/ 814
FFFF_8178	UART 7816 Control Register (UART2_C7816)	8	R/W	00h	36.3.22/ 814
FFFF_8179	UART 7816 Interrupt Enable Register (UART2_IE7816)	8	R/W	00h	36.3.23/ 816
FFFF_817A	UART 7816 Interrupt Status Register (UART2_IS7816)	8	R/W	00h	36.3.24/ 817
FFFF_817B	UART 7816 Wait Parameter Register (UART2_WP7816T0)	8	R/W	0Ah	36.3.25/ 818
FFFF_817B	UART 7816 Wait Parameter Register (UART2_WP7816T1)	8	R/W	0Ah	36.3.26/ 819
FFFF_817C	UART 7816 Wait N Register (UART2_WN7816)	8	R/W	00h	36.3.27/ 819
FFFF_817D	UART 7816 Wait FD Register (UART2_WF7816)	8	R/W	01h	36.3.28/ 820
FFFF_817E	UART 7816 Error Threshold Register (UART2_ET7816)	8	R/W	00h	36.3.29/ 820
FFFF_817F	UART 7816 Transmit Length Register (UART2_TL7816)	8	R/W	00h	36.3.30/ 821

36.3.1 UART Baud Rate Registers: High (UARTx_BDH)

This register, along with the BDL register, controls the prescale divisor for UART baud rate generation. To update the 13-bit baud rate setting (SBR[12:0]), first write to BDH to buffer the high half of the new value and then write to BDL. The working value in BDH does not change until BDL is written.

BDL is reset to a nonzero value, but after reset, the baud rate generator remains disabled until the first time the receiver or transmitter is enabled, that is, when C2[RE] or C2[TE] is set.

Address: Base address + 0h offset

Bit	7	6	5	4	3	2	1	0
Read	LBKDIE	RXEDGIE	0	SBR				
Write								
Reset	0	0	0	0	0	0	0	0

UARTx_BDH field descriptions

Field	Description
7 LBKDIE	LIN Break Detect Interrupt Enable Enables the LIN break detect flag, LBKDIF, to generate interrupt requests based on the state of LBKDDMAS. 0 LBKDIF interrupt requests disabled. 1 LBKDIF interrupt requests enabled.
6 RXEDGIE	RxD Input Active Edge Interrupt Enable Enables the receive input active edge, RXEDGIF, to generate interrupt requests. 0 Hardware interrupts from RXEDGIF disabled using polling. 1 RXEDGIF interrupt request enabled.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–0 SBR	UART Baud Rate Bits The baud rate for the UART is determined by the 13 SBR fields. See Baud rate generation for details. **NOTE:** • The baud rate generator is disabled until C2[TE] or C2[RE] is set for the first time after reset.The baud rate generator is disabled when SBR = 0. • Writing to BDH has no effect without writing to BDL, because writing to BDH puts the data in a temporary location until BDL is written.

36.3.2 UART Baud Rate Registers: Low (UARTx_BDL)

This register, along with the BDH register, controls the prescale divisor for UART baud rate generation. To update the 13-bit baud rate setting, SBR[12:0], first write to BDH to buffer the high half of the new value and then write to BDL. The working value in BDH does not change until BDL is written. BDL is reset to a nonzero value, but after reset, the baud rate generator remains disabled until the first time the receiver or transmitter is enabled, that is, when C2[RE] or C2[TE] is set.

Address: Base address + 1h offset

Bit	7	6	5	4	3	2	1	0
Read Write	SBR							
Reset	0	0	0	0	0	1	0	0

UARTx_BDL field descriptions

Field	Description
7–0 SBR	UART Baud Rate Bits The baud rate for the UART is determined by the 13 SBR fields. See Baud rate generation for details.

UARTx_BDL field descriptions (continued)

Field	Description
	NOTE: • The baud rate generator is disabled until C2[TE] or C2[RE] is set for the first time after reset.The baud rate generator is disabled when SBR = 0. • Writing to BDH has no effect without writing to BDL, because writing to BDH puts the data in a temporary location until BDL is written.

36.3.3 UART Control Register 1 (UARTx_C1)

This read/write register controls various optional features of the UART system.

Address: Base address + 2h offset

Bit	7	6	5	4	3	2	1	0
Read Write	LOOPS	UARTSWAI	RSRC	M	WAKE	ILT	PE	PT
Reset	0	0	0	0	0	0	0	0

UARTx_C1 field descriptions

Field	Description
7 LOOPS	Loop Mode Select When LOOPS is set, the RxD pin is disconnected from the UART and the transmitter output is internally connected to the receiver input. The transmitter and the receiver must be enabled to use the loop function. 0 Normal operation. 1 Loop mode where transmitter output is internally connected to receiver input. The receiver input is determined by RSRC.
6 UARTSWAI	UART Stops in Wait Mode 0 UART clock continues to run in Wait mode. 1 UART clock freezes while CPU is in Wait 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 Selects internal loop back mode. The receiver input is internally connected to transmitter output. 1 Single wire UART mode where the receiver input is connected to the transmit pin input signal.
4 M	9-bit or 8-bit Mode Select This field must be set when C7816[ISO_7816E] is set/enabled. 0 Normal—start + 8 data bits (MSB/LSB first as determined by MSBF) + stop. 1 Use—start + 9 data bits (MSB/LSB first as determined by MSBF) + stop.
3 WAKE	Receiver Wakeup Method Select Determines which condition wakes the UART: • Address mark in the most significant bit position of a received data character, or • An idle condition on the receive pin input signal.

Table continues on the next page...

UARTx_C1 field descriptions (continued)

Field	Description
	0 Idle line wakeup. 1 Address mark wakeup.
2 ILT	Idle Line Type Select 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 UART is programmed with ILT = 1, a logic of 1'b0 is automatically shifted after a received stop bit, therefore resetting the idle count. • In case the UART is programmed for IDLE line wakeup (RWU = 1 and WAKE = 0), ILT has no effect on when the receiver starts counting logic 1s as idle character bits. In idle line wakeup, an idle character is recognized at anytime the receiver sees 10, 11, or 12 1s depending on the M, PE, and C4[M10] fields. 0 Idle character bit count starts after start bit. 1 Idle character bit count starts after stop bit.
1 PE	Parity Enable Enables the parity function. When parity is enabled, parity function inserts a parity bit in the bit position immediately preceding the stop bit. This field must be set when C7816[ISO_7816E] is set/enabled. 0 Parity function disabled. 1 Parity function enabled.
0 PT	Parity Type Determines whether the UART generates and checks for even parity or odd parity. With even parity, an even number of 1s clears the parity bit and an odd number of 1s sets the parity bit. With odd parity, an odd number of 1s clears the parity bit and an even number of 1s sets the parity bit. This field must be cleared when C7816[ISO_7816E] is set/enabled. 0 Even parity. 1 Odd parity.

36.3.4 UART Control Register 2 (UARTx_C2)

This register can be read or written at any time.

Address: Base address + 3h offset

Bit	7	6	5	4	3	2	1	0
Read Write	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset	0	0	0	0	0	0	0	0

UARTx_C2 field descriptions

Field	Description
7 TIE	Transmitter Interrupt or DMA Transfer Enable.

Table continues on the next page...

UARTx_C2 field descriptions (continued)

Field	Description
	Enables S1[TDRE] to generate interrupt requests or DMA transfer requests, based on the state of C5[TDMAS]. **NOTE:** If C2[TIE] and C5[TDMAS] are both set, then TCIE must be cleared, and D[D] must not be written unless servicing a DMA request. 0 TDRE interrupt and DMA transfer requests disabled. 1 TDRE interrupt or DMA transfer requests enabled.
6 TCIE	Transmission Complete Interrupt or DMA Transfer Enable Enables the transmission complete flag, S1[TC], to generate interrupt requests . or DMA transfer requests based on the state of C5[TCDMAS] **NOTE:** If C2[TCIE] and C5[TCDMAS] are both set, then TIE must be cleared, and D[D] must not be written unless servicing a DMA request. 0 TC interrupt and DMA transfer requests disabled. 1 TC interrupt or DMA transfer requests enabled.
5 RIE	Receiver Full Interrupt or DMA Transfer Enable Enables S1[RDRF] to generate interrupt requests or DMA transfer requests, based on the state of C5[RDMAS]. 0 RDRF interrupt and DMA transfer requests disabled. 1 RDRF interrupt or DMA transfer requests enabled.
4 ILIE	Idle Line Interrupt Enable Enables the idle line flag, S1[IDLE], to generate interrupt requests 0 IDLE interrupt requests disabled. 1 IDLE interrupt requests enabled.
3 TE	Transmitter Enable Enables the UART transmitter. TE can be used to queue an idle preamble by clearing and then setting TE. When C7816[ISO_7816E] is set/enabled and C7816[TTYPE] = 1, this field is automatically cleared after the requested block has been transmitted. This condition is detected when TL7816[TLEN] = 0 and four additional characters are transmitted. 0 Transmitter off. 1 Transmitter on.
2 RE	Receiver Enable Enables the UART receiver. 0 Receiver off. 1 Receiver on.
1 RWU	Receiver Wakeup Control This field can be set to place the UART receiver in a standby state. RWU automatically clears when an RWU event occurs, that is, an IDLE event when C1[WAKE] is clear or an address match when C1[WAKE] is set. This field must be cleared when C7816[ISO_7816E] is set.

Table continues on the next page...

UARTx_C2 field descriptions (continued)

Field	Description
	NOTE: RWU must be set only with C1[WAKE] = 0 (wakeup on idle) if the channel is currently not idle. This can be determined by S2[RAF]. If the flag is set to wake up an IDLE event and the channel is already idle, it is possible that the UART will discard data. This is because the data must be received or a LIN break detected after an IDLE is detected before IDLE is allowed to reasserted. 0 Normal operation. 1 RWU enables the wakeup function and inhibits further receiver interrupt requests. Normally, hardware wakes the receiver by automatically clearing RWU.
0 SBK	Send Break Toggling SBK sends one break character from the following: See Transmitting break characters for the number of logic 0s for the different configurations. Toggling implies clearing the SBK field before the break character has finished transmitting. As long as SBK is set, the transmitter continues to send complete break characters (10, 11, or 12 bits, or 13 or 14 bits). • 10, 11, or 12 logic 0s if S2[BRK13] is cleared • 13 or 14 logic 0s if S2[BRK13] is set. This field must be cleared when C7816[ISO_7816E] is set. 0 Normal transmitter operation. 1 Queue break characters to be sent.

36.3.5 UART Status Register 1 (UARTx_S1)

The S1 register provides inputs to the MCU for generation of UART interrupts or DMA requests. This register can also be polled by the MCU to check the status of its fields. To clear a flag, the status register should be read followed by a read or write to D register, depending on the interrupt flag type. Other instructions can be executed between the two steps as long the handling of I/O is not compromised, but the order of operations is important for flag clearing. When a flag is configured to trigger a DMA request, assertion of the associated DMA done signal from the DMA controller clears the flag.

NOTE

- If the condition that results in the assertion of the flag, interrupt, or DMA request is not resolved prior to clearing the flag, the flag, and interrupt/DMA request, reasserts. For example, if the DMA or interrupt service routine fails to write sufficient data to the transmit buffer to raise it above the watermark level, the flag reasserts and generates another interrupt or DMA request.
- Reading an empty data register to clear one of the flags of the S1 register causes the FIFO pointers to become misaligned. A receive FIFO flush reinitializes the pointers. A better way to prevent this situation is to always leave one

byte in FIFO and this byte will be read eventually in clearing the flag bit.

Address: Base address + 4h offset

Bit	7	6	5	4	3	2	1	0
Read	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
Write								
Reset	1	1	0	0	0	0	0	0

UARTx_S1 field descriptions

Field	Description
7 TDRE	Transmit Data Register Empty Flag TDRE will set when the number of datawords in the transmit buffer (D and C3[T8])is equal to or less than the number indicated by TWFIFO[TXWATER]. A character that is in the process of being transmitted is not included in the count. To clear TDRE, read S1 when TDRE is set and then write to the UART data register (D). For more efficient interrupt servicing, all data except the final value to be written to the buffer must be written to D/C3[T8]. Then S1 can be read before writing the final data value, resulting in the clearing of the TRDE flag. This is more efficient because the TDRE reasserts until the watermark has been exceeded. So, attempting to clear the TDRE with every write will be ineffective until sufficient data has been written. 0 The amount of data in the transmit buffer is greater than the value indicated by TWFIFO[TXWATER]. 1 The amount of data in the transmit buffer is less than or equal to the value indicated by TWFIFO[TXWATER] at some point in time since the flag has been cleared.
6 TC	Transmit 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 reading S1 with TC set and then doing one of the following: When C7816[ISO_7816E] is set/enabled, this field is set after any NACK signal has been received, but prior to any corresponding guard times expiring. • Writing to D to transmit new data. • Queuing a preamble by clearing and then setting C2[TE]. • Queuing a break character by writing 1 to SBK in C2. 0 Transmitter active (sending data, a preamble, or a break). 1 Transmitter idle (transmission activity complete).
5 RDRF	Receive Data Register Full Flag RDRF is set when the number of datawords in the receive buffer is equal to or more than the number indicated by RWFIFO[RXWATER]. A dataword that is in the process of being received is not included in the count. To clear RDRF, read S1 when RDRF is set and then read D. For more efficient interrupt and DMA operation, read all data except the final value from the buffer, using D/C3[T8]/ED. Then read S1 and the final data value, resulting in the clearing of the RDRF flag. Even if RDRF is set, data will continue to be received until an overrun condition occurs.RDRF is prevented from setting while S2[LBKDE] is set. Additionally, when S2[LBKDE] is set, the received datawords are stored in the receive buffer but over-write each other. 0 The number of datawords in the receive buffer is less than the number indicated by RXWATER. 1 The number of datawords in the receive buffer is equal to or greater than the number indicated by RXWATER at some point in time since this flag was last cleared.
4 IDLE	Idle Line Flag

Table continues on the next page...

UARTx_S1 field descriptions (continued)

Field	Description
	After the IDLE flag is cleared, a frame must be received (although not necessarily stored in the data buffer, for example if C2[RWU] is set), or a LIN break character must set the S2[LBKDIF] flag before an idle condition can set the IDLE flag. To clear IDLE, read UART status S1 with IDLE set and then read D. IDLE is set when either of the following appear on the receiver input: • 10 consecutive logic 1s if C1[M] = 0 • 11 consecutive logic 1s if C1[M] = 1 and C4[M10] = 0 • 12 consecutive logic 1s if C1[M] = 1, C4[M10] = 1, and C1[PE] = 1 Idle detection is not supported when7816Eis set/enabled and hence this flag is ignored. **NOTE:** When RWU is set and WAKE is cleared, an idle line condition sets the IDLE flag if RWUID is set, else the IDLE flag does not become set. 0 Receiver input is either active now or has never become active since the IDLE flag was last cleared. 1 Receiver input has become idle or the flag has not been cleared since it last asserted.
3 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 UART data registers is not affected. If the OR flag is set, no data is stored in the data buffer even if sufficient room exists. Additionally, while the OR flag is set, the RDRF and IDLE flags are blocked from asserting, that is, transition from an inactive to an active state. To clear OR, read S1 when OR is set and then read D. See functional description for more details regarding the operation of the OR bit.If LBKDE is enabled and a LIN Break is detected, the OR field asserts if S2[LBKDIF] is not cleared before the next data character is received. In 7816 mode, it is possible to configure a NACK to be returned by programing C7816[ONACK]. 0 No overrun has occurred since the last time the flag was cleared. 1 Overrun has occurred or the overrun flag has not been cleared since the last overrun occured.
2 NF	Noise Flag NF is set when the UART detects noise on the receiver input. NF does not become set in the case of an overrun or while the LIN break detect feature is enabled (S2[LBKDE] = 1). When NF is set, it indicates only that a dataword has been received with noise since the last time it was cleared. There is no guarantee that the first dataword read from the receive buffer has noise or that there is only one dataword in the buffer that was received with noise unless the receive buffer has a depth of one. To clear NF, read S1 and then read D. 0 No noise detected since the last time this flag was cleared. If the receive buffer has a depth greater than 1 then there may be data in the receiver buffer that was received with noise. 1 At least one dataword was received with noise detected since the last time the flag was cleared.
1 FE	Framing Error Flag FE is set when a logic 0 is accepted as the stop bit. FE does not set in the case of an overrun or while the LIN break detect feature is enabled (S2[LBKDE] = 1). FE inhibits further data reception until it is cleared. To clear FE, read S1 with FE set and then read D. The last data in the receive buffer represents the data that was received with the frame error enabled. Framing errors are not supported when 7816E is set/ enabled. However, if this flag is set, data is still not received in 7816 mode. 0 No framing error detected. 1 Framing error.
0 PF	Parity Error Flag

Table continues on the next page...

UARTx_S1 field descriptions (continued)

Field	Description
	PF is set when PE is set and the parity of the received data does not match its parity bit. The PF is not set in the case of an overrun condition. When PF is set, it indicates only that a dataword was received with parity error since the last time it was cleared. There is no guarantee that the first dataword read from the receive buffer has a parity error or that there is only one dataword in the buffer that was received with a parity error, unless the receive buffer has a depth of one. To clear PF, read S1 and then read D., S2[LBKDE] is disabled,Within the receive buffer structure the received dataword is tagged if it is received with a parity error. This information is available by reading the ED register prior to reading the D register. 0 No parity error detected since the last time this flag was cleared. If the receive buffer has a depth greater than 1, then there may be data in the receive buffer what was received with a parity error. 1 At least one dataword was received with a parity error since the last time this flag was cleared.

36.3.6 UART Status Register 2 (UARTx_S2)

The S2 register provides inputs to the MCU for generation of UART interrupts or DMA requests. Also, this register can be polled by the MCU to check the status of these bits. This register can be read or written at any time, with the exception of the MSBF and RXINV bits, which should be changed by the user only between transmit and receive packets.

Address: Base address + 5h offset

Bit	7	6	5	4	3	2	1	0
Read	LBKDIF	RXEDGIF	MSBF	RXINV	RWUID	BRK13	LBKDE	RAF
Write								
Reset	0	0	0	0	0	0	0	0

UARTx_S2 field descriptions

Field	Description
7 LBKDIF	LIN Break Detect Interrupt Flag LBKDIF is set when LBKDE is set and a LIN break character is detected on the receiver input. The LIN break characters are 11 consecutive logic 0s if C1[M] = 0 or 12 consecutive logic 0s if C1[M] = 1. LBKDIF is set after receiving the last LIN break character. LBKDIF is cleared by writing a 1 to it. 0 No LIN break character detected. 1 LIN break character detected.
6 RXEDGIF	RxD Pin Active Edge Interrupt Flag RXEDGIF is set when an active edge occurs on the RxD pin. The active edge is falling if RXINV = 0, and rising if RXINV=1. RXEDGIF is cleared by writing a 1 to it. See for additional details. RXEDGIF description **NOTE:** The active edge is detected only in two wire mode and on receiving data coming from the RxD pin. 0 No active edge on the receive pin has occurred. 1 An active edge on the receive pin has occurred.

Table continues on the next page...

UARTx_S2 field descriptions (continued)

Field	Description
5 MSBF	Most Significant Bit First Setting this field reverses the order of the bits that are transmitted and received on the wire. This field does not affect the polarity of the bits, the location of the parity bit, or the location of the start or stop bits. This field is automatically set when C7816[INIT] and C7816[ISO7816E] are enabled and an initial character is detected in T = 0 protocol mode. 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 (bit8, bit7 or bit6) is the first bit that is transmitted following the start bit, depending on the setting of C1[M] and C1[PE]. Further, the first bit received after the start bit is identified as bit8, bit7, or bit6, depending on the setting of C1[M] and C1[PE].
4 RXINV	Receive Data Inversion Setting this field reverses the polarity of the received data input. In NRZ format, a one is represented by a mark and a zero is represented by a space for normal polarity, and the opposite for inverted polarity. This field is automatically set when C7816[INIT] and C7816[ISO7816E] are enabled and an initial character is detected in T = 0 protocol mode. **NOTE:** Setting RXINV inverts the RxD input for data bits, start and stop bits, break, and idle. When C7816[ISO7816E] is set/enabled, only the data bits and the parity bit are inverted. 0 Receive data is not inverted. 1 Receive data is inverted.
3 RWUID	Receive Wakeup Idle Detect When RWU is set and WAKE is cleared, this field controls whether the idle character that wakes the receiver sets S1[IDLE]. This field must be cleared when C7816[ISO7816E] is set/enabled. 0 S1[IDLE] is not set upon detection of an idle character. 1 S1[IDLE] is set upon detection of an idle character.
2 BRK13	Break Transmit Character Length Determines whether the transmit break character is 10, 11, or 12 bits long, or 13 or 14 bits long. See for the length of the break character for the different configurations. The detection of a framing error is not affected by this field. Transmitting break characters 0 Break character is 10, 11, or 12 bits long. 1 Break character is 13 or 14 bits long.
1 LBKDE	LIN Break Detection Enable Selects a longer break character detection length. While LBKDE is set, S1[RDRF], S1[NF], S1[FE], and S1[PF] are prevented from setting. When LBKDE is set, see . Overrun operationLBKDE must be cleared when C7816[ISO7816E] is set. 0 Break character is detected at one of the following lengths: • 10 bit times if C1[M] = 0 • 11 bit times if C1[M] = 1 and C4[M10] = 0 • 12 bit times if C1[M] = 1, C4[M10] = 1, and S1[PE] = 1 1 Break character is detected at length of 11 bit times if C1[M] = 0 or 12 bits time if C1[M] = 1.
0 RAF	Receiver Active Flag

Table continues on the next page...

UARTx_S2 field descriptions (continued)

Field	Description
	RAF is set when the UART receiver detects a logic 0 during the RT1 time period of the start bit search. RAF is cleared when the receiver detects an idle character when C7816[ISO7816E] is cleared/disabled. When C7816[ISO7816E] is enabled, the RAF is cleared if the C7816[TTYPE] = 0 expires or the C7816[TTYPE] = 1 expires. **NOTE:** In case C7816[ISO7816E] is set and C7816[TTYPE] = 0, it is possible to configure the guard time to 12. However, if a NACK is required to be transmitted, the data transfer actually takes 13 ETU with the 13th ETU slot being a inactive buffer. Therefore, in this situation, the RAF may deassert one ETU prior to actually being inactive. 0 UART receiver idle/inactive waiting for a start bit. 1 UART receiver active, RxD input not idle.

36.3.7 UART Control Register 3 (UARTx_C3)

Writing R8 does not have any effect. TXDIR and TXINV can be changed only between transmit and receive packets.

Address: Base address + 6h offset

Bit	7	6	5	4	3	2	1	0
Read	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
Write								
Reset	0	0	0	0	0	0	0	0

UARTx_C3 field descriptions

Field	Description
7 R8	Received Bit 8 R8 is the ninth data bit received when the UART is configured for 9-bit data format, that is, if C1[M] = 1 or C4[M10] = 1. The R8 value corresponds to the current data value in the UARTx_D register. To read the 9th bit, read the value of UARTx_C3[R8], then read the UARTx_D register.
6 T8	Transmit Bit 8 T8 is the ninth data bit transmitted when the UART is configured for 9-bit data format, that is, if C1[M] = 1 or C4[M10] = 1. **NOTE:** If the value of T8 is the same as in the previous transmission, T8 does not have to be rewritten. The same value is transmitted until T8 is rewritten. To correctly transmit the 9th bit, write UARTx_C3[T8] to the desired value, then write the UARTx_D register with the remaining data.
5 TXDIR	Transmitter Pin Data Direction in Single-Wire mode Determines whether the TXD pin is used as an input or output in the single-wire mode of operation. This field is relevant only to the single wire mode. When C7816[ISO7816E] is set/enabled and C7816[TTYPE] = 1, this field is automatically cleared after the requested block is transmitted. This condition is detected when TL7816[TLEN] = 0 and 4 additional characters are transmitted. Additionally, if C7816[ISO7816E] is set/enabled and C7816[TTYPE] = 0 and a NACK is being transmitted, the hardware automatically

Table continues on the next page...

UARTx_C3 field descriptions (continued)

Field	Description
	overrides this field as needed. In this situation, TXDIR does not reflect the temporary state associated with the NACK. 0 TXD pin is an input in single wire mode. 1 TXD pin is an output in single wire mode.
4 TXINV	Transmit Data Inversion. Setting this field reverses the polarity of the transmitted data output. In NRZ format, a one is represented by a mark and a zero is represented by a space for normal polarity, and the opposite for inverted polarity. This field is automatically set when C7816[INIT] and C7816[ISO7816E] are enabled and an initial character is detected in T = 0 protocol mode. **NOTE:** Setting TXINV inverts all transmitted values, including idle, break, start, and stop bits. In loop mode, if TXINV is set, the receiver gets the transmit inversion bit when RXINV is disabled. When C7816[ISO7816E] is set/enabled then only the transmitted data bits and parity bit are inverted. 0 Transmit data is not inverted. 1 Transmit data is inverted.
3 ORIE	Overrun Error Interrupt Enable Enables the overrun error flag, S1[OR], to generate interrupt requests. 0 OR interrupts are disabled. 1 OR interrupt requests are enabled.
2 NEIE	Noise Error Interrupt Enable Enables the noise flag, S1[NF], to generate interrupt requests. 0 NF interrupt requests are disabled. 1 NF interrupt requests are enabled.
1 FEIE	Framing Error Interrupt Enable Enables the framing error flag, S1[FE], to generate interrupt requests. 0 FE interrupt requests are disabled. 1 FE interrupt requests are enabled.
0 PEIE	Parity Error Interrupt Enable Enables the parity error flag, S1[PF], to generate interrupt requests. 0 PF interrupt requests are disabled. 1 PF interrupt requests are enabled.

36.3.8 UART Data Register (UARTx_D)

This register is actually two separate registers. Reads return the contents of the read-only receive data register and writes go to the write-only transmit data register.

NOTE

- In 8-bit or 9-bit data format, only UART data register (D) needs to be accessed to clear the S1[RDRF] bit (assuming receiver buffer level is less than RWFIFO[RXWATER]). The C3 register needs to be read, prior to the D register, only if the ninth bit of data needs to be captured. Similarly, the ED register needs to be read, prior to the D register, only if the additional flag data for the dataword needs to be captured.
- In the normal 8-bit mode (M bit cleared) if the parity is enabled, you get seven data bits and one parity bit. That one parity bit is loaded into the D register. So, for the data bits, mask off the parity bit from the value you read out of this register.
- When transmitting in 9-bit data format and using 8-bit write instructions, write first to transmit bit 8 in UART control register 3 (C3[T8]), then D. A write to C3[T8] stores the data in a temporary register. If D register is written first, and then the new data on data bus is stored in D, the temporary value written by the last write to C3[T8] gets stored in the C3[T8] register.

Address: Base address + 7h offset

Bit	7	6	5	4	3	2	1	0
Read Write	RT							
Reset	0	0	0	0	0	0	0	0

UARTx_D field descriptions

Field	Description
7–0 RT	Reads return the contents of the read-only receive data register and writes go to the write-only transmit data register.

36.3.9 UART Match Address Registers 1 (UARTx_MA1)

The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated C4[MAEN] field is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded. These registers can be read and written at anytime.

Address: Base address + 8h offset

Bit	7	6	5	4	3	2	1	0
Read Write	MA							
Reset	0	0	0	0	0	0	0	0

UARTx_MA1 field descriptions

Field	Description
7–0 MA	Match Address

36.3.10 UART Match Address Registers 2 (UARTx_MA2)

These registers can be read and written at anytime. The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated C4[MAEN] field is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded.

Address: Base address + 9h offset

Bit	7	6	5	4	3	2	1	0
Read Write	MA							
Reset	0	0	0	0	0	0	0	0

UARTx_MA2 field descriptions

Field	Description
7–0 MA	Match Address

36.3.11 UART Control Register 4 (UARTx_C4)

Address: Base address + Ah offset

Bit	7	6	5	4	3	2	1	0
Read Write	MAEN1	MAEN2	M10	BRFA				
Reset	0	0	0	0	0	0	0	0

UARTx_C4 field descriptions

Field	Description
7 MAEN1	Match Address Mode Enable 1 See Match address operation for more information. 0 All data received is transferred to the data buffer if MAEN2 is cleared. 1 All data received with the most significant bit cleared, is discarded. All data received with the most significant bit set, is compared with contents of MA1 register. If no match occurs, the data is discarded. If match occurs, data is transferred to the data buffer. This field must be cleared when C7816[ISO7816E] is set/enabled.
6 MAEN2	Match Address Mode Enable 2 See Match address operation for more information. 0 All data received is transferred to the data buffer if MAEN1 is cleared. 1 All data received with the most significant bit cleared, is discarded. All data received with the most significant bit set, is compared with contents of MA2 register. If no match occurs, the data is discarded. If a match occurs, data is transferred to the data buffer. This field must be cleared when C7816[ISO7816E] is set/enabled.
5 M10	10-bit Mode select Causes a tenth, non-memory mapped bit to be part of the serial transmission. This tenth bit is generated and interpreted as a parity bit. The M10 field does not affect the LIN send or detect break behavior. If M10 is set, then both C1[M] and C1[PE] must also be set. This field must be cleared when C7816[ISO7816E] is set/enabled. See Data format (non ISO-7816) for more information. 0 The parity bit is the ninth bit in the serial transmission. 1 The parity bit is the tenth bit in the serial transmission.
4–0 BRFA	Baud Rate Fine Adjust This bit field is used to add more timing resolution to the average baud frequency, in increments of 1/32. See Baud rate generation for more information.

36.3.12 UART Control Register 5 (UARTx_C5)

Address: Base address + Bh offset

Bit	7	6	5	4	3	2	1	0
Read	TDMAS	TCDMAS	RDMAS	0				
Write								
Reset	0	0	0	0	0	0	0	0

UARTx_C5 field descriptions

Field	Description
7 TDMAS	Transmitter DMA Select Configures the transmit data register empty flag, S1[TDRE], to generate interrupt or DMA requests if C2[TIE] is set.

Table continues on the next page...

UARTx_C5 field descriptions (continued)

Field	Description
	NOTE: • If C2[TIE] is cleared, TDRE DMA and TDRE interrupt request signals are not asserted when the TDRE flag is set, regardless of the state of TDMAS. • If C2[TIE] and TDMAS are both set, then C2[TCIE] must be cleared, and D must not be written unless a DMA request is being serviced. 0 If C2[TIE] is set and the S1[TDRE] flag is set, the TDRE interrupt request signal is asserted to request interrupt service. 1 If C2[TIE] is set and the S1[TDRE] flag is set, the TDRE DMA request signal is asserted to request a DMA transfer.
6 TCDMAS	Transmission Complete DMA Select Configures the transmission complete flag, S1[TC], to generate interrupt or DMA requests if C2[TCIE] is set. **NOTE:** • If C2[TCIE] is cleared, the TC DMA and TC interrupt request signals are not asserted when the S1[TC] flag is set, regardless of the state of TCDMAS. • If C2[TCIE] and TCDMAS are both set, then C2[TIE] must be cleared, and D must not be written unless a DMA request is being serviced. 0 If C2[TCIE] is set and the S1[TC] flag is set, the TC interrupt request signal is asserted to request an interrupt service. 1 If C2[TCIE] is set and the S1[TC] flag is set, the TC DMA request signal is asserted to request a DMA transfer.
5 RDMAS	Receiver Full DMA Select Configures the receiver data register full flag, S1[RDRF], to generate interrupt or DMA requests if C2[RIE] is set. **NOTE:** If C2[RIE] is cleared, and S1[RDRF] is set, the RDRF DMA and RDFR interrupt request signals are not asserted, regardless of the state of RDMAS. 0 If C2[RIE] and S1[RDRF] are set, the RDFR interrupt request signal is asserted to request an interrupt service. 1 If C2[RIE] and S1[RDRF] are set, the RDRF DMA request signal is asserted to request a DMA transfer.
4–0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

36.3.13 UART Extended Data Register (UARTx_ED)

This register contains additional information flags that are stored with a received dataword. This register may be read at any time but contains valid data only if there is a dataword in the receive FIFO.

NOTE

- The data contained in this register represents additional information regarding the conditions on which a dataword was received. The importance of this data varies with the application, and in some cases maybe completely optional.

These fields automatically update to reflect the conditions of the next dataword whenever D is read.

- If S1[NF] and S1[PF] have not been set since the last time the receive buffer was empty, the NOISY and PARITYE fields will be zero.

Address: Base address + Ch offset

Bit	7	6	5	4	3	2	1	0
Read	NOISY	PARITYE	0					
Write								
Reset	0	0	0	0	0	0	0	0

UARTx_ED field descriptions

Field	Description
7 NOISY	The current received dataword contained in D and C3[R8] was received with noise. 0 The dataword was received without noise. 1 The data was received with noise.
6 PARITYE	The current received dataword contained in D and C3[R8] was received with a parity error. 0 The dataword was received without a parity error. 1 The dataword was received with a parity error.
5–0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

36.3.14 UART Modem Register (UARTx_MODEM)

The MODEM register controls options for setting the modem configuration.

NOTE

RXRTSE, TXRTSPOL, TXRTSE, and TXCTSE must all be cleared when C7816[ISO7816EN] is enabled. This will cause the RTS to deassert during ISO-7816 wait times. The ISO-7816 protocol does not use the RTS and CTS signals.

Address: Base address + Dh offset

Bit	7	6	5	4	3	2	1	0
Read	0				RXRTSE	TXRTSPOL	TXRTSE	TXCTSE
Write								
Reset	0	0	0	0	0	0	0	0

UARTx_MODEM field descriptions

Field	Description
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
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 is deasserted if the number of characters in the receiver data register (FIFO) is equal to or greater than RWFIFO[RXWATER]. RTS is asserted when the number of characters in the receiver data register (FIFO) is less than RWFIFO[RXWATER].
2 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. 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. (FIFO)(FIFO)
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.

36.3.15 UART FIFO Parameters (UARTx_PFIFO)

This register provides the ability for the programmer to turn on and off FIFO functionality. It also provides the size of the FIFO that has been implemented. This register may be read at any time. This register must be written only when C2[RE] and C2[TE] are cleared/not set and when the data buffer/FIFO is empty.

Address: Base address + 10h offset

Bit	7	6	5	4	3	2	1	0
Read	TXFE	TXFIFOSIZE			RXFE	RXFIFOSIZE		
Write								
Reset	0	*	*	*	0	*	*	*

* Notes:
- TXFIFOSIZE field: The reset value depends on whether the specific UART instance supports the FIFO and on the size of that FIFO. See the Chip Configuration details for more information on the FIFO size supported for each UART instance.
- RXFIFOSIZE field: The reset value depends on whether the specific UART instance supports the FIFO and on the size of that FIFO. See the Chip Configuration details for more information on the FIFO size supported for each UART instance.

UARTx_PFIFO field descriptions

Field	Description
7 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 C2[TE] and C2[RE] must be cleared prior to changing this field. Additionally, TXFLUSH and RXFLUSH commands must be issued immediately after changing this field. 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 Reserved.
3 RXFE	Receive FIFO Enable

Table continues on the next page...

UARTx_PFIFO field descriptions (continued)

Field	Description
	When this field is set, the built in FIFO structure for the receive buffer is enabled. The size of the FIFO structure is indicated by the RXFIFOSIZE field. If this field is not set, the receive buffer operates as a FIFO of depth one dataword regardless of the value in RXFIFOSIZE. Both C2[TE] and C2[RE] must be cleared prior to changing this field. Additionally, TXFLUSH and RXFLUSH commands must be issued immediately after changing this field. 0 Receive FIFO is not enabled. Buffer is depth 1. (Legacy support) 1 Receive FIFO is enabled. Buffer is depth indicted by RXFIFOSIZE.
2–0 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. 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 Reserved.

36.3.16 UART FIFO Control Register (UARTx_CFIFO)

This register provides the ability to program various control fields for FIFO operation. This register may be read or written at any time. Note that writing to TXFLUSH and RXFLUSH may result in data loss and requires careful action to prevent unintended/ unpredictable behavior. Therefore, it is recommended that TE and RE be cleared prior to flushing the corresponding FIFO.

Address: Base address + 11h offset

Bit	7	6	5	4	3	2	1	0
Read	0	0	0			RXOFE	TXOFE	RXUFE
Write	TXFLUSH	RXFLUSH						
Reset	0	0	0	0	0	0	0	0

UARTx_CFIFO field descriptions

Field	Description
7 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.

Table continues on the next page...

UARTx_CFIFO field descriptions (continued)

Field	Description
6 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.
5–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 RXOFE	Receive FIFO Overflow Interrupt Enable When this field is set, the RXOF flag generates an interrupt to the host. 0 RXOF flag does not generate an interrupt to the host. 1 RXOF flag generates an interrupt to the host.
1 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.
0 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.

36.3.17 UART FIFO Status Register (UARTx_SFIFO)

This register provides status information regarding the transmit and receiver buffers/ FIFOs, including interrupt information. This register may be written to or read at any time.

Address: Base address + 12h offset

Bit	7	6	5	4	3	2	1	0
Read	TXEMPT	RXEMPT	0			RXOF	TXOF	RXUF
Write								
Reset	1	1	0	0	0	0	0	0

UARTx_SFIFO field descriptions

Field	Description
7 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.

Table continues on the next page...

UARTx_SFIFO field descriptions (continued)

Field	Description
	0 Transmit buffer is not empty. 1 Transmit buffer is empty.
6 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.
5–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 RXOF	Receiver Buffer Overflow Flag Indicates that more data has been written to the receive buffer than it can hold. This field will assert regardless of the value of CFIFO[RXOFE]. However, an interrupt will be issued to the host only if CFIFO[RXOFE] is set. This flag is cleared by writing a 1. 0 No receive buffer overflow has occurred since the last time the flag was cleared. 1 At least one receive buffer overflow has occurred since the last time the flag was cleared.
1 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 CFIFO[TXOFE]. However, an interrupt will be issued to the host only if CFIFO[TXOFE] is set. This flag is cleared by writing a 1. 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.
0 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 CFIFO[RXUFE]. However, an interrupt will be issued to the host only if CFIFO[RXUFE] is set. This flag is cleared by writing a 1. 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.

36.3.18 UART FIFO Transmit Watermark (UARTx_TWFIFO)

This register provides the ability to set a programmable threshold for notification of needing additional transmit data. This register may be read at any time but must be written only when C2[TE] is not set. Changing the value of the watermark will not clear the S1[TDRE] flag.

Address: Base address + 13h offset

Bit	7	6	5	4	3	2	1	0
Read Write	TXWATER							
Reset	0	0	0	0	0	0	0	0

UARTx_TWFIFO field descriptions

Field	Description
7–0 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 via S1[TDRE] or a DMA request via C5[TDMAS] is generated as determined by C5[TDMAS] and C2[TIE]. For proper operation, the value in TXWATER must be set to be less than the size of the transmit buffer/FIFO size as indicated by PFIFO[TXFIFOSIZE] and PFIFO[TXFE].

36.3.19 UART FIFO Transmit Count (UARTx_TCFIFO)

This is a read only register that indicates how many datawords are currently in the transmit buffer/FIFO. It may be read at any time.

Address: Base address + 14h offset

Bit	7	6	5	4	3	2	1	0
Read	TXCOUNT							
Write								
Reset	0	0	0	0	0	0	0	0

UARTx_TCFIFO field descriptions

Field	Description
7–0 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 PFIFO[TXFIFOSIZE] to calculate how much room is left in the transmit FIFO/buffer.

36.3.20 UART FIFO Receive Watermark (UARTx_RWFIFO)

This register provides the ability to set a programmable threshold for notification of the need to remove data from the receiver FIFO/buffer. This register may be read at any time but must be written only when C2[RE] is not asserted. Changing the value in this register will not clear S1[RDRF].

Address: Base address + 15h offset

Bit	7	6	5	4	3	2	1	0
Read Write	RXWATER							
Reset	0	0	0	0	0	0	0	1

UARTx_RWFIFO field descriptions

Field	Description
7–0 RXWATER	Receive Watermark When the number of datawords in the receive FIFO/buffer is equal to or greater than the value in this register field, an interrupt via S1[RDRF] or a DMA request via C5[RDMAS] is generated as determined by C5[RDMAS] and C2[RIE]. For proper operation, the value in RXWATER must be set to be less than the receive FIFO/buffer size as indicated by PFIFO[RXFIFOSIZE] and PFIFO[RXFE] and must be greater than 0.

36.3.21 UART FIFO Receive Count (UARTx_RCFIFO)

This is a read only register that indicates how many datawords are currently in the receive FIFO/buffer. It may be read at any time.

Address: Base address + 16h offset

Bit	7	6	5	4	3	2	1	0
Read	RXCOUNT							
Write								
Reset	0	0	0	0	0	0	0	0

UARTx_RCFIFO field descriptions

Field	Description
7–0 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 PFIFO[RXFIFOSIZE] to calculate how much room is left in the receive FIFO/buffer.

36.3.22 UART 7816 Control Register (UARTx_C7816)

The C7816 register is the primary control register for ISO-7816 specific functionality. This register is specific to 7816 functionality and the values in this register have no effect on UART operation and should be ignored if ISO_7816E is not set/enabled. This register may be read at any time but values must be changed only when ISO_7816E is not set.

Address: Base address + 18h offset

Bit	7	6	5	4	3	2	1	0
Read	0			ONACK	ANACK	INIT	TTYPE	ISO_7816E
Write								
Reset	0	0	0	0	0	0	0	0

UARTx_C7816 field descriptions

Field	Description
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 ONACK	Generate NACK on Overflow When this field is set, the receiver automatically generates a NACK response if a receive buffer overrun occurs, as indicated by S1[OR]. In many systems, this results in the transmitter resending the packet that overflowed until the retransmit threshold for that transmitter is reached. A NACK is generated only if TTYPE=0. This field operates independently of ANACK. See . Overrun NACK considerations 0 The received data does not generate a NACK when the receipt of the data results in an overflow event. 1 If the receiver buffer overflows, a NACK is automatically sent on a received character.
3 ANACK	Generate NACK on Error When this field is set, the receiver automatically generates a NACK response if a parity error occurs or if INIT is set and an invalid initial character is detected. A NACK is generated only if TTYPE = 0. If ANACK is set, the UART attempts to retransmit the data indefinitely. To stop retransmission attempts, clear C2[TE] or ISO_7816E and do not set until S1[TC] sets C2[TE] again. 0 No NACK is automatically generated. 1 A NACK is automatically generated if a parity error is detected or if an invalid initial character is detected.
2 INIT	Detect Initial Character When this field is set, all received characters are searched for a valid initial character. If an invalid initial character is identified, and ANACK is set, a NACK is sent. All received data is discarded and error flags blocked (S1[NF], S1[OR], S1[FE], S1[PF], IS7816[WT], IS7816[CWT], IS7816[BWT], IS7816[GTV]) until a valid initial character is detected. Upon detecting a valid initial character, the configuration values S2[MSBF], C3[TXINV], and S2[RXINV] are automatically updated to reflect the initial character that was received. The actual INIT data value is not stored in the receive buffer. Additionally, upon detection of a valid initial character, IS7816[INITD] is set and an interrupt issued as programmed by IE7816[INITDE]. When a valid initial character is detected, INIT is automatically cleared. This Initial Character Detect feature is supported only in T = 0 protocol mode. 0 Normal operating mode. Receiver does not seek to identify initial character. 1 Receiver searches for initial character.
1 TTYPE	Transfer Type Indicates the transfer protocol being used. See ISO-7816 / smartcard support for more details. 0 T = 0 per the ISO-7816 specification. 1 T = 1 per the ISO-7816 specification.
0 ISO_7816E	ISO-7816 Functionality Enabled Indicates that the UART is operating according to the ISO-7816 protocol. **NOTE:** This field must be modified only when no transmit or receive is occurring. If this field is changed during a data transfer, the data being transmitted or received may be transferred incorrectly. 0 ISO-7816 functionality is turned off/not enabled. 1 ISO-7816 functionality is turned on/enabled.

36.3.23 UART 7816 Interrupt Enable Register (UARTx_IE7816)

The IE7816 register controls which flags result in an interrupt being issued. This register is specific to 7816 functionality, the corresponding flags that drive the interrupts are not asserted when 7816E is not set/enabled. However, these flags may remain set if they are asserted while 7816E was set and not subsequently cleared. This register may be read or written to at any time.

Address: Base address + 19h offset

Bit	7	6	5	4	3	2	1	0
Read Write	WTE	CWTE	BWTE	INITDE	0	GTVE	TXTE	RXTE
Reset	0	0	0	0	0	0	0	0

UARTx_IE7816 field descriptions

Field	Description
7 WTE	Wait Timer Interrupt Enable 0 The assertion of IS7816[WT] does not result in the generation of an interrupt. 1 The assertion of IS7816[WT] results in the generation of an interrupt.
6 CWTE	Character Wait Timer Interrupt Enable 0 The assertion of IS7816[CWT] does not result in the generation of an interrupt. 1 The assertion of IS7816[CWT] results in the generation of an interrupt.
5 BWTE	Block Wait Timer Interrupt Enable 0 The assertion of IS7816[BWT] does not result in the generation of an interrupt. 1 The assertion of IS7816[BWT] results in the generation of an interrupt.
4 INITDE	Initial Character Detected Interrupt Enable 0 The assertion of IS7816[INITD] does not result in the generation of an interrupt. 1 The assertion of IS7816[INITD] results in the generation of an interrupt.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 GTVE	Guard Timer Violated Interrupt Enable 0 The assertion of IS7816[GTV] does not result in the generation of an interrupt. 1 The assertion of IS7816[GTV] results in the generation of an interrupt.
1 TXTE	Transmit Threshold Exceeded Interrupt Enable 0 The assertion of IS7816[TXT] does not result in the generation of an interrupt. 1 The assertion of IS7816[TXT] results in the generation of an interrupt.
0 RXTE	Receive Threshold Exceeded Interrupt Enable 0 The assertion of IS7816[RXT] does not result in the generation of an interrupt. 1 The assertion of IS7816[RXT] results in the generation of an interrupt.

36.3.24 UART 7816 Interrupt Status Register (UARTx_IS7816)

The IS7816 register provides a mechanism to read and clear the interrupt flags. All flags/ interrupts are cleared by writing a 1 to the field location. Writing a 0 has no effect. All bits are "sticky", meaning they indicate that only the flag condition that occurred since the last time the bit was cleared, not that the condition currently exists. The status flags are set regardless of whether the corresponding field in the IC7816 is set or cleared. The IC7816 controls only if an interrupt is issued to the host processor. This register is specific to 7816 functionality and the values in this register have no affect on UART operation and should be ignored if 7816E is not set/enabled. This register may be read or written at anytime.

Address: Base address + 1Ah offset

Bit	7	6	5	4	3	2	1	0
Read	WT	CWT	BWT	INITD	0	GTV	TXT	RXT
Write								
Reset	0	0	0	0	0	0	0	0

UARTx_IS7816 field descriptions

Field	Description
7 WT	Wait Timer Interrupt Indicates that the wait time, the time between the leading edge of a character being transmitted and the leading edge of the next response character, has exceeded the programmed value. This flag asserts only when C7816[TTYPE] = 0. This interrupt is cleared by writing 1. 0 Wait time (WT) has not been violated. 1 Wait time (WT) has been violated.
6 CWT	Character Wait Timer Interrupt Indicates that the character wait time, the time between the leading edges of two consecutive characters in a block, has exceeded the programmed value. This flag asserts only when C7816[TTYPE] = 1. This interrupt is cleared by writing 1. 0 Character wait time (CWT) has not been violated. 1 Character wait time (CWT) has been violated.
5 BWT	Block Wait Timer Interrupt Indicates that the block wait time, the time between the leading edge of first received character of a block and the leading edge of the last character the previously transmitted block, has exceeded the programmed value. This flag asserts only when C7816[TTYPE] = 1.This interrupt is cleared by writing 1. 0 Block wait time (BWT) has not been violated. 1 Block wait time (BWT) has been violated.
4 INITD	Initial Character Detected Interrupt Indicates that a valid initial character is received. This interrupt is cleared by writing 1.

Table continues on the next page...

UARTx_IS7816 field descriptions (continued)

Field	Description
	0 A valid initial character has not been received. 1 A valid initial character has been received.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 GTV	Guard Timer Violated Interrupt Indicates that one or more of the character guard time, block guard time, or guard time are violated. This interrupt is cleared by writing 1. 0 A guard time (GT, CGT, or BGT) has not been violated. 1 A guard time (GT, CGT, or BGT) has been violated.
1 TXT	Transmit Threshold Exceeded Interrupt Indicates that the transmit NACK threshold has been exceeded as indicated by ET7816[TXTHRESHOLD]. Regardless of whether this flag is set, the UART continues to retransmit indefinitely. This flag asserts only when C7816[TTYPE] = 0. If 7816E is cleared/disabled, ANACK is cleared/disabled, C2[TE] is cleared/disabled, C7816[TTYPE] = 1, or packet is transferred without receiving a NACK, the internal NACK detection counter is cleared and the count restarts from zero on the next received NACK. This interrupt is cleared by writing 1. 0 The number of retries and corresponding NACKS does not exceed the value in ET7816[TXTHRESHOLD]. 1 The number of retries and corresponding NACKS exceeds the value in ET7816[TXTHRESHOLD].
0 RXT	Receive Threshold Exceeded Interrupt Indicates that there are more than ET7816[RXTHRESHOLD] consecutive NACKS generated in response to parity errors on received data. This flag requires ANACK to be set. Additionally, this flag asserts only when C7816[TTYPE] = 0. Clearing this field also resets the counter keeping track of consecutive NACKS. The UART will continue to attempt to receive data regardless of whether this flag is set. If 7816E is cleared/disabled, RE is cleared/disabled, C7816[TTYPE] = 1, or packet is received without needing to issue a NACK, the internal NACK detection counter is cleared and the count restarts from zero on the next transmitted NACK. This interrupt is cleared by writing 1. 0 The number of consecutive NACKS generated as a result of parity errors and buffer overruns is less than or equal to the value in ET7816[RXTHRESHOLD]. 1 The number of consecutive NACKS generated as a result of parity errors and buffer overruns is greater than the value in ET7816[RXTHRESHOLD].

36.3.25 UART 7816 Wait Parameter Register (UARTx_WP7816T0)

The WP7816 register contains constants used in the generation of various wait timer counters. To save register space, this register is used differently when C7816[TTYPE] = 0 and C7816[TTYPE] = 1. This register may be read at any time. This register must be written to only when C7816[ISO_7816E] is not set.

Address: Base address + 1Bh offset

Bit	7	6	5	4	3	2	1	0
Read Write	WI							
Reset	0	0	0	0	1	0	1	0

UARTx_WP7816T0 field descriptions

Field	Description
7–0 WI	Wait Timer Interrupt (C7816[TTYPE] = 0) Used to calculate the value used for the WT counter. It represents a value between 1 and 255. The value of zero is not valid. This value is used only when C7816[TTYPE] = 0. See Wait time and guard time parameters.

36.3.26 UART 7816 Wait Parameter Register (UARTx_WP7816T1)

The WP7816 register contains constants used in the generation of various wait timer counters. To save register space, this register is used differently when C7816[TTYPE] = 0 and C7816[TTYPE] = 1. This register may be read at any time. This register must be written to only when C7816[ISO_7816E] is not set.

Address: Base address + 1Bh offset

Bit	7	6	5	4	3	2	1	0
Read Write	CWI				BWI			
Reset	0	0	0	0	1	0	1	0

UARTx_WP7816T1 field descriptions

Field	Description
7–4 CWI	Character Wait Time Integer (C7816[TTYPE] = 1) Used to calculate the value used for the CWT counter. It represents a value between 0 and 15. This value is used only when C7816[TTYPE] = 1. See Wait time and guard time parameters .
3–0 BWI	Block Wait Time Integer(C7816[TTYPE] = 1) Used to calculate the value used for the BWT counter. It represent a value between 0 and 15. This value is used only when C7816[TTYPE] = 1. See Wait time and guard time parameters .

36.3.27 UART 7816 Wait N Register (UARTx_WN7816)

The WN7816 register contains a parameter that is used in the calculation of the guard time counter. This register may be read at any time. This register must be written to only when C7816[ISO_7816E] is not set.

Address: Base address + 1Ch offset

Bit	7	6	5	4	3	2	1	0
Read Write	GTN							
Reset	0	0	0	0	0	0	0	0

UARTx_WN7816 field descriptions

Field	Description
7–0 GTN	Guard Band N Defines a parameter used in the calculation of GT, CGT, and BGT counters. The value represents an integer number between 0 and 255. See Wait time and guard time parameters .

36.3.28 UART 7816 Wait FD Register (UARTx_WF7816)

The WF7816 contains parameters that are used in the generation of various counters including GT, CGT, BGT, WT, and BWT. This register may be read at any time. This register must be written to only when C7816[ISO_7816E] is not set.

Address: Base address + 1Dh offset

Bit	7	6	5	4	3	2	1	0
Read Write	GTFD							
Reset	0	0	0	0	0	0	0	1

UARTx_WF7816 field descriptions

Field	Description
7–0 GTFD	FD Multiplier Used as another multiplier in the calculation of WT and BWT. This value represents a number between 1 and 255. The value of 0 is invalid. This value is not used in baud rate generation. See Wait time and guard time parameters and Baud rate generation .

36.3.29 UART 7816 Error Threshold Register (UARTx_ET7816)

The ET7816 register contains fields that determine the number of NACKs that must be received or transmitted before the host processor is notified. This register may be read at anytime. This register must be written to only when C7816[ISO_7816E] is not set.

Address: Base address + 1Eh offset

Bit	7	6	5	4	3	2	1	0
Read Write	TXTHRESHOLD				RXTHRESHOLD			
Reset	0	0	0	0	0	0	0	0

UARTx_ET7816 field descriptions

Field	Description
7–4 TXTHRESHOLD	Transmit NACK Threshold The value written to this field indicates the maximum number of failed attempts (NACKs) a transmitted character can have before the host processor is notified. This field is meaningful only when

Table continues on the next page...

UARTx_ET7816 field descriptions (continued)

Field	Description
	C7816[TTYPE] = 0 and C7816[ANACK] = 1. The value read from this field represents the number of consecutive NACKs that have been received since the last successful transmission. This counter saturates at 4'hF and does not wrap around. Regardless of how many NACKs that are received, the UART continues to retransmit indefinitely. This flag only asserts when C7816[TTYPE] = 0. For additional information see the IS7816[TXT] field description. 0 TXT asserts on the first NACK that is received. 1 TXT asserts on the second NACK that is received.
3–0 RXTHRESHOLD	Receive NACK Threshold The value written to this field indicates the maximum number of consecutive NACKs generated as a result of a parity error or receiver buffer overruns before the host processor is notified. After the counter exceeds that value in the field, the IS7816[RXT] is asserted. This field is meaningful only when C7816[TTYPE] = 0. The value read from this field represents the number of consecutive NACKs that have been transmitted since the last successful reception. This counter saturates at 4'hF and does not wrap around. Regardless of the number of NACKs sent, the UART continues to receive valid packets indefinitely. For additional information, see IS7816[RXT] field description.

36.3.30 UART 7816 Transmit Length Register (UARTx_TL7816)

The TL7816 register is used to indicate the number of characters contained in the block being transmitted. This register is used only when C7816[TTYPE] = 1. This register may be read at anytime. This register must be written only when C2[TE] is not enabled.

Address: Base address + 1Fh offset

Bit	7	6	5	4	3	2	1	0
Read Write	TLEN							
Reset	0	0	0	0	0	0	0	0

UARTx_TL7816 field descriptions

Field	Description
7–0 TLEN	Transmit Length This value plus four indicates the number of characters contained in the block being transmitted. This register is automatically decremented by 1 for each character in the information field portion of the block. Additionally, this register is automatically decremented by 1 for the first character of a CRC in the epilogue field. Therefore, this register must be programmed with the number of bytes in the data packet if an LRC is being transmitted, and the number of bytes + 1 if a CRC is being transmitted. This register is not decremented for characters that are assumed to be part of the Prologue field, that is, the first three characters transmitted in a block, or the LRC or last CRC character in the Epilogue field, that is, the last character transmitted. This field must be programed or adjusted only when C2[TE] is cleared.

36.4 Functional description

This section provides a complete functional description of the UART block.

The UART allows full duplex, asynchronous, NRZ serial communication between the CPU and remote devices, including other CPUs. The UART transmitter and receiver operate independently, although they use the same baud rate generator. The CPU monitors the status of the UART, writes the data to be transmitted, and processes received data.

36.4.1 Transmitter

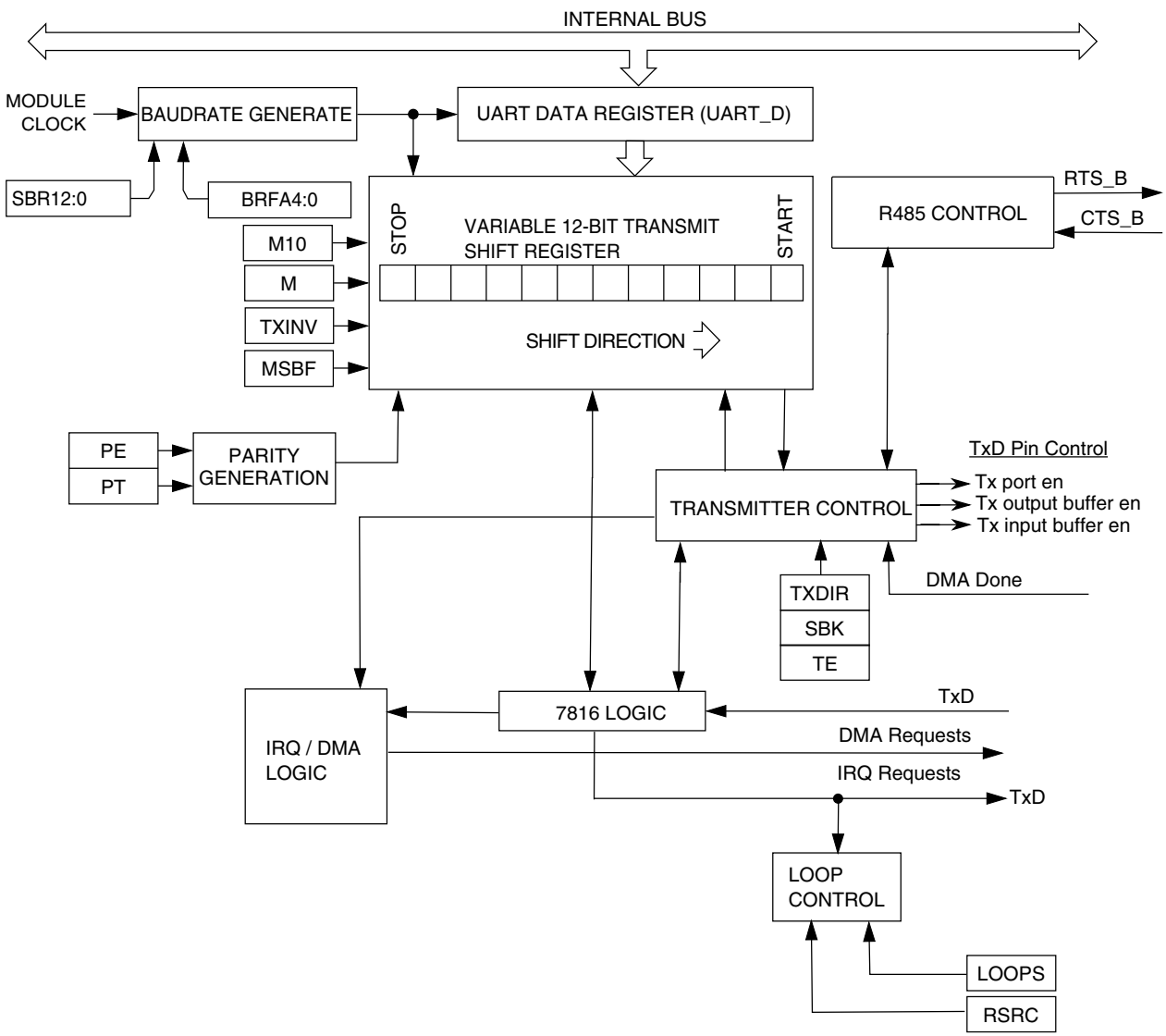


Figure 36-121. Transmitter Block Diagram

36.4.1.1 Transmitter character length

The UART transmitter can accommodate either 8, 9, or 10-bit data characters. The state of the C1[M] and C1[PE] bits and the C4[M10] bit determine the length of data characters. When transmitting 9-bit data, bit C3[T8] is the ninth bit (bit 8).

36.4.1.2 Transmission bit order

When S2[MSBF] is set, the UART automatically transmits the MSB of the data word as the first bit after the start bit. Similarly, the LSB of the data word is transmitted immediately preceding the parity bit, or the stop bit if parity is not enabled. All necessary bit ordering is handled automatically by the module. Therefore, the format of the data written to D for transmission is completely independent of the S2[MSBF] setting.

36.4.1.3 Character transmission

To transmit data, the MCU writes the data bits to the UART transmit buffer using UART data registers C3[T8] and D. Data in the transmit buffer is then transferred to the transmitter shift register as needed. The transmit shift register then shifts a frame out through the transmit data output signal after it has prefaced it with any required start and stop bits. The UART data registers, C3[T8] and D, provide access to the transmit buffer structure.

The UART also sets a flag, the transmit data register empty flag S1[TDRE], and generates an interrupt or DMA request (C5[TDMAS]) whenever the number of datawords in the transmit buffer is equal to or less than the value indicated by TWFIFO[TXWATER]. The transmit driver routine may respond to this flag by writing additional datawords to the transmit buffer using C3[T8]/D as space permits.

See Application information for specific programing sequences.

Setting C2[TE] automatically loads the transmit shift register with the following preamble:

- 10 logic 1s if C1[M] = 0
- 11 logic 1s if C1[M] = 1 and C4[M10] = 0
- 12 logic 1s if C1[M] = 1, C4[M10] = 1, C1[PE] = 1

After the preamble shifts out, control logic transfers the data from the D register into the transmit shift register. The transmitter automatically transmits the correct start bit and stop bit before and after the dataword.

When C7816[ISO_7816E] = 1, setting C2[TE] does not result in a preamble being generated. The transmitter starts transmitting as soon as the corresponding guard time expires. When C7816[TTYPE] = 0, the value in GT is used. When C7816[TTYPE] = 1, the value in BGT is used, because C2[TE] will remain asserted until the end of the block transfer. C2[TE] is automatically cleared when C7816[TTYPE] = 1 and the block being transmitted has completed. When C7816[TTYPE] = 0, the transmitter listens for a NACK indication. If no NACK is received, it is assumed that the character was correctly received. If a NACK is received, the transmitter resends the data, assuming that the number of retries for that character, that is, the number of NACKs received, is less than or equal to the value in ET7816[TXTHRESHOLD].

Hardware supports odd or even parity. When parity is enabled, the bit immediately preceding the stop bit is the parity bit.

When the transmit shift register is not transmitting a frame, the transmit data output signal goes to the idle condition, logic 1. If at any time software clears C2[TE], the transmitter enable signal goes low and the transmit signal goes idle.

If the software clears C2[TE] while a transmission is in progress, the character in the transmit shift register continues to shift out, provided S1[TC] was cleared during the data write sequence. To clear S1[TC], the S1 register must be read followed by a write to D register. If the TC DMA interface is used for data transfer, S1[TC] is cleared automatically when the DMA finishes transferring the requested data. A read of the S1 register is not required to clear it.

If S1[TC] is cleared during character transmission and C2[TE] is cleared, the transmission enable signal is deasserted at the completion of the current frame. Following this, the transmit data out signal enters the idle state even if there is data pending in the UART transmit data buffer. To ensure that all the data written in the FIFO is transmitted on the link before clearing C2[TE], wait for S1[TC] to set. Alternatively, the same can be achieved by setting TWFIFO[TXWATER] to 0x0 and waiting for S1[TDRE] to set.

36.4.1.4 Transmitting break characters

Setting C2[SBK] loads the transmit shift register with a break character. A break character contains all logic 0s and has no start, stop, or parity bit. Break character length depends on C1[M] and C1[PE], S2[BRK13], and C4[M10]. See the following table.

Table 36-127. Transmit break character length

S2[BRK13]	C1[M]	C4[M10]	C1[PE]	Bits transmitted
0	0	—	—	10
0	1	0	—	11

Table continues on the next page...

Table 36-127. Transmit break character length (continued)

S2[BRK13]	C1[M]	C4[M10]	C1[PE]	Bits transmitted
0	1	1	0	11
0	1	1	1	12
1	0	—	—	13
1	1	—	—	14

As long as C2[SBK] is set, the transmitter logic continuously loads break characters into the transmit shift register. After the software clears C2[SBK], the shift register finishes transmitting the last break character and then transmits at least one logic 1. The automatic logic 1 at the end of a break character guarantees the recognition of the start bit of the next character. Break bits are not supported when C7816[ISO_7816E] is set/enabled.

NOTE

When queuing a break character, it will be transmitted following the completion of the data value currently being shifted out from the shift register. This means that, if data is queued in the data buffer to be transmitted, the break character preempts that queued data. The queued data is then transmitted after the break character is complete.

36.4.1.5 Idle characters

An idle character contains all logic 1s and has no start, stop, or parity bit. Idle character length depends on C1[M], C1[PE], and C4[M10]. The preamble is a synchronizing idle character that begins the first transmission initiated after setting C2[TE]. When C7816[ISO_7816E] is set/enabled, idle characters are not sent or detected. When data is not being transmitted, the data I/O line is in an inactive state.

If C2[TE] is cleared during a transmission, the transmit data output signal becomes idle after completion of the transmission in progress. Clearing and then setting C2[TE] during a transmission queues an idle character to be sent after the dataword currently being transmitted.

Note

When queuing an idle character, the idle character will be transmitted following the completion of the data value currently being shifted out from the shift register. This means that if data is queued in the data buffer to be transmitted, the idle character preempts that queued data. The queued data is then transmitted after the idle character is complete.

If C2[TE] is cleared and the transmission is completed, the UART is not the master of the TXD pin.

36.4.1.6 Hardware flow control

The transmitter supports hardware flow control by gating the transmission with the value of CTS. If the clear-to-send operation is enabled, the character is transmitted when CTS is asserted. If CTS is deasserted in the middle of a transmission with characters remaining in the receiver data buffer, the character in the shift register is sent and TXD remains in the mark state until CTS is reasserted.

If the clear-to-send operation is disabled, the transmitter ignores the state of CTS. Also, if the transmitter is forced to send a continuous low condition because it is sending a break character, the transmitter ignores the state of CTS regardless of whether the clear-to-send operation is enabled.

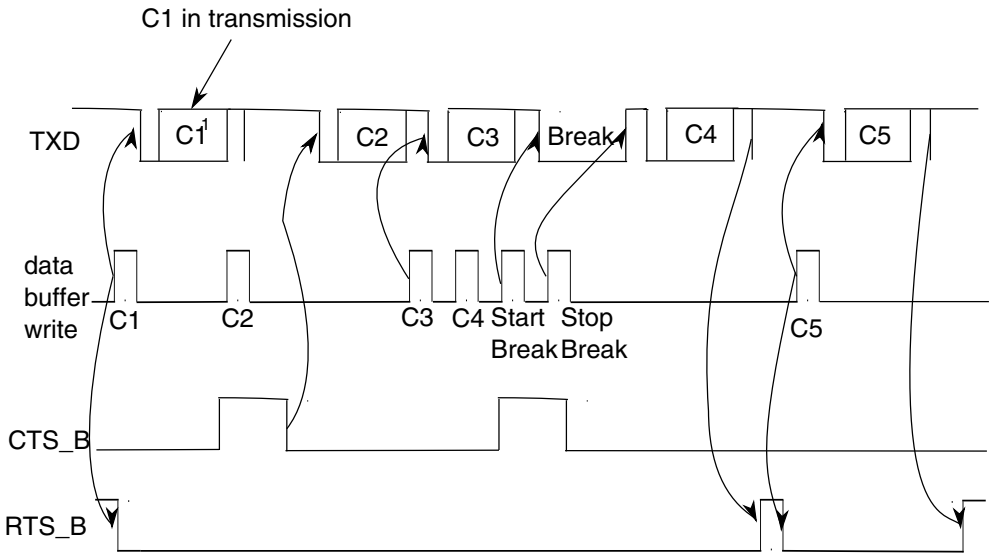
The transmitter's CTS signal can also be enabled even if the same UART receiver's RTS signal is disabled.

36.4.1.7 Transceiver driver enable

The transmitter can use RTS as an enable signal for the driver of an external transceiver. See Transceiver driver enable using RTS for details. If the request-to-send operation is enabled, when a character is placed into an empty transmitter data buffer, RTS asserts one bit time before the start bit is transmitted. RTS remains asserted for the whole time that the transmitter data buffer has any characters. RTS deasserts one bit time after all characters in the transmitter data buffer and shift register are completely sent, including the last stop bit. Transmitting a break character also asserts RTS, with the same assertion and deassertion timing as having a character in the transmitter data buffer.

The transmitter's RTS signal asserts only when the transmitter is enabled. However, the transmitter's RTS signal is unaffected by its CTS signal. RTS will remain asserted until the transfer is completed, even if the transmitter is disabled mid-way through a data transfer.

The following figure shows the functional timing information for the transmitter. Along with the actual character itself, TXD shows the start bit. The stop bit is also indicated, with a dashed line if necessary.



1. Cn = transmit characters

Figure 36-122. Transmitter RTS and CTS timing diagram

36.4.2 Receiver

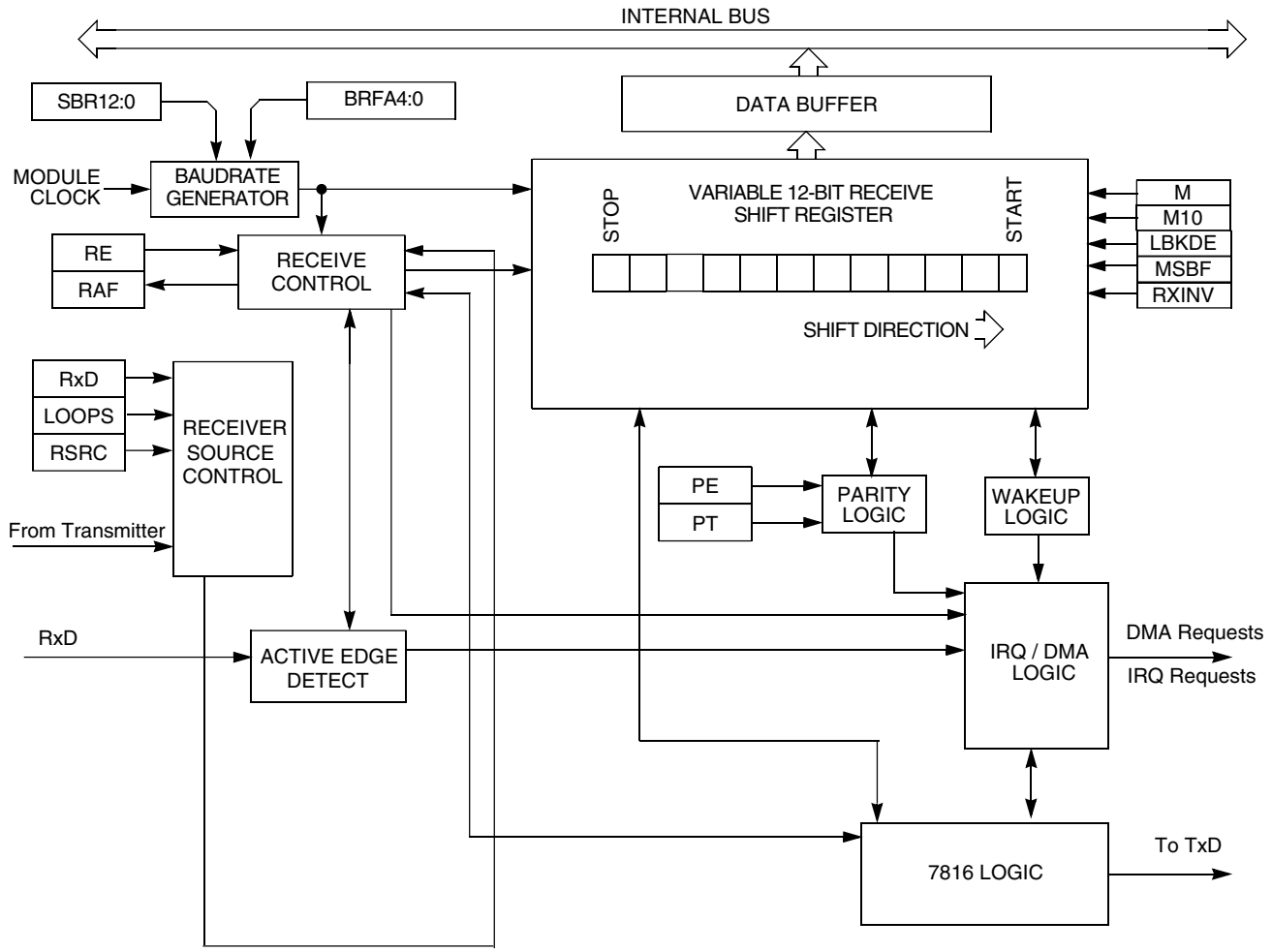


Figure 36-123. UART receiver block diagram

36.4.2.1 Receiver character length

The UART receiver can accommodate 8-, 9-, or 10-bit data characters. The states of C1[M], C1[PE], and C4[M10] determine the length of data characters. When receiving 9 or 10-bit data, C3[R8] is the ninth bit (bit 8).

36.4.2.2 Receiver bit ordering

When S2[MSBF] is set, the receiver operates such that the first bit received after the start bit is the MSB of the dataword. Similarly, the bit received immediately preceding the parity bit, or the stop bit if parity is not enabled, is treated as the LSB for the dataword. All necessary bit ordering is handled automatically by the module. Therefore, the format of the data read from receive data buffer is completely independent of S2[MSBF].

36.4.2.3 Character reception

During UART reception, the receive shift register shifts a frame in from the unsynchronized receiver input signal. After a complete frame shifts into the receive shift register, the data portion of the frame transfers to the UART receive buffer. Additionally, the noise and parity error flags that are calculated during the receive process are also captured in the UART receive buffer. The receive data buffer is accessible via the D and C3[T8] registers. Additional received information flags regarding the receive dataword can be read in ED register. S1[RDRF] is set if the number of resulting datawords in the receive buffer is equal to or greater than the number indicated by RWFIFO[RXWATER]. If the C2[RIE] is also set, RDRF generates an RDRF interrupt request. Alternatively, by programming C5[RDMAS], a DMA request can be generated.

When C7816[ISO_7816E] is set/enabled and C7816[TTYPE] = 0, character reception operates slightly differently. Upon receipt of the parity bit, the validity of the parity bit is checked. If C7816[ANACK] is set and the parity check fails, or if INIT and the received character is not a valid initial character, then a NACK is sent by the receiver. If the number of consecutive receive errors exceeds the threshold set by ET7816[RXTHRESHOLD], then IS7816[RXT] is set and an interrupt generated if IE7816[RXTE] is set. If an error is detected due to parity or an invalid initial character, the data is not transferred from the receive shift register to the receive buffer. Instead, the data is overwritten by the next incoming data.

When the C7816[ISO_7816E] is set/enabled, C7816[ONACK] is set/enabled, and the received character results in the receive buffer overflowing, a NACK is issued by the receiver. Additionally, S1[OR] is set and an interrupt is issued if required, and the data in the shift register is discarded.

36.4.2.4 Data sampling

The receiver samples the unsynchronized receiver input signal at the RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch, the RT clock (see the following figure) is re-synchronized:

- After every start bit.
- After the receiver detects a data bit change from logic 1 to logic 0 (after the majority of data bit samples at RT8, RT9, and RT10 returns a valid logic 1 and the majority of the next RT8, RT9, and RT10 samples returns a valid logic 0).

To locate the start bit, data recovery logic does an asynchronous search for a logic 0 preceded by three logic 1s. When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.

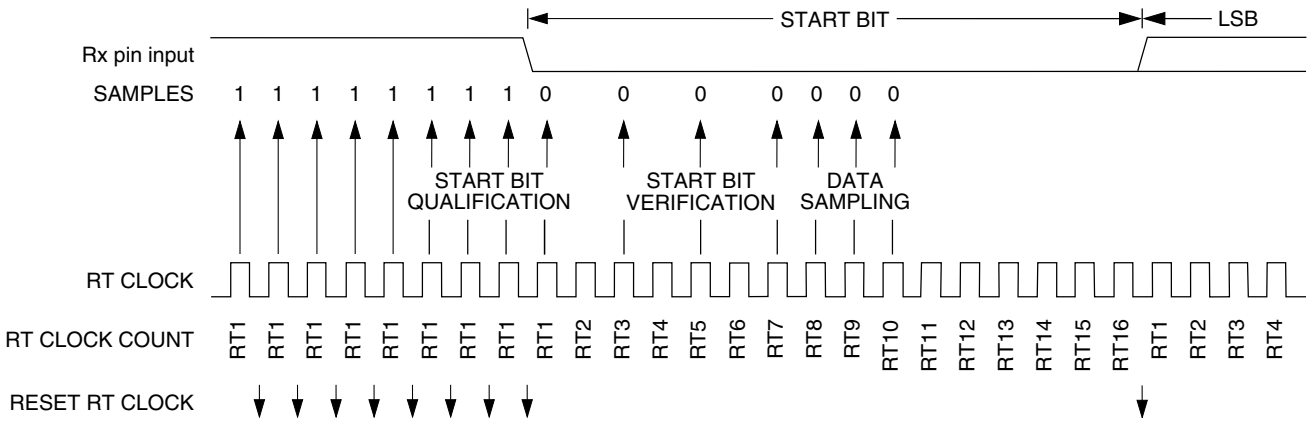


Figure 36-124. Receiver data sampling

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7 when C7816[ISO_7816E] is cleared/disabled and RT8, RT9 and RT10 when C7816[ISO_7816E] is set/enabled. The following table summarizes the results of the start bit verification samples.

Table 36-128. Start bit verification

RT3, RT5, and RT7 samples RT8, RT9, RT10 samples when 7816E	Start bit verification	Noise flag
000	Yes	0
001	Yes	1
010	Yes	1
011	No	0
100	Yes	1
101	No	0
110	No	0
111	No	0

If start bit verification is not successful, the RT clock is reset and a new search for a start bit begins.

To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. The following table summarizes the results of the data bit samples.

Table 36-129. Data bit recovery

RT8, RT9, and RT10 samples	Data bit determination	Noise flag
000	0	0
001	0	1
010	0	1
011	1	1
100	0	1
101	1	1
110	1	1
111	1	0

Note

The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are logic 1s following a successful start bit verification, the noise flag (S1[NF]) is set and the receiver assumes that the bit is a start bit (logic 0). With the exception of when C7816[ISO_7816E] is set/enabled, where the values of RT8, RT9 and RT10 exclusively determine if a start bit exists.

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. The following table summarizes the results of the stop bit samples. In the event that C7816[ISO_7816E] is set/enabled and C7816[TTYPE] = 0, verification of a stop bit does not take place. Rather, starting with RT8 the receiver transmits a NACK as programmed until time RT9 of the following time period. Framing Error detection is not supported when C7816[ISO_7816E] is set/enabled.

Table 36-130. Stop bit recovery

RT8, RT9, and RT10 samples	Framing error flag	Noise flag
000	1	0
001	1	1
010	1	1
011	0	1
100	1	1
101	0	1
110	0	1
111	0	0

In the following figure, the verification samples RT3 and RT5 determine that the first low detected was noise and not the beginning of a start bit. In this example C7816[ISO_7816E] = 0. The RT clock is reset and the start bit search begins again. The noise flag is not set because the noise occurred before the start bit was found.

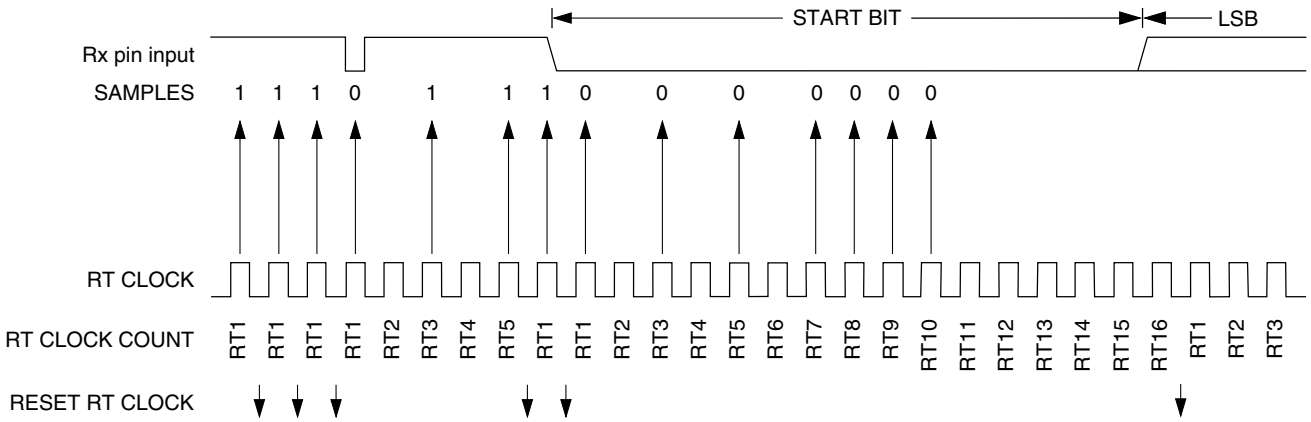


Figure 36-125. Start bit search example 1 (C7816[ISO_7816E] = 0)

In the following figure, verification sample at RT3 is high. In this example C7816[ISO_7816E] = 0. The RT3 sample sets the noise flag. Although the perceived bit time is misaligned, the data samples RT8, RT9, and RT10 are within the bit time and data recovery is successful.

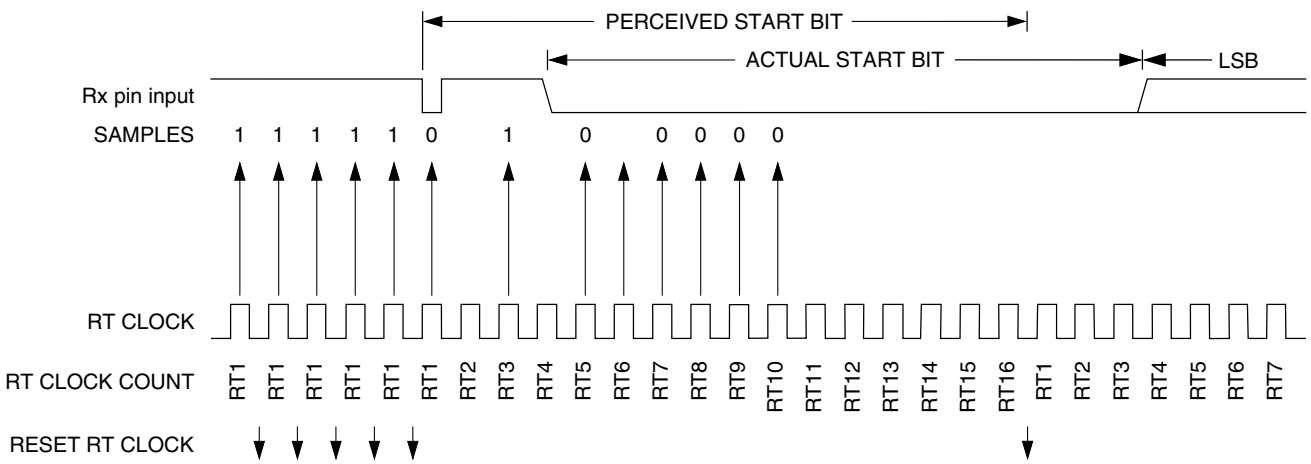


Figure 36-126. Start bit search example 2 (C7816[ISO_7816E] = 0)

In the following figure, a large burst of noise is perceived as the beginning of a start bit, although the test sample at RT5 is high. In this example C7816[ISO_7816E] = 0. The RT5 sample sets the noise flag. Although this is a worst-case misalignment of perceived bit time, the data samples RT8, RT9, and RT10 are within the bit time and data recovery is successful.

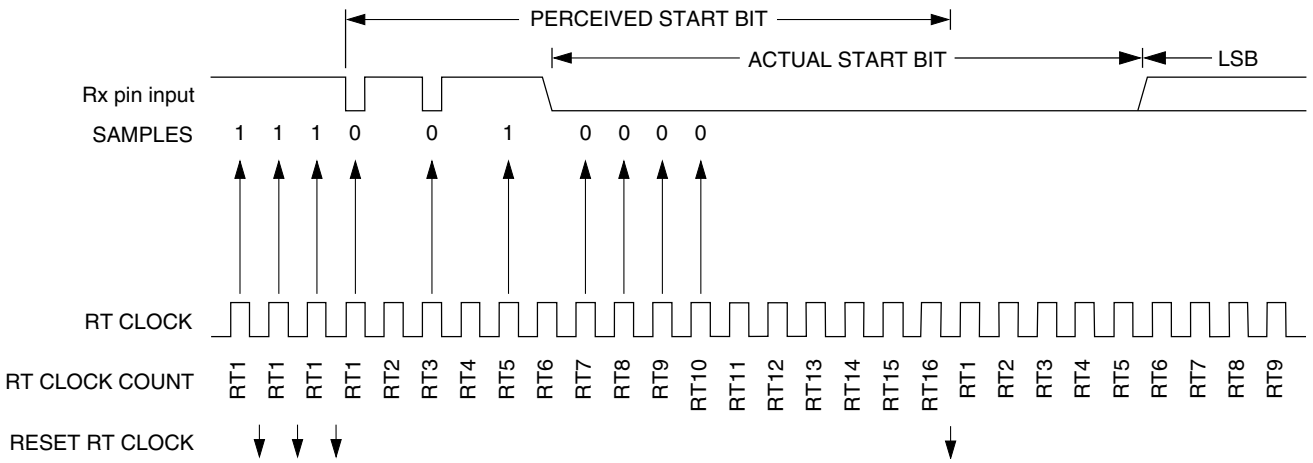


Figure 36-127. Start bit search example 3 (C7816[ISO_7816E] = 0)

The following figure shows the effect of noise early in the start bit time. In this example C7816[ISO_7816E] = 0. Although this noise does not affect proper synchronization with the start bit time, it does set the noise flag.

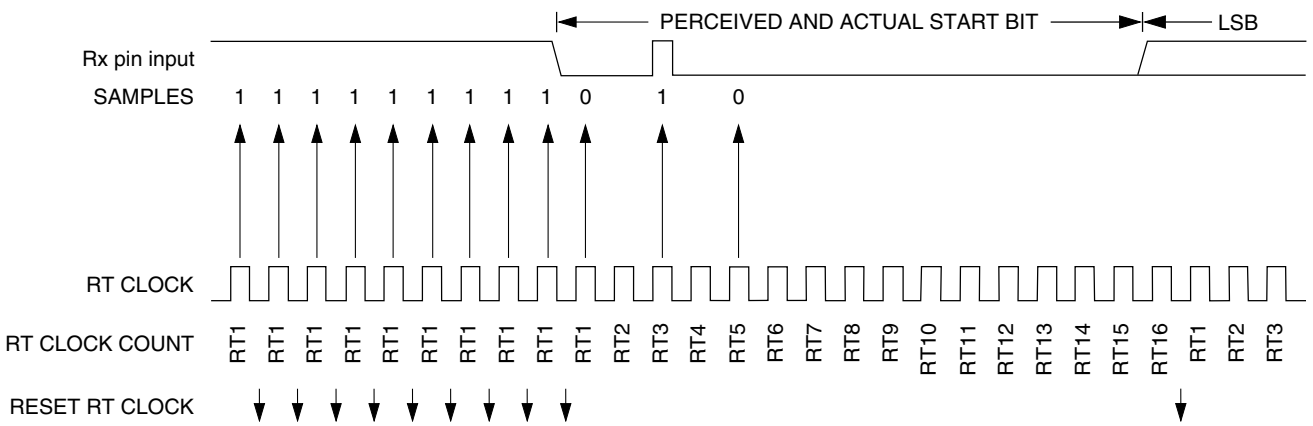


Figure 36-128. Start bit search example 4 (C7816[ISO_7816E] = 0)

The following figure shows a burst of noise near the beginning of the start bit that resets the RT clock. In this example C7816[ISO_7816E] = 0. The sample after the reset is low but is not preceded by three high samples that would qualify as a falling edge. Depending on the timing of the start bit search and on the data, the frame may be missed entirely or it may set the framing error flag.

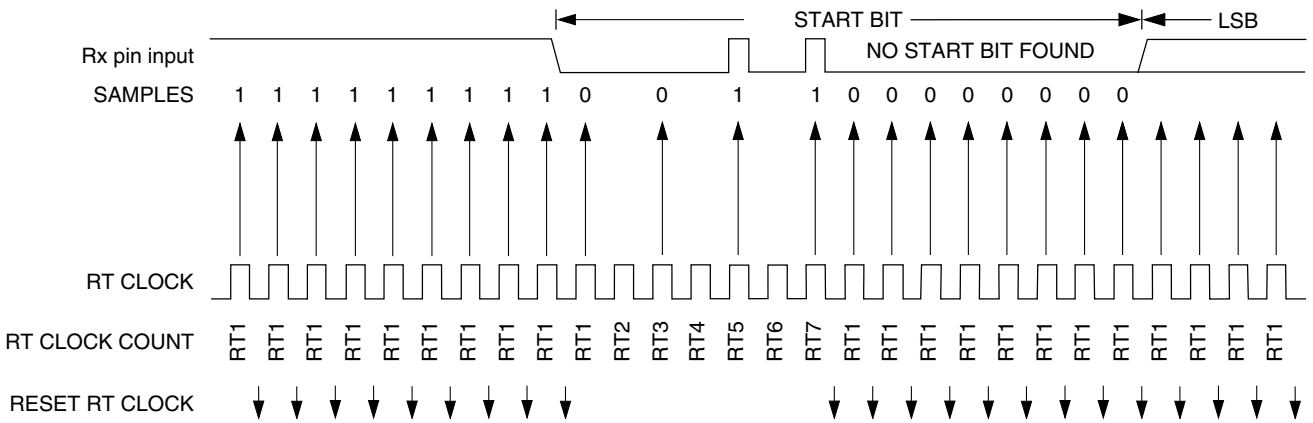


Figure 36-129. Start bit search example 5 (C7816[ISO_7816E] = 0)

In the following figure, a noise burst makes the majority of data samples RT8, RT9, and RT10 high. In this example C7816[ISO_7816E] = 0. This sets the noise flag but does not reset the RT clock. In start bits only, the RT8, RT9, and RT10 data samples are ignored. In this example, if C7816[ISO_7816E] = 1 then a start bit would not have been detected at all since at least two of the three samples (RT8, RT9, RT10) were high.

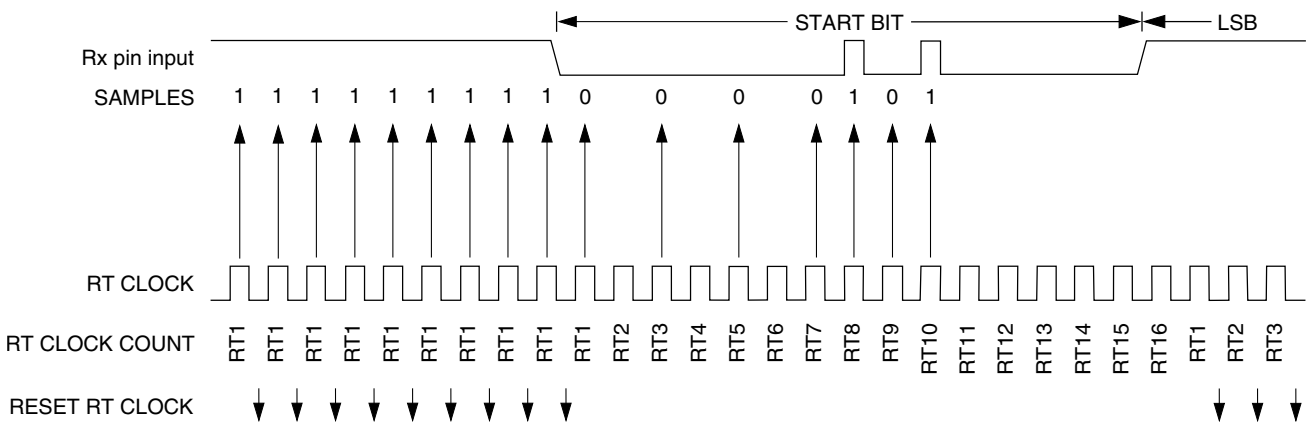


Figure 36-130. Start bit search example 6

36.4.2.5 Framing errors

If the data recovery logic does not detect a logic 1 where the stop bit should be in an incoming frame, it sets the framing error flag, S1[FE], if S2[LBKDE] is disabled. When S2[LBKDE] is disabled, a break character also sets the S1[FE] because a break character has no stop bit. S1[FE] is set at the same time that received data is placed in the receive data buffer. Framing errors are not supported when C7816[ISO7816E] is set/enabled. However, if S1[FE] is set, data will not be received when C7816[ISO7816E] is set.

36.4.2.6 Receiving break characters

The UART recognizes a break character when a start bit is followed by eight, nine, or ten logic 0 data bits and a logic 0 where the stop bit should be. Receiving a break character has these effects on UART registers:

- Sets the framing error flag, S1[FE].
- Writes an all 0 dataword to the data buffer, which may cause S1[RDRF] to set, depending on the watermark and number of values in the data buffer.
- May set the overrun flag, S1[OR], noise flag, S1[NF], parity error flag, S1[PE], or the receiver active flag, S2[RAF].

The detection threshold for a break character can be adjusted when using an internal oscillator in a LIN system by setting S2[LBKDE]. The UART break character detection threshold depends on C1[M], C1[PE], S2[LBKDE] and C4[M10]. See the following table.

Table 36-131. Receive break character detection threshold

LBKDE	M	M10	PE	Threshold (bits)
0	0	—	—	10
0	1	0	—	11
0	1	1	0	11
0	1	1	1	12
1	0	—	—	11
1	1	—	—	12

While S2[LBKDE] is set, it will have these effects on the UART registers:

- Prevents S1[RDRF], S1[FE], S1[NF], and S1[PF] from being set. However, if they are already set, they will remain set.
- Sets the LIN break detect interrupt flag, S2[LBKDIF], if a LIN break character is received.

Break characters are not detected or supported when C7816[ISO_7816E] is set/enabled.

36.4.2.7 Hardware flow control

To support hardware flow control, the receiver can be programmed to automatically deassert and assert RTS.

- RTS remains asserted until the transfer is complete, even if the transmitter is disabled midway through a data transfer. See Transceiver driver enable using RTS for more details.
- If the receiver request-to-send functionality is enabled, the receiver automatically deasserts RTS if the number of characters in the receiver data register is equal to or greater than receiver data buffer's watermark, RWFIFO[RXWATER].
- The receiver asserts RTS when the number of characters in the receiver data register is less than the watermark. It is not affected if RDRF is asserted.
- Even if RTS is deasserted, the receiver continues to receive characters until the receiver data buffer is full or is overrun.
- If the receiver request-to-send functionality is disabled, the receiver RTS remains deasserted.

The following figure shows receiver hardware flow control functional timing. Along with the actual character itself, RXD shows the start bit. The stop bit can also indicated, with a dashed line, if necessary. The watermark is set to 2.

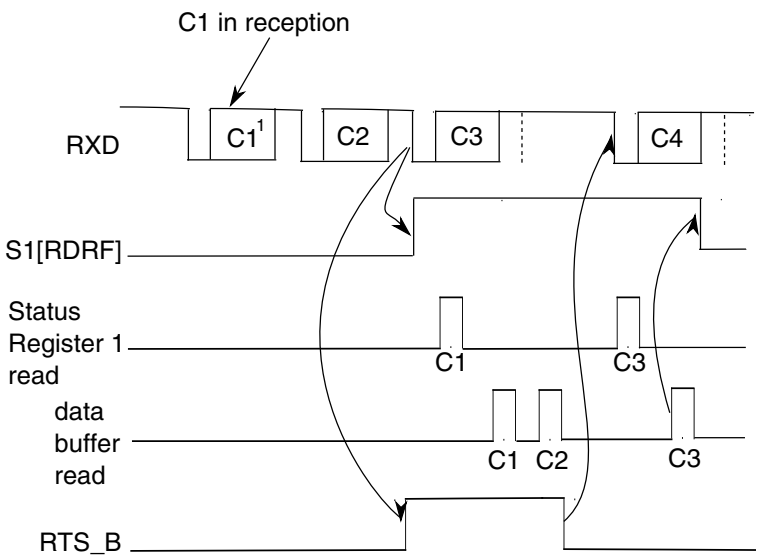


Figure 36-131. Receiver hardware flow control timing diagram

36.4.2.8 Baud rate tolerance

A transmitting device may be operating at a baud rate below or above the receiver baud rate. Accumulated bit time misalignment can cause one of the three stop bit data samples (RT8, RT9, and RT10) to fall outside the actual stop bit. A noise error will occur if the RT8, RT9, and RT10 samples are not all the same logical values. A framing error will occur if the receiver clock is misaligned in such a way that the majority of the RT8, RT9, and RT10 stop bit samples are a logic 0.

As the receiver samples an incoming frame, it resynchronizes the RT clock on any valid falling edge within the frame. Resynchronization within frames corrects a misalignment between transmitter bit times and receiver bit times.

36.4.2.8.1 Slow data tolerance

The following figure shows how much a slow received frame can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

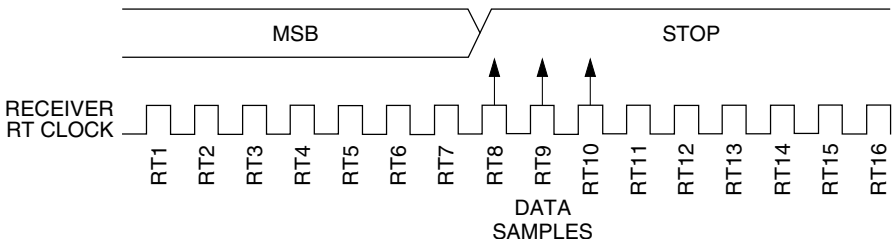


Figure 36-132. Slow data

For an 8-bit data character, data sampling of the stop bit takes the receiver 154 RT cycles (9 bit times × 16 RT cycles + 10 RT cycles).

With the misaligned character shown in the Figure 36-132, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 147 RT cycles (9 bit times × 16 RT cycles + 3 RT cycles).

The maximum percent difference between the receiver count and the transmitter count of a slow 8-bit data character with no errors is:

$$((154 - 147) \div 154) \times 100 = 4.54\%$$

For a 9-bit data character, data sampling of the stop bit takes the receiver 170 RT cycles (10 bit times × 16 RT cycles + 10 RT cycles).

With the misaligned character shown in the Figure 36-132, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 163 RT cycles (10 bit times × 16 RT cycles + 3 RT cycles).

The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is:

$$((170 - 163) \div 170) \times 100 = 4.12\%$$

36.4.2.8.2 Fast data tolerance

The following figure shows how much a fast received frame can be misaligned. The fast stop bit ends at RT10 instead of RT16 but is still sampled at RT8, RT9, and RT10.

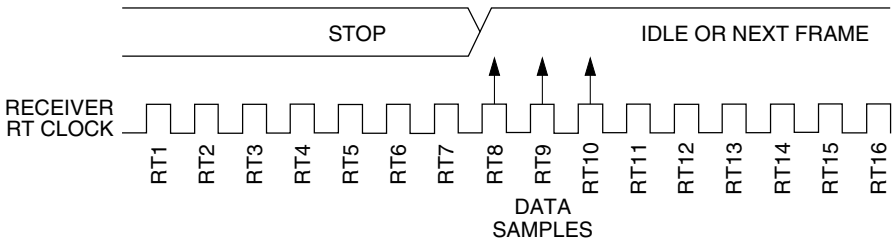


Figure 36-133. Fast data

For an 8-bit data character, data sampling of the stop bit takes the receiver 154 RT cycles (9 bit times × 16 RT cycles + 10 RT cycles).

With the misaligned character shown in the Figure 36-133, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 160 RT cycles (10 bit times × 16 RT cycles).

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is:

$$((154 - 160) \div 154) \times 100 = 3.90\%$$

For a 9-bit data character, data sampling of the stop bit takes the receiver 170 RT cycles (10 bit times × 16 RT cycles + 10 RT cycles).

With the misaligned character shown in the Figure 36-133, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 176 RT cycles (11 bit times × 16 RT cycles).

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is:

$$((170 - 176) \div 170) \times 100 = 3.53\%$$

36.4.2.9 Receiver wakeup

C1[WAKE] determines how the UART is brought out of the standby state to process an incoming message. C1[WAKE] enables either idle line wakeup or address mark wakeup.

Receiver wakeup is not supported when C7816[ISO_7816E] is set/enabled because multi-receiver systems are not allowed.

36.4.2.9.1 Idle input line wakeup (C1[WAKE] = 0)

In this wakeup method, an idle condition on the unsynchronized receiver input signal clears C2[RWU] and wakes the UART. The initial frame or frames of every message contain addressing information. All receivers evaluate the addressing information, and receivers for which the message is addressed process the frames that follow. Any receiver for which a message is not addressed can set its C2[RWU] and return to the standby state. C2[RWU] remains set and the receiver remains in standby until another idle character appears on the unsynchronized receiver input signal.

Idle line wakeup requires that messages be separated by at least one idle character and that no message contains idle characters.

When C2[RWU] is 1 and S2[RWUID] is 0, the idle character that wakes the receiver does not set S1[IDLE] or the receive data register full flag, S1[RDRF]. The receiver wakes and waits for the first data character of the next message which is stored in the receive data buffer. When S2[RWUID] and C2[RWU] are set and C1[WAKE] is cleared, any idle condition sets S1[IDLE] and generates an interrupt if enabled.

Idle input line wakeup is not supported when C7816[ISO_7816E] is set/enabled.

36.4.2.9.2 Address mark wakeup (C1[WAKE] = 1)

In this wakeup method, a logic 1 in the bit position immediately preceding the stop bit of a frame clears C2[RWU] and wakes the UART. A logic 1 in the bit position immediately preceeding the stop bit marks a frame as an address frame that contains addressing information. All receivers evaluate the addressing information, and the receivers for which the message is addressed process the frames that follow. Any receiver for which a message is not addressed can set its C2[RWU] and return to the standby state. C2[RWU] remains set and the receiver remains in standby until another address frame appears on the unsynchronized receiver input signal.

A logic 1 in the bit position immediately preceding the stop bit clears the receiver's C2[RWU] before the stop bit is received and places the received data into the receiver data buffer.

Address mark wakeup allows messages to contain idle characters but requires that the bit position immediately preceding the stop bit be reserved for use in address frames.

If module is in standby mode and nothing triggers to wake the UART, no error flag is set even if an invalid error condition is detected on the receiving data line.

Address mark wakeup is not supported when C7816[ISO_7816E] is set/enabled.

36.4.2.9.3 Match address operation

Match address operation is enabled when C4[MAEN1] or C4[MAEN2] is set. In this function, a frame received by the RX pin with a logic 1 in the bit position immediately preceding the stop bit is considered an address and is compared with the associated MA1 or MA2 register. The frame is transferred to the receive buffer, and S1[RDRF] is set, only if the comparison matches. All subsequent frames received with a logic 0 in the bit position 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 frames with logic 0 in the bit position immediately preceding the stop bit are also discarded. If both C4[MAEN1] and C4[MAEN2] are negated, the receiver operates normally and all data received is transferred to the receive data buffer.

Match address operation functions in the same way for both MA1 and MA2 registers.

- If only one of C4[MAEN1] and C4[MAEN2] is asserted, a marked address is compared only with the associated match register and data is transferred to the receive data buffer only on a match.
- If C4[MAEN1] and C4[MAEN2] are asserted, a marked address is compared with both match registers and data is transferred only on a match with either register.

Address match operation is not supported when C7816[ISO_7816E] is set/enabled.

36.4.3 Baud rate generation

A 13-bit modulus counter and a 5-bit fractional fine-adjust counter in the baud rate generator derive the baud rate for both the receiver and the transmitter. The value from 1 to 8191 written to SBR[12:0] determines the module clock divisor. The SBR bits are in the UART baud rate registers, BDH and BDL. The baud rate clock is synchronized with the module clock and drives the receiver. The fractional fine-adjust counter adds fractional delays to the baud rate clock to allow fine trimming of the baud rate to match the system baud rate. The transmitter is driven by the baud rate clock divided by 16. The receiver has an acquisition rate of 16 samples per bit time.

Baud rate generation is subject to two sources of error:

- Integer division of the module clock may not give the exact target frequency. This error can be reduced with the fine-adjust counter.
- Synchronization with the module clock can cause phase shift.

The Table 36-132 lists the available baud divisor fine adjust values.

UART baud rate = UART module clock / (16 × (SBR[12:0] + BRFD))

The following table lists some examples of achieving target baud rates with a module clock frequency of 10.2 MHz, with and without fractional fine adjustment.

Table 36-132. Baud rates (example: module clock = 10.2 MHz)

Bits SBR (decimal)	Bits BRFA	BRFD value	Receiver clock (Hz)	Transmitter clock (Hz)	Target Baud rate	Error (%)
17	00000	0	600,000.0	37,500.0	38,400	2.3
16	10011	19/32=0.59375	614,689.3	38,418.08	38,400	0.047
33	00000	0	309,090.9	19,318.2	19,200	0.62
33	00110	6/32=0.1875	307,344.6	19,209.04	19,200	0.047
66	00000	0	154,545.5	9659.1	9600	0.62
133	00000	0	76,691.7	4793.2	4800	0.14
266	00000	0	38,345.9	2396.6	2400	0.14
531	00000	0	19,209.0	1200.6	1200	0.11
1062	00000	0	9604.5	600.3	600	0.05
2125	00000	0	4800.0	300.0	300	0.00
4250	00000	0	2400.0	150.0	150	0.00
5795	00000	0	1760.1	110.0	110	0.00

Table 36-133. Baud rate fine adjust

BRFA	Baud Rate Fractional Divisor (BRFD)
0 0 0 0 0	0/32 = 0
0 0 0 0 1	1/32 = 0.03125
0 0 0 1 0	2/32 = 0.0625
0 0 0 1 1	3/32 = 0.09375
0 0 1 0 0	4/32 = 0.125
0 0 1 0 1	5/32 = 0.15625
0 0 1 1 0	6/32 = 0.1875
0 0 1 1 1	7/32 = 0.21875
0 1 0 0 0	8/32 = 0.25
0 1 0 0 1	9/32 = 0.28125
0 1 0 1 0	10/32 = 0.3125
0 1 0 1 1	11/32 = 0.34375
0 1 1 0 0	12/32 = 0.375
0 1 1 0 1	13/32 = 0.40625
0 1 1 1 0	14/32 = 0.4375
0 1 1 1 1	15/32 = 0.46875
1 0 0 0 0	16/32 = 0.5

Table continues on the next page...

Table 36-133. Baud rate fine adjust (continued)

BRFA	Baud Rate Fractional Divisor (BRFD)
1 0 0 0 1	17/32 = 0.53125
1 0 0 1 0	18/32 = 0.5625
1 0 0 1 1	19/32 = 0.59375
1 0 1 0 0	20/32 = 0.625
1 0 1 0 1	21/32 = 0.65625
1 0 1 1 0	22/32 = 0.6875
1 0 1 1 1	23/32 = 0.71875
1 1 0 0 0	24/32 = 0.75
1 1 0 0 1	25/32 = 0.78125
1 1 0 1 0	26/32 = 0.8125
1 1 0 1 1	27/32 = 0.84375
1 1 1 0 0	28/32 = 0.875
1 1 1 0 1	29/32 = 0.90625
1 1 1 1 0	30/32 = 0.9375
1 1 1 1 1	31/32 = 0.96875

36.4.4 Data format (non ISO-7816)

Each data character is contained in a frame that includes a start bit and a stop bit. The rest of the data format depends upon C1[M], C1[PE], S2[MSBF], and C4[M10].

36.4.4.1 Eight-bit configuration

Clearing C1[M] configures the UART for 8-bit data characters, that is, eight bits are memory mapped in D. A frame with eight data bits has a total of 10 bits. The most significant bit of the eight data bits can be used as an address mark to wake the receiver. If the most significant bit is used in this way, then it serves as an address or data indication, leaving the remaining seven bits as actual data. When C1[PE] is set, the eighth data bit is automatically calculated as the parity bit. See the following table.

Table 36-134. Configuration of 8-bit data format

UART_C1[PE]	Start bit	Data bits	Address bits	Parity bits	Stop bit
0	1	8	0	0	1
0	1	7	1[1]	0	1
1	1	7	0	1	1

1. The address bit identifies the frame as an address character. See Receiver wakeup.

36.4.4.2 Nine-bit configuration

When C1[M] is set and C4[M10] is cleared, the UART is configured for 9-bit data characters. If C1[PE] is enabled, the ninth bit is either C3[T8/R8] or the internally generated parity bit. This results in a frame consisting of a total of 11 bits. In the event that the ninth data bit is selected to be C3[T8], it will remain unchanged after transmission and can be used repeatedly without rewriting it, unless the value needs to be changed. This feature may be useful when the ninth data bit is being used as an address mark.

When C1[M] and C4[M10] are set, the UART is configured for 9-bit data characters, but the frame consists of a total of 12 bits. The 12 bits include the start and stop bits, the 9 data character bits, and a tenth internal data bit. Note that if C4[M10] is set, C1[PE] must also be set. In this case, the tenth bit is the internally generated parity bit. The ninth bit can either be used as an address mark or a ninth data bit.

See the following table.

Table 36-135. Configuration of 9-bit data formats

C1[PE]	UC1[M]	C1[M10]	Start bit	Data bits	Address bits	Parity bits	Stop bit
0	0	0	See Eight-bit configuration				
0	0	1	Invalid configuration				
0	1	0	1	9	0	0	1
0	1	0	1	8	1[1]	0	1
0	1	1	Invalid Configuration				
1	0	0	See Eight-bit configuration				
1	0	1	Invalid Configuration				
1	1	0	1	8	0	1	1
1	1	1	1	9	0	1	1
1	1	1	1	8	1[2]	1	1

1. The address bit identifies the frame as an address character.
2. The address bit identifies the frame as an address character.

Note

Unless in 9-bit mode with M10 set, do not use address mark wakeup with parity enabled.

36.4.4.3 Timing examples

Timing examples of these configurations in the NRZ mark/space data format are illustrated in the following figures. The timing examples show all of the configurations in the following sub-sections along with the LSB and MSB first variations.

36.4.4.3.1 Eight-bit format with parity disabled

The most significant bit can be used for address mark wakeup.

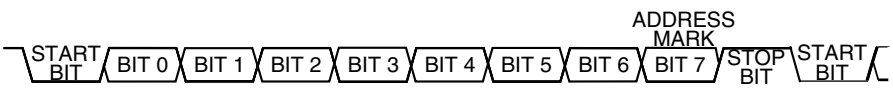


Figure 36-134. Eight bits of data with LSB first

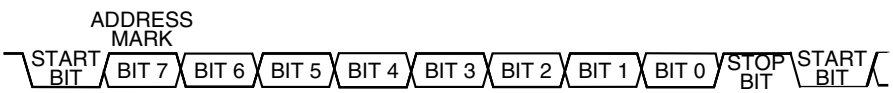


Figure 36-135. Eight bits of data with MSB first

36.4.4.3.2 Eight-bit format with parity enabled

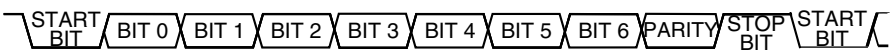


Figure 36-136. Seven bits of data with LSB first and parity

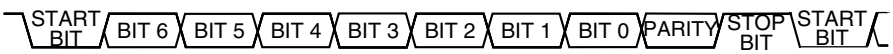


Figure 36-137. Seven bits of data with MSB first and parity

36.4.4.3.3 Nine-bit format with parity disabled

The most significant bit can be used for address mark wakeup.

Figure 36-138. Nine bits of data with LSB first

Figure 36-139. Nine bits of data with MSB first

36.4.4.3.4 Nine-bit format with parity enabled

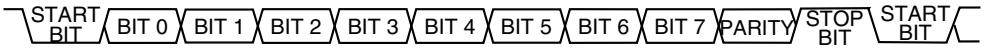


Figure 36-140. Eight bits of data with LSB first and parity

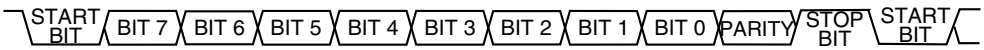


Figure 36-141. Eight bits of data with MSB first and parity

36.4.4.3.5 Non-memory mapped tenth bit for parity

The most significant memory-mapped bit can be used for address mark wakeup.

Figure 36-142. Nine bits of data with LSB first and parity

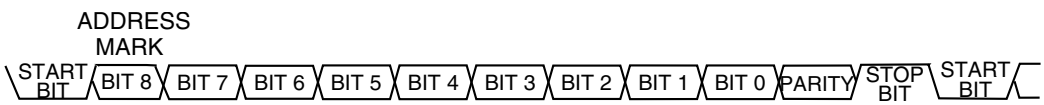


Figure 36-143. Nine bits of data with MSB first and parity

36.4.5 Single-wire operation

Normally, the UART uses two pins for transmitting and receiving. In single wire operation, the RXD pin is disconnected from the UART and the UART implements a half-duplex serial connection. The UART uses the TXD pin for both receiving and transmitting.

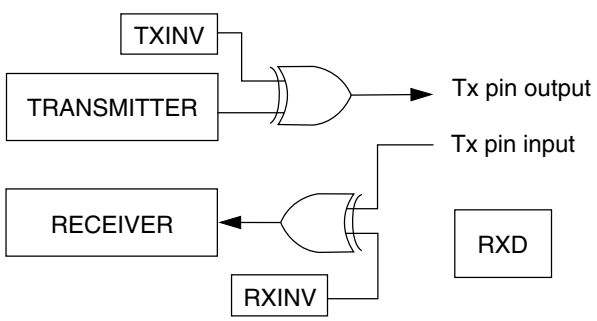


Figure 36-144. Single-wire operation (C1[LOOPS] = 1, C1[RSRC] = 1)

Enable single wire operation by setting C1[LOOPS] and the receiver source field, C1[RSRC]. Setting C1[LOOPS] disables the path from the unsynchronized receiver input signal to the receiver. Setting C1[RSRC] connects the receiver input to the output of the

TXD pin driver. Both the transmitter and receiver must be enabled (C2[TE] = 1 and C2[RE] = 1). When C7816[ISO_7816EN] is set, it is not required that both C2[TE] and C2[RE] are set.

36.4.6 Loop operation

In loop operation, the transmitter output goes to the receiver input. The unsynchronized receiver input signal is disconnected from the UART.

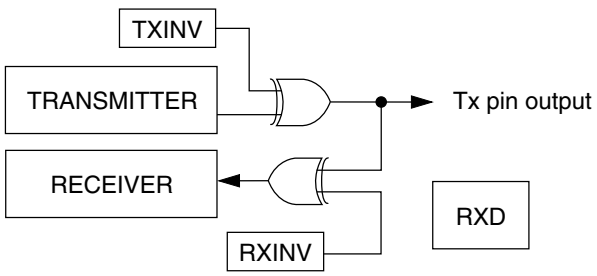


Figure 36-145. Loop operation (C1[LOOPS] = 1, C1[RSRC] = 0)

Enable loop operation by setting C1[LOOPS] and clearing C1[RSRC]. Setting C1[LOOPS] disables the path from the unsynchronized receiver input signal to the receiver. Clearing C1[RSRC] connects the transmitter output to the receiver input. Both the transmitter and receiver must be enabled (C2[TE] = 1 and C2[RE] = 1). When C7816[ISO_7816EN] is set, it is not required that both C2[TE] and C2[RE] are set.

36.4.7 ISO-7816/smartcard support

The UART provides mechanisms to support the ISO-7816 protocol that is commonly used to interface with smartcards. The ISO-7816 protocol is an NRZ, single wire, half-duplex interface. The TxD pin is used in open-drain mode because the data signal is used for both transmitting and receiving. There are multiple subprotocols within the ISO-7816 standard. The UART supports both T = 0 and T = 1 protocols. The module also provides for automated initial character detection and configuration, which allows for support of both direct convention and inverse convention data formats. A variety of interrupts specific to 7816 are provided in addition to the general interrupts to assist software. Additionally, the module is able to provide automated NACK responses and has programmed automated retransmission of failed packets. An assortment of programmable timeouts and guard band times are also supported.

The term elemental time unit (ETU) is frequently used in the context of ISO-7816. This concept is used to relate the frequency that the system (UART) is running at and the frequency that data is being transmitted and received. One ETU represents the time it

takes to transmit or receive a single bit. For example, a standard 7816 packet, excluding any guard time or NACK elements is 10 ETUs (start bit, 8 data bits, and a parity bit). Guard times and wait times are also measured in ETUs.,

NOTE

The ISO-7816 specification may have certain configuration options that are reserved. To maintain maximum flexibility to support future 7816 enhancements or devices that may not strictly conform to the specification, the UART does not prevent those options being used. Further, the UART may provide configuration options that exceed the flexibility of options explicitly allowed by the 7816 specification. Failure to correctly configure the UART may result in unexpected behavior or incompatibility with the ISO-7816 specification.

36.4.7.1 Initial characters

In ISO-7816 with T = 0 mode, the UART can be configured to use C7816[INIT] to detect the next valid initial character, referred to by the ISO-7816 specifically as a TS character. When the initial character is detected, the UART provides the host processor with an interrupt if IE7816[INITDE] is set. Additionally, the UART will alter S2[MSBF], C3[TXINV], and S2[RXINV] automatically, based on the initial character. The corresponding initial character and resulting register settings are listed in the following table.

Table 36-136. Initial character automated settings

Initial character (bit 1-10)	Initial character (hex)	MSBF	TXINV	RXINV
LHHL LLL LLH inverse convention	3F	1	1	1
LHHL HHH LLH direct convention	3B	0	0	0

S2[MSBF], C3[TXINV], and S2[RXINV] must be reset to their default values before C7816[INIT] is set. Once C7816[INIT] is set, the receiver searches all received data for the first valid initial character. Detecting a Direct Convention Initial Character will cause no change to S2[MSBF], C3[TXINV], and S2[RXINV], while detecting an Inverse Convention Initial Character will cause these fields to set automatically. All data received, which is not a valid initial character, is ignored and all flags resulting from the invalid data are blocked from asserting. If C7816[ANACK] is set, a NACK is returned for invalid received initial characters and an RXT interrupt is generated as programmed.

36.4.7.2 Protocol T = 0

When T = 0 protocol is selected, a relatively complex error detection scheme is used. Data characters are formatted as illustrated in the following figure. This scheme is also used for answer to reset and Peripheral Pin Select (PPS) formats.

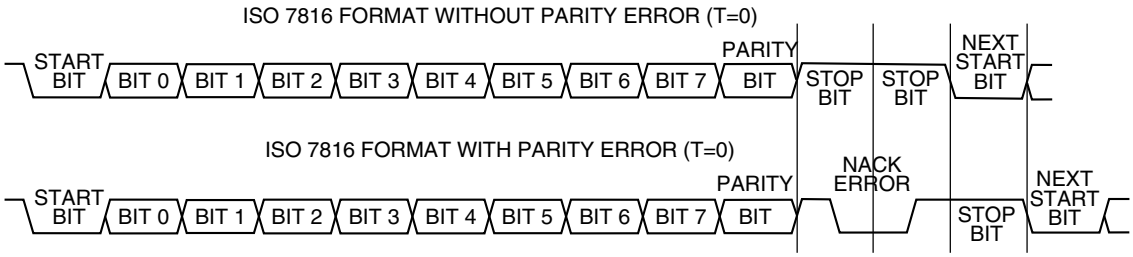


Figure 36-146. ISO-7816 T = 0 data format

As with other protocols supported by the UART, the data character includes a start bit. However, in this case, there are two stop bits rather than the typical single stop bit. In addition to a standard even parity check, the receiver has the ability to generate and return a NACK during the second half of the first stop bit period. The NACK must be at least one time period (ETU) in length and no more than two time periods (ETU) in length. The transmitter must wait for at least two time units (ETU) after detection of the error signal before attempting to retransmit the character.

It is assumed that the UART and the device (smartcard) know in advance which device is receiving and which is transmitting. No special mechanism is supplied by the UART to control receive and transmit in the mode other than C2[TE] and C2[RE]. Initial Character Detect feature is also supported in this mode.

36.4.7.3 Protocol T = 1

When T = 1 protocol is selected, the NACK error detection scheme is not used. Rather, the parity bit is used on a character basis and a CRC or LRC is used on the block basis, that is, for each group of characters. In this mode, the data format allows for a single stop bit although additional inactive bit periods may be present between the stop bit and the next start bit. Data characters are formatted as illustrated in the following figure.

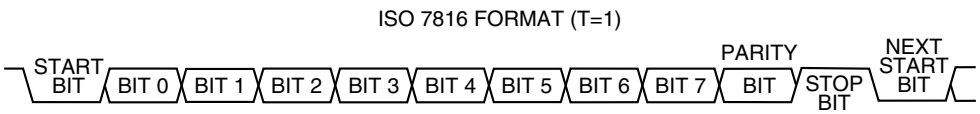


Figure 36-147. ISO 7816 T=1 data format

The smallest data unit that is transferred is a block. A block is made up of several data characters and may vary in size depending on the block type. The UART does not provide a mechanism to decode the block type. As part of the block, an LRC or CRC is included. The UART does not calculate the CRC or LRC for transmitted blocks, nor does it verify the validity of the CRC or LRC for received blocks. The 7816 protocol requires that the initiator and the smartcard (device) takes alternate turns in transmitting and receiving blocks. When the UART detects that the last character in a block has been transmitted it will automatically clear C2[TE] and enter receive mode. Therefore, the software must program the transmit buffer with the next data to be transmitted, and then enable C2[TE], once the software has determined that the last character of the received block has been received. The UART detects that the last character of the transmit block has been sent when TL7816[TLEN] = 0 and four additional characters have been sent. The four additional characters are made up of three prior to TL7816[TLEN] decrementing (prologue) and one after TL7816[TLEN] = 0, the final character of the epilogue.

36.4.7.4 Wait time and guard time parameters

The ISO-7816 specification defines several wait time and guard time parameters. The UART allows for flexible configuration and violation detection of these settings. On reset, the wait time (IS7816[WT]) defaults to 9600 ETUs and guard time (GT) to 12 ETUs. These values are controlled by parameters in the WP7816, WN7816, and WF7816 registers. Additionally, the value of C7816[TTYPE] also factors into the calculation. The formulae used to calculate the number ETUs for each wait time and guard time value are shown in Table 36-137.

Wait time (WT) is defined as the maximum allowable time between the leading edge of a character transmitted by the smartcard device and the leading edge of the previous character that was transmitted by the UART or the device. Similarly, character wait time (CWT) is defined as the maximum allowable time between the leading edge of two characters within the same block. Block wait time (BWT) is defined as the maximum time between the leading edge character of the last block received by the smartcard device and the leading edge of the first character transmitted by the smartcard device.

Guard time (GT) is defined as the minimum allowable time between the leading edge of two consecutive characters. Character guard time (CGT) is the minimum allowable time between the leading edges of two consecutive characters in the same direction, that is, transmission or reception. Block guard time (BGT) is the minimum allowable time between the leading edges of two consecutive characters in opposite directions, that is, transmission then reception or reception then transmission.

The GT and WT counters reset whenever C7816[TTYPE] = 1 or C7816[ISO_7816E] = 0 or a new dataword start bit has been received or transmitted as specified by the counter descriptions. The CWT, CGT, BWT, BGT counters reset whenever C7816[TTYPE] = 0 or C7816[ISO_7816E] = 0 or a new dataword start bit is received or transmitted as specified by the counter descriptions. When C7816[TTYPE] = 1, some of the counter values require an assumption regarding the first data transferred when the UART first starts. This assumption is required when the 7816E is disabled, when transition from C7816[TTYPE] = 0 to C7816[TTYPE] = 1 or when coming out of reset. In this case, it is assumed that the previous non-existent transfer was a received transfer.

The UART will automatically handle GT, CGT, and BGT such that the UART will not send a packet before the corresponding guard time expiring.

Table 36-137. Wait and guard time calculations

Parameter	Reset value [ETU]	C7816[TTYPE] = 0 [ETU]	C7816[TTYPE] = 1 [ETU]
Wait time (WT)	9600	WI × 960 × GTFD	Not used
Character wait time (CWT)	Not used	Not used	$11 + 2^{CWI}$
Block wait time (BWT)	Not used	Not used	$11 + 2^{BWI} \times 960 \times GTFD$
Guard time (GT)	12	**GTN not wqual to 255** 12 + GTN **GTN wqual to 255** 12	Not used
Character guard time (CGT)	Not used	Not used	**GTN not equal to 255** 12 + GTN **GTN equal to 255** 11
Block guard time (BGT)	Not used	Not used	22

36.4.7.5 Baud rate generation

The value in WF7816[GTFD] does not impact the clock frequency. SBR and BRFD are used to generate the clock frequency. This clock frequency is used by the UART only and is not seen by the smartcard device. The transmitter clocks operates at 1/16 the frequency of the receive clock so that the receiver is able to sample the received value 16 times during the ETU.

36.4.7.6 UART restrictions in ISO-7816 operation

Due to the flexibility of the UART module, there are several features and interrupts that are not supported while running in ISO-7816 mode. These restrictions are documented within the register field definitions.

36.5 Reset

All registers reset to a particular value are indicated in Memory map and registers.

36.6 System level interrupt sources

There are several interrupt signals that are sent from the UART. The following table lists the interrupt sources generated by the UART. The local enables for the UART interrupt sources are described in this table. Details regarding the individual operation of each interrupt are contained under various sub-sections of Memory map and registers. However, RXEDGIF description also outlines additional details regarding the RXEDGIF interrupt because of its complexity of operation. Any of the UART interrupt requests listed in the table can be used to bring the CPU out of Wait mode.

Table 36-138. UART interrupt sources

Interrupt Source	Flag	Local enable	DMA select
Transmitter	TDRE	TIE	TDMAS = 0
Transmitter	TC	TCIE	TCDMAS = 0
Receiver	IDLE	ILIE	
Receiver	RDRF	RIE	RDMAS = 0
Receiver	LBKDIF	LBKDIE	-
Receiver	RXEDGIF	RXEDGIE	-
Receiver	OR	ORIE	-
Receiver	NF	NEIE	-
Receiver	FE	FEIE	-
Receiver	PF	PEIE	-
Receiver	RXUF	RXUFE	-
Transmitter	TXOF	TXOFE	-
Receiver	WT	WTWE	-
Receiver	CWT	CWTE	-
Receiver	BWT	BWTE	-
Receiver	INITD	INITDE	-
Receiver	TXT	TXTE	-

Table continues on the next page...

Table 36-138. UART interrupt sources (continued)

Interrupt Source	Flag	Local enable	DMA select
Receiver	RXT	RXTE	-
Receiver	GTV	GTVE	-

36.6.1 RXEDGIF description

S2[RXEDGIF] is set when an active edge is detected on the RxD pin. Therefore, the active edge can be detected only when in two wire mode. A RXEDGIF interrupt is generated only when S2[RXEDGIF] is set. If RXEDGIE is not enabled before S2[RXEDGIF] is set, an interrupt is not generated until S2[RXEDGIF] is set.

36.6.1.1 RxD edge detect sensitivity

Edge sensitivity can be software programmed to be either falling or rising. The polarity of the edge sensitivity is selected using S2[RXINV]. To detect the falling edge, S2[RXINV] is programmed to 0. To detect the rising edge, S2[RXINV] is programmed to 1.

Synchronizing logic is used prior to detect edges. Prior to detecting an edge, the receive data on RxD input must be at the deasserted logic level. A falling edge is detected when the RxD input signal is seen as a logic 1 (the deasserted level) during one module clock cycle, and then a logic 0 (the asserted level) during the next cycle. A rising edge is detected when the input is seen as a logic 0 during one module clock cycle and then a logic 1 during the next cycle.

36.6.1.2 Clearing RXEDGIF interrupt request

Writing a logic 1 to S2[RXEDGIF] immediately clears the RXEDGIF interrupt request even if the RxD input remains asserted. S2[RXEDGIF] remains set if another active edge is detected on RxD while attempting to clear S2[RXEDGIF] by writing a 1 to it.

36.6.1.3 Exit from low-power modes

The receive input active edge detect circuit is still active on low power modes (Wait and Stop). An active edge on the receive input brings the CPU out of low power mode if the interrupt is not masked (S2[RXEDGIF] = 1).

36.7 DMA operation

In the transmitter, S1[TDRE] and S1[TC] can be configured to assert a DMA transfer request. In the receiver, S1[RDRF], can be configured to assert a DMA transfer request. The following table shows the configuration field settings required to configure each flag for DMA operation.

Table 36-139. DMA configuration

Flag	Request enable bit	DMA select bit
TDRE	TIE = 1	TDMAS = 1
RDRF	RIE = 1	RDMAS = 1

When a flag is configured for a DMA request, its associated DMA request is asserted when the flag is set. When S1[RDRF] is configured as a DMA request, the clearing mechanism of reading S1, followed by reading D, does not clear the associated flag. The DMA request remains asserted until an indication is received that the DMA transactions are done. When this indication is received, the flag bit and the associated DMA request is cleared. If the DMA operation failed to remove the situation that caused the DMA request, another request is issued.

36.8 Application information

This section describes the UART application information.

36.8.1 Transmit/receive data buffer operation

The UART has independent receive and transmit buffers. The size of these buffers may vary depending on the implementation of the module. The implemented size of the buffers is a fixed constant via PFIFO[TXFIFOSIZE] and PFIFO[RXFIFOSIZE]. Additionally, legacy support is provided that allows for the FIFO structure to operate as a depth of one. This is the default/reset behavior of the module and can be adjusted using the PFIFO[RXFE] and PFIFO[TXFE] bits. Individual watermark levels are also provided for transmit and receive.

There are multiple ways to ensure that a data block, which is a set of characters, has completed transmission. These methods include:

1. Set TXFIFO[TXWATER] to 0. TDRE asserts when there is no further data in the transmit buffer. Alternatively the S1[TC] flag can be used to indicate when the transmit shift register is also empty.
2. Poll TCFIFO[TXCOUNT]. Assuming that only data for a data block has been put into the data buffer, when TCFIFO[TXCOUNT] = 0, all data has been transmitted or is in the process of transmission.
3. S1[TC] can be monitored. When S1[TC] asserts, it indicates that all data has been transmitted and there is no data currently being transmitted in the shift register.

36.8.2 ISO-7816 initialization sequence

This section outlines how to program the UART for ISO-7816 operation. Elements such as procedures to power up or power down the smartcard, and when to take those actions, are beyond the scope of this description. To set up the UART for ISO-7816 operation:

1. Select a baud rate. Write this value to the UART baud registers (BDH/L) to begin the baud rate generator. Remember that the baud rate generator is disabled when the baud rate is zero. Writing to the BDH has no effect without also writing to BDL. According to the 7816 specification the initial (default) baud rating setting should be Fi = 372 and Di = 1 and a maximum frequency of 5 MHz. In other words, the BDH, BDL, and C4 registers should be programmed such that the transmission frequency provided to the smartcard device must be 1/372th of the clock and must not exceed 5 MHz.
2. Write to set BDH[LBKDIE] = 0.
3. Write to C1 to configure word length, parity, and other configuration fields (LOOPS, RSRC) and set C1[M] = 1, C1[PE] = 1, and C1[PT] = 0.
4. Write to set S2[RWUID] = 0 and S2[LBKDE] = 0.
5. Write to set MODEM[RXRTSE] = 0, MODEM[TXRTSPOL] = 0, MODEM[TXRTSE] = 0, and MODEM[TXCTSE] = 0.
6. Write to set up interrupt enable fields desired (C3[ORIE], C3[NEIE], C3[PEIE], and C3[FEIE])
7. Write to set C4[MAEN1] = 0 and C4[MAEN2] = 0.
8. Write to C5 register and configure DMA control register fields as desired for application.

9. Write to set C7816[INIT] = 1,C7816[TTYPE] = 0, and C7816[ISO_7816E] = 1. Program C7816[ONACK] and C7816[ANACK] as desired.
10. Write to IE7816 to set interrupt enable parameters as desired.
11. Write to ET7816 and set as desired.
12. Write to set C2[ILIE] = 0, C2[RE] = 1, C2[TE] = 1, C2[RWU] = 0, and C2[SBK] = 0. Set up interrupt enables C2[TIE], C2[TCIE], and C2[RIE] as desired.

At this time, the UART will start listening for an initial character. After being identified, it will automatically adjust S2[MSBF], C3[TXINV], and S2[RXINV]. The software must then receive and process an answer to reset. Upon processing the answer to reset, the software must write to set C2[RE] = 0 and C2[TE] = 0. The software should then adjust 7816 specific and UART generic parameters to match and configure data that was received during the answer on reset period. After the new settings have been programmed, including the new baud rate and C7816[TTYPE], C2[RE] and C2[TE] can be reenabled as required.

36.8.2.1 Transmission procedure for (C7816[TTYPE] = 0)

When the protocol selected is C7816[TTYPE] = 0, it is assumed that the software has a prior knowledge of who should be transmitting and receiving. Therefore, no mechanism is provided for automated transmission/receipt control. The software must monitor S1[TDRE], or configure for an interrupt, and provide additional data for transmission, as appropriate. Additionally, software should set C2[TE] = 1 and control TXDIR whenever it is the UART's turn to transmit information. For ease of monitoring, it is suggested that only data be transmitted until the next receiver/transmit switchover is loaded into the transmit FIFO/buffer.

36.8.2.2 Transmission procedure for (C7816[TTYPE] = 1)

When the protocol selected is C7816[TTYPE] = 1, data is transferred in blocks. Before starting a transmission, the software must write the size, in number of bytes, for the Information Field portion of the block into TLEN. If a CRC is being transmitted for the block, the value in TLEN must be one more than the size of the information field. The software must then set C2[TE] = 1 and C2[RE] = 1. The software must then monitor S1[TDRE]/interrupt and write the prologue, information, and epilogue field to the transmit buffer. TLEN automatically decrements, except for prologue bytes and the final epilogue byte. When the final epilogue byte has been transmitted, the UART automatically clears C2[TE] to 0, and the UART automatically starts capturing the

response to the block that was transmitted. After the software has detected the receipt of the response, the transmission process must be repeated as needed with sufficient urgency to ensure that the block wait time and character wait times are not violated.

36.8.3 Initialization sequence (non ISO-7816)

To initiate a UART transmission:

1. Configure the UART.
   a. Select a baud rate. Write this value to the UART baud registers (BDH/L) to begin the baud rate generator. Remember that the baud rate generator is disabled when the baud rate is zero. Writing to the BDH has no effect without also writing to BDL.
   b. Write to C1 to configure word length, parity, and other configuration bits (LOOPS, RSRC, M, WAKE, ILT, PE, and PT). Write to C4, MA1, and MA2 to configure.
   c. Enable the transmitter, interrupts, receiver, and wakeup as required, by writing to C2 (TIE, TCIE, RIE, ILIE, TE, RE, RWU, and SBK), S2 (MSBF and BRK13), and C3 (ORIE, NEIE, PEIE, and FEIE). A preamble or idle character is then shifted out of the transmitter shift register.
2. Transmit procedure for each byte.
   a. Monitor S1[TDRE] by reading S1 or responding to the TDRE interrupt. The amount of free space in the transmit buffer directly using TCFIFO[TXCOUNT] can also be monitored.
   b. If the TDRE flag is set, or there is space in the transmit buffer, write the data to be transmitted to (C3[T8]/D). A new transmission will not result until data exists in the transmit buffer.
3. Repeat step 2 for each subsequent transmission.

Note

During normal operation, S1[TDRE] is set when the shift register is loaded with the next data to be transmitted from the transmit buffer and the number of datawords contained in the transmit buffer is less than or equal to the value in TWFIFO[TXWATER]. This occurs 9/16ths of a bit time after the start of the stop bit of the previous frame.

To separate messages with preambles with minimum idle line time, use this sequence between messages.

1. Write the last dataword of the first message to C3[T8]/D.
2. Wait for S1[TDRE] to go high with TWFIFO[TXWATER] = 0, indicating the transfer of the last frame to the transmit shift register.
3. Queue a preamble by clearing and then setting C2[TE].
4. Write the first and subsequent datawords of the second message to C3[T8]/D.

36.8.4 Overrun (OR) flag implications

To be flexible, the overrun flag (OR) operates slight differently depending on the mode of operation. There may be implications that need to be carefully considered. This section clarifies the behavior and the resulting implications. Regardless of mode, if a dataword is received while S1[OR] is set, S1[RDRF] and S1[IDLE] are blocked from asserting. If S1[RDRF] or S1[IDLE] were previously asserted, they will remain asserted until cleared.

36.8.4.1 Overrun operation

The assertion of S1[OR] indicates that a significant event has occurred. The assertion indicates that received data has been lost because there was a lack of room to store it in the data buffer. Therefore, while S1[OR] is set, no further data is stored in the data buffer until S1[OR] is cleared. This ensures that the application will be able to handle the overrun condition.

In most applications, because the total amount of lost data is known, the application will attempt to return the system to a known state. Before S1[OR] is cleared, all received data will be dropped. For this, the software does the following.

1. Remove data from the receive data buffer. This could be done by reading data from the data buffer and processing it if the data in the FIFO was still valuable when the overrun event occurred, or using CFIFO[RXFLUSH] to clear the buffer.
2. Clear S1[OR]. Note that if data was cleared using CFIFO[RXFLUSH], then clearing S1[OR] will result in SFIFO[RXUF] asserting. This is because the only way to clear S1[OR] requires reading additional information from the FIFO. Care should be taken to disable the SFIFO[RXUF] interrupt prior to clearing the OR flag and then clearing SFIFO[RXUF] after the OR flag has been cleared.

Note that, in some applications, if an overrun event is responded to fast enough, the lost data can be recovered. For example, when C7816[ISO_7816E] is asserted, C7816[TTYPE]=1 and C7816[ONACK] = 1, the application may reasonably be able to determine whether the lost data will be resent by the device. In this scenario, flushing the receiver data buffer may not be required. Rather, if S1[OR] is cleared, the lost data may be resent and therefore may be recoverable.

When LIN break detect (LBKDE) is asserted, S1[OR] has significantly different behavior than in other modes. S1[OR] will be set, regardless of how much space is actually available in the data buffer, if a LIN break character has been detected and the corresponding flag, S2[LBKDIF], is not cleared before the first data character is received after S2[LBKDIF] asserted. This behavior is intended to allow the software sufficient time to read the LIN break character from the data buffer to ensure that a break character was actually detected. The checking of the break character was used on some older implementations and is therefore supported for legacy reasons. Applications that do not require this checking can simply clear S2[LBKDIF] without checking the stored value to ensure it is a break character.

36.8.5 Overrun NACK considerations

When C7816[ISO_7816E] is enabled and C7816[TTYPE] = 0, the retransmission feature of the 7816 protocol can be used to help avoid lost data when the data buffer overflows. Using C7816[ONACK], the module can be programmed to issue a NACK on an overflow event. Assuming that the smartcard device has implemented retransmission, the lost data will be retransmitted. While useful, there is a programming implication that may require special consideration. The need to transmit a NACK must be determined and committed to prior to the dataword being fully received. While the NACK is being received, it is possible that the application code will read the data buffer such that sufficient room will be made to store the dataword that is being NACKed. Even if room has been made in the data buffer after the transmission of a NACK is completed, the received data will always be discarded as a result of an overflow and the ET7816[RXTHRESHOLD] value will be incremented by one. However, if sufficient space now exists to write the received data which was NACK'ed, S1[OR] will be blocked and kept from asserting.

36.8.6 Match address registers

The two match address registers allow a second match address function for a broadcast or general call address to the serial bus, as an example.

36.8.7 Modem feature

This section describes the modem features.

36.8.7.1 Ready-to-receive using RTS

To help to stop overrun of the receiver data buffer, the RTS signal can be used by the receiver to indicate to another UART that it is ready to receive data. The other UART can send the data when its CTS signal is asserted. This handshaking conforms to the TIA-232-E standard. A transceiver is necessary if the required voltage levels of the communication link do not match the voltage levels of the UART's RTS and CTS signals.

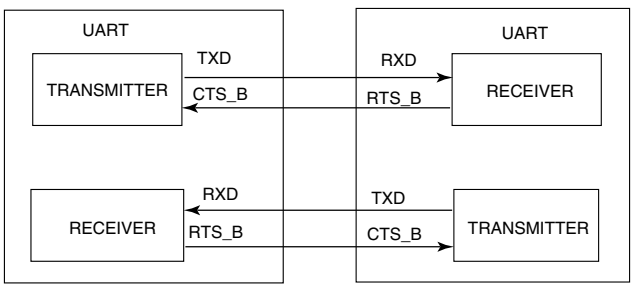


Figure 36-148. Ready-to-receive

The transmitter's CTS signal can be used for hardware flow control whether its RTS signal is used for hardware flow control, transceiver driver enable, or not at all.

36.8.7.2 Transceiver driver enable using RTS

RS-485 is a multiple drop communication protocol in which the UART transceiver's driver is 3-stated unless the UART is driving. The RTS signal can be used by the transmitter to enable the driver of a transceiver. The polarity of RTS can be matched to the polarity of the transceiver's driver enable signal. See the following figure.

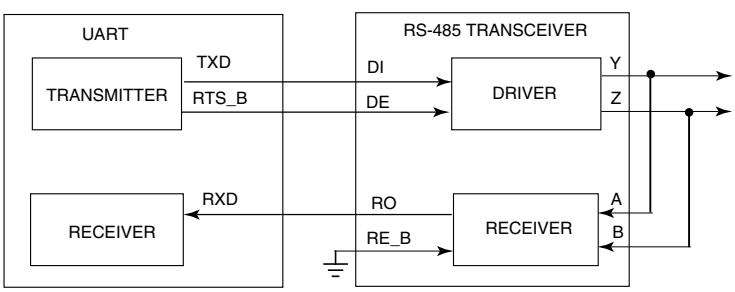


Figure 36-149. Transceiver driver enable using RTS

In the figure, the receiver enable signal is asserted. Another option for this connection is to connect RTS_B to both DE and RE_B. The transceiver's receiver is disabled while driving. A pullup can pull RXD to a non-floating value during this time. This option can be refined further by operating the UART in single wire mode, freeing the RXD pin for other uses.

36.8.8 Clearing 7816 wait timer (WT, BWT, CWT) interrupts

The 7816 wait timer interrupts associated with IS7816[WT], IS7816[BWT], and IS7816[CWT] will automatically reassert if they are cleared and the wait time is still violated. This behavior is similar to most of the other interrupts on the UART. In most cases, if the condition that caused the interrupt to trigger still exists when the interrupt is cleared, then the interrupt will reassert. For example, consider the following scenario:

1. IS7816[WT] is programmed to assert after 9600 cycles of unresponsiveness.
2. The 9600 cycles pass without a response resulting in the WT interrupt asserting.
3. The IS7816[WT] is cleared at cycle 9700 by the interrupt service routine.
4. After the WT interrupt has been cleared, the smartcard remains unresponsive. At cycle 9701 the WT interrupt will be reasserted.

If the intent of clearing the interrupt is such that it does not reassert, the interrupt service routine must remove or clear the condition that originally caused the interrupt to assert prior to clearing the interrupt. There are multiple ways that this can be accomplished, including ensuring that an event that results in the wait timer resetting occurs, such as, the transmission of another packet.

36.8.9 Legacy and reverse compatibility considerations

Recent versions of the UART have added several new features. Whenever reasonably possible, reverse compatibility was maintained. However, in some cases this was either not feasible or the behavior was deemed as not intended. This section describes several differences to legacy operation that resulted from these recent enhancements. If application code from previous versions is used, it must be reviewed and modified to take the following items into account. Depending on the application code, additional items that are not listed here may also need to be considered.

1. Various reserved registers and register bits are used, such as, MSFB and M10.
2. This module now generates an error when invalid address spaces are used.
3. While documentation indicated otherwise, in some cases it was possible for S1[IDLE] to assert even if S1[OR] was set.

4. S1[OR] will be set only if the data buffer (FIFO) does not have sufficient room. Previously, the data buffer was always a fixed size of one and the S1[OR] flag would set so long as S1[RDRF] was set even if there was room in the data buffer. While the clearing mechanism has remained the same for S1[RDRF], keeping the OR flag assertion tied to the RDRF event rather than the data buffer being full would have greatly reduced the usefulness of the buffer when its size is larger than one.
5. Previously, when C2[RWU] was set (and WAKE = 0), the IDLE flag could reassert up to every bit period causing an interrupt and requiring the host processor to reassert C2[RWU]. This behavior has been modified. Now, when C2[RWU] is set (and WAKE = 0), at least one non-idle bit must be detected before an idle can be detected.

Chapter 37
Rapid GPIO (RGPIO)

37.1 Introduction

The Rapid GPIO (RGPIO) module provides a 16-bit general-purpose I/O module directly connected to the processor's high-speed 32-bit local bus. This connection plus support for single-cycle, zero wait-state data transfers allows the RGPIO module to provide improved pin performance when compared to more traditional GPIO modules located on the internal slave peripheral bus.

Many of the pins associated with a device may be used for several different functions. Their primary functions are to provide external interfaces to access off-chip resources. When not used for their primary function, many of the pins may be used as general-purpose digital I/O (GPIO) pins. The definition of the exact pin functions and the affected signals is specific to each device. Every GPIO port, including the RGPIO module, has registers that configure, monitor, and control the port pins.

Note

Most pin functions default to GPIO and must be software configured before using RGPIO.

37.1.1 Overview

The RGPIO module provides 16-bits of high-speed GPIO functionality, mapped to the processor's bus. The key features of this module include:

- 16 bits of high-speed GPIO functionality connected to the processor's local 32-bit bus
- Memory-mapped device connected to the ColdFire core's local bus

- Support for all access sizes: byte, word, and longword
- All reads and writes complete in a single data phase cycle for zero wait-state response

- Data bits can be accessed directly or via alternate addresses to provide set, clear, and toggle functions
  - Alternate addresses allow set, clear, toggle functions using simple store operations without the need for read-modify-write references
- Unique data direction and pin enable control registers
- Package pin toggle rates typically 1.5–3.5x faster than comparable pin mapped onto peripheral bus

A simplified block diagram of the RGPIO module is shown in the following figure. The details of the pin muxing and pad logic are device -specific.

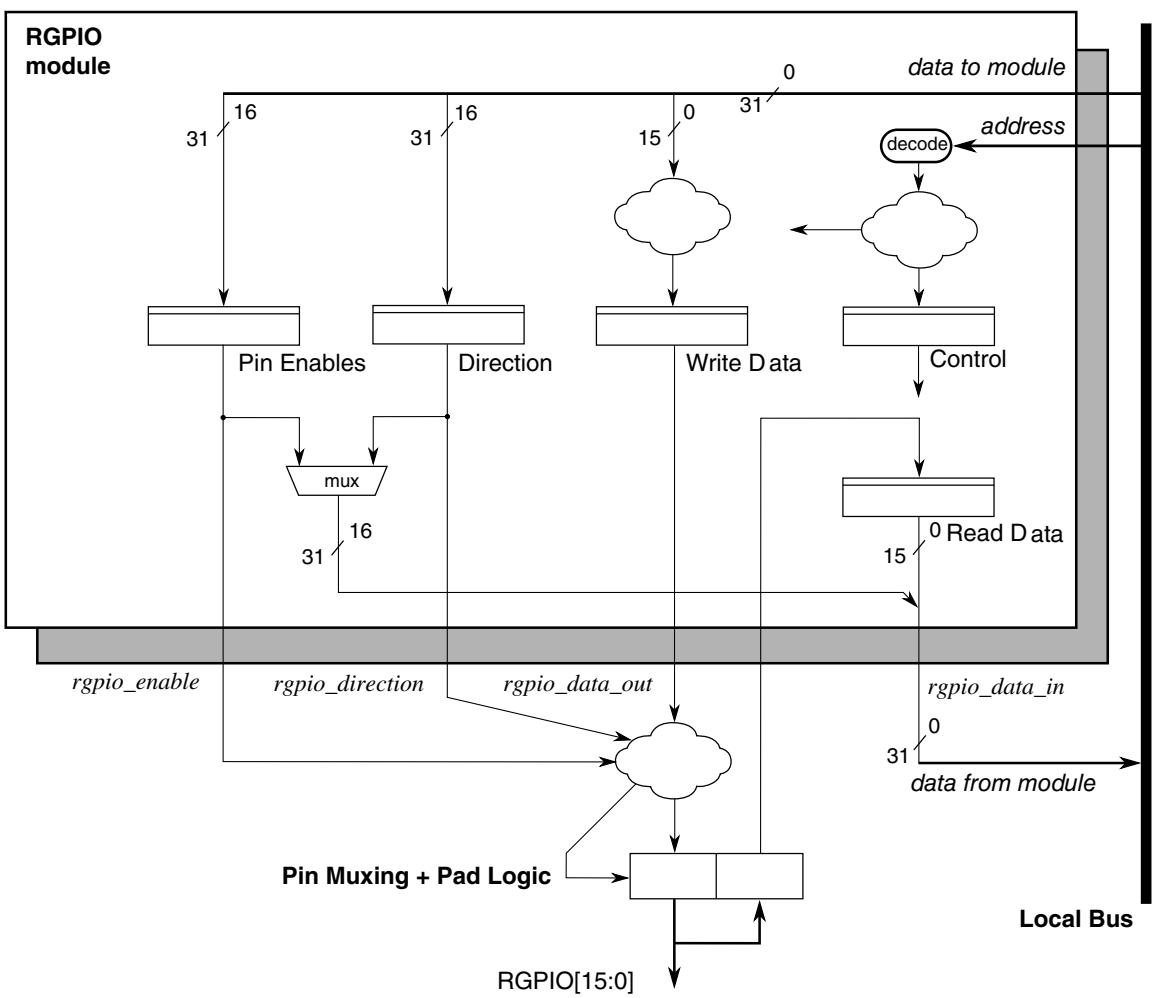


Figure 37-1. RGPIO Block Diagram

37.1.2 Features

The major features of the RGPIO module providing 16 bits of high-speed general-purpose input/output are:

- Small memory-mapped device connected to the processor's local bus
  - All memory references complete in a single cycle to provide zero wait-state responses
  - Located in processor's high-speed clock domain
- Simple programming model
  - Four 16-bit registers, mapped as three program-visible locations
    - Register for pin enables
    - Register for controlling the pin data direction
    - Register for storing output pin data
    - Register for reading current pin state
    - The two data registers (read, write) are mapped to a single program-visible location
  - Alternate addresses to perform data set, clear, and toggle functions using simple writes
  - Separate read and write programming model views enable simplified driver software
    - Support for any access size (byte, word, or longword)

37.1.3 Modes of Operation

The RGPIO module does not support any special modes of operation. As a memory-mapped device located on the processor's high-speed local bus, it responds based strictly on memory address and does not consider the CPU operating mode (supervisor, user) of its references.

37.2 External Signal Description

37.2.1 Overview

As shown in Figure 37-1, the RGPIO module's interface to external logic is indirect via the device pin-muxing and pad logic. The following table shows a list of the associated RGPIO input/output signals.

Table 37-1. RGPIO Module External I/O Signals

Signal Name	Type	Description
RGPIO[15:0]	I/O	RGPIO Data Input/Output

37.2.2 Detailed Signal Descriptions

The following table provides descriptions of the RGPIO module's input and output signals.

Table 37-2. RGPIO Detailed Signal Descriptions

Signal	I/O	Description		
RGPIO[15:0]	I/O	Data Input/Output. When configured as an input, the state of this signal is reflected in the read data register. When configured as an output, this signal is the output of the write data register.		
		State Meaning	Asserted—	
				Input: Indicates the RGPIO pin was sampled as a logic high at the time of the read.
				Output: Indicates a properly-enabled RGPIO output pin is to be driven high.
			Negated—	
				Input: Indicates the RGPIO pin was sampled as a logic low at the time of the read.
				Output: Indicates a properly-enabled RGPIO output pin is to be driven low.
		Timing	Assertion/Negation—	
				Input: Anytime. The input signal is sampled at the rising-edge of the processor's high-speed clock on the data phase cycle of a read transfer of this register.
				Output: Occurs at the rising-edge of the processor's high-speed clock on the data phase cycle of a write transfer to this register. This output is asynchronously cleared by system reset.

37.3 Memory Map and Registers

The RGPIO module provides a compact 16-byte programming model based at a system memory address of 0x(00)C0_0000 (noted as RGPIO_BASE throughout the chapter). The programming model views are different between reads and writes to enable simplified software for manipulating the RGPIO pins.

Additionally, the RGPIO programming model is defined with a 32-bit organization. The basic size of each program-visible register is 16 bits, but the programming model may be referenced using byte (8-bit), word (16-bit) or longword (32-bit) accesses. Performance is typically maximized using 32-bit accesses.

NOTE

Writes to the two-byte fields at RGPIO_BASE + 0x8 and RGPIO_BASE + 0xC are allowed, but do not affect any program-visible register within the RGPIO module.

RGPIO memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C0_0000	RGPIO Data Direction Register (RGPIO_DIR)	16	R/W	0000h	37.3.1/867
C0_0002	RGPIO Data Register (RGPIO_DATA)	16	R/W	0000h	37.3.2/868
C0_0004	RGPIO Pin Enable Register (RGPIO_ENB)	16	R/W	0000h	37.3.3/869
C0_0006	RGPIO Clear Data Register (RGPIO_CLR)	16	W	Undefined	37.3.4/870
C0_0008	RGPIO Data Direction Register (RGPIO_DIR)	16	R	0000h	37.3.5/870
C0_000A	RGPIO Set Data Register (RGPIO_SET)	16	W	Undefined	37.3.6/871
C0_000C	RGPIO Data Direction Register (RGPIO_DIR)	16	R	0000h	37.3.7/871
C0_000E	RGPIO Toggle Data Register (RGPIO_TOG)	16	W	Undefined	37.3.8/872

37.3.1 RGPIO Data Direction Register (RGPIO_DIR)

The read/write RGPIO_DIR register defines whether a properly-enabled RGPIO pin is configured as an input or output:

- Setting any bit in RGPIO_DIR configures a properly-enabled RGPIO port pin as an output
- Clearing any bit in RGPIO_DIR configures a properly-enabled RGPIO port pin as an input

At reset, all bits in the RGPIO_DIR are cleared.

Address: C0_0000h base + 0h offset = C0_0000h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	DIR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RGPIO_DIR field descriptions

Field	Description
15–0 DIR	Data direction 0 A properly-enabled RGPIO pin is configured as an input. 1 A properly-enabled RGPIO pin is configured as an output.

37.3.2 RGPIO Data Register (RGPIO_DATA)

The read/write RGPIO_DATA register specifies the write data for a properly-enabled RGPIO output pin or the sampled read data value for a properly-enabled pin. An attempted read of the RGPIO_DATA register returns undefined data for disabled pins because the data value depends on the chip-level pin muxing and pad implementation. At reset, all bits in the RGPIO_DATA registers are cleared.

To set bits in the RGPIO_DATA register, directly set the RGPIO_DATA bits or set the corresponding bits in the RGPIO_SET register. To clear bits in the RGPIO_DATA register, directly clear the RGPIO_DATA bits or clear the corresponding bits in the RGPIO_CLR register. Setting a bit in the RGPIO_TOG register inverts (toggles) the state of the corresponding bit in the RGPIO_DATA register.

As shown in Figure 37-1, the *rgpio_data_in* value is registered using the contents of the *rgpio_data* input bus. For situations where the data direction specifies driving the pins from the *rpgio_data_out* register, the *rgpio_data_in* register value is delayed by one cycle following an update of the write data register. This consecutive sequence is an inherent read-after-write data hazard that can occur with consecutive BCHG instructions and any other operand read following an operand write. Application code must be aware that operand reads of the RGPIO_DATA register immediately (on the next cycle) after an operand write to the RGPIO_DATA register may return a stale value. To prevent this consecutive cycle read-after-write sequence and the corresponding stale data value in RGPIO_DATA, insert a "TPF" instruction (opcode = 0x51FC) between the write and read operations to guarantee the updated value is read.

Address: C0_0000h base + 2h offset = C0_0002h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RGPIO_DATA field descriptions

Field	Description
15–0 DATA	RGPIO data 0 A properly-enabled RGPIO output pin is driven with a logic 0, or a properly-enabled RGPIO pin was read as a logic 0. 1 A properly-enabled RGPIO output pin is driven with a logic 1, or a properly-enabled RGPIO pin was read as a logic 1.

37.3.3 RGPIO Pin Enable Register (RGPIO_ENB)

The read/write RGPIO_ENB register configures the corresponding package pin as an RGPIO pin instead of the normal GPIO pin mapped onto the peripheral bus.

At reset, all bits in the RGPIO_ENB register are cleared, disabling the RGPIO functionality.

Address: C0_0000h base + 4h offset = C0_0004h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	ENB															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RGPIO_ENB field descriptions

Field	Description
15–0 ENB	Enable pin for RGPIO 0 The corresponding package pin is configured for use as a normal GPIO pin, not an RGPIO pin. 1 The corresponding package pin is configured for use as an RGPIO pin.

37.3.4 RGPIO Clear Data Register (RGPIO_CLR)

The RGPIO_CLR register provides a mechanism to clear specific bits in the RGPIO_DATA by performing a simple write. Clearing a bit in RGPIO_CLR clears the corresponding bit in the RGPIO_DATA register. Setting it has no effect. The RGPIO_CLR register is write-only; reads of this address return the RGPIO_DATA register.

Address: C0_0000h base + 6h offset = C0_0006h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read																
Write	CLR															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

RGPIO_CLR field descriptions

Field	Description
15–0 CLR	Clear data 0 Clears the corresponding bit in the RGPIO_DATA register. 1 No effect.

37.3.5 RGPIO Data Direction Register (RGPIO_DIR)

Reading this read-only register returns the value of the RGPIO_DIR register at offset 0h.

Address: C0_0000h base + 8h offset = C0_0008h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DIR															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RGPIO_DIR field descriptions

Field	Description
15–0 DIR	Data direction 0 A properly-enabled RGPIO pin is configured as an input. 1 A properly-enabled RGPIO pin is configured as an output.

37.3.6 RGPIO Set Data Register (RGPIO_SET)

The RGPIO_SET register provides a mechanism to set specific bits in the RGPIO_DATA register by performing a simple write. Setting a bit in RGPIO_SET asserts the corresponding bit in the RGPIO_DATA register. Clearing it has no effect. The RGPIO_SET register is write-only; reads of this address return the RGPIO_DATA register.

Address: C0_0000h base + Ah offset = C0_000Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read																
Write	SET															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

RGPIO_SET field descriptions

Field	Description
15–0 SET	Set data 0 No effect. 1 Sets the corresponding bit in the RGPIO_DATA register.

37.3.7 RGPIO Data Direction Register (RGPIO_DIR)

Reading this read-only register returns the value of the RGPIO_DIR register at offset 0h.

Address: C0_0000h base + Ch offset = C0_000Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DIR															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RGPIO_DIR field descriptions

Field	Description
15–0 DIR	Data direction 0 A properly-enabled RGPIO pin is configured as an input. 1 A properly-enabled RGPIO pin is configured as an output.

37.3.8 RGPIO Toggle Data Register (RGPIO_TOG)

The RGPIO_TOG register provides a mechanism to invert (toggle) specific bits in the RGPIO_DATA register by performing a simple write. Setting a bit in RGPIO_TOG inverts the corresponding bit in the RGPIO_DATA register. Clearing it has no effect. The RGPIO_TOG register is write-only; reads of this address return the RGPIO_DATA register.

Address: C0_0000h base + Eh offset = C0_000Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read																
Write	TOG															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

RGPIO_TOG field descriptions

Field	Description
15–0 TOG	Toggle data 0 No effect 1 Inverts the corresponding bit in RGPIO_DATA

37.4 Functional Description

The RGPIO module is a relatively-simple design with its behavior controlled by the program-visible registers defined within its programming model.

The RGPIO module is connected to the processor's local two-stage pipelined bus with the stages of the ColdFire core's operand execution pipeline (OEP) mapped directly onto the bus. This structure allows the processor access to the RGPIO module for single-cycle pipelined reads and writes with a zero wait-state response (as viewed in the system bus data phase stage).

37.5 Initialization Information

The reset state of the RGPIO module disables the entire 16-bit data port. Prior to using the RGPIO port, software typically:

- Defines the contents of the data register (RGPIO_DATA) if the pin is to be an output
- Configures the pin direction in RGPIO_DIR
- Enables the appropriate pins in RGPIO_ENB

37.6 Application Information

This section examines the relative performance of the RGPIO output pins for two simple applications

- The processor executes a loop to toggle an output pin for a specific number of cycles, producing a square-wave output
- The processor transmits a 16-bit message using a three-pin SPI-like interface with a serial clock, serial chip select, and serial data bit.

In both applications, the relative speed of the GPIO output is presented as a function of the location of the output bit (RGPIO versus peripheral bus GPIO).

37.6.1 Application 1: Simple Square-Wave Generation

In this example, several different instruction loops are executed, each generating a square-wave output with a 50% duty cycle. For this analysis, the executed code is mapped into the processor's RAM. This configuration is selected to remove any jitter from the output square wave caused by the limitations defined by the two-cycle flash memory accesses and restrictions on the initiation of a flash access. The following instruction loops were studied:

- BCHG_LOOP — In this loop, a bit change instruction was executed using the GPIO data byte as the operand. This instruction performs a read-modify-write operation and inverts the addressed bit. A pulse counter is decremented until the appropriate number of square-wave pulses have been generated. When using back to back BCHG instructions, insert the TPF instruction between the BCHG instructions.
- SET+CLR_LOOP — For this construct, two store instructions are executed: one to set the GPIO data pin and another to clear it. Single-cycle NOP instructions (the TPF opcode) are included to maintain the 50% duty cycle of the generated square wave. The pulse counter is decremented until the appropriate number of square-wave pulse have been generated.

The square-wave output frequency was measured and the relative performance results are presented in the following table. The relative performance is stated as a fraction of the processor's operating frequency, defined as *f* MHz. The performance of the BCHG loop operating on a GPIO output is selected as the reference.

Table 37-12. Square-Wave Output Performance

Loop	Peripheral Bus-mapped GPIO			RGPIO		
	Sq-Wave Frequency	Frequency @ CPU *f* = 50 MHz	Relative Speed	Sq-Wave Frequency	Frequency @ CPU *f* = 50 MHz	Relative Speed
bchg	(1/24) × *f* MHz	2.083 MHz	1.00x	(1/14) × *f* MHz	3.571 MHz	1.71x
set+clr (+toggle)	(1/12) × *f* MHz	4.167 MHz	2.00x	(1/8) × *f* MHz	6.250 MHz	3.00x

Note

The square-wave frequency is measured from rising-edge to rising-edge, where the output wave has a 50% duty cycle.

37.6.2 Application 2: 16-bit Message Transmission using SPI Protocol

In this second example, a 16-bit message is transmitted using three programmable output pins. The output pins include a serial clock, an active-high chip select, and the serial data bit. The software is configured to sample the serial data bit at the rising-edge of the clock with the data sent in a most-significant to least-significant bit order. The resulting 3-bit output is shown in the following figure.

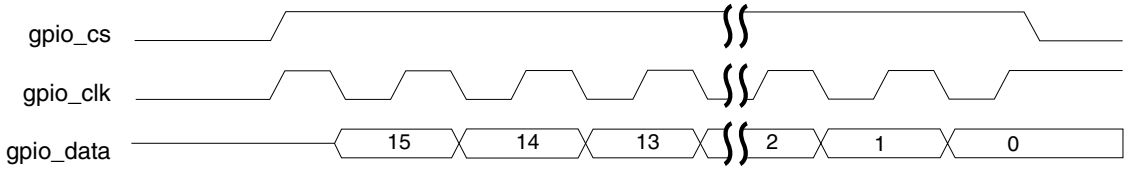


Figure 37-10. GPIO SPI Example Timing Diagram

For this example, the processing of the SPI message is considerably more complex than the generation of a simple square wave of the previous example. The code snippet used to extract the data bit from the message and build the required GPIO data register writes is shown in the following example.

```
# subtest: send a 16-bit message via a SPI interface using a RGPIO
                      # the SPI protocol uses a 3-bit value: clock, chip-select, data
                      # the data is centered around the rising-edge of the clock

                              align   16
                      send_16b_spi_message_rgpio:
00510: 4fef fff4               lea     -12(%sp),%sp          # allocate stack space
00514: 48d7 008c               movm.l  &0x8c,(%sp)           # save d2,d3,d7
00518: 3439 0080 0582          mov.w   RAM_BASE+message2,%d2   # get 16-bit message
0051e: 760f                    movq.l  &15,%d3               # static shift count
00520: 7e10                    movq.l  &16,%d7               # message bit length
00522: 207c 00c0 0003          mov.l   &RGPIO_DATA+1,%a0     # pointer to low-order data byte
00528: 203c 0000 ffff          mov.l   &0xffff,%d0           # data value for _ENB and _DIR regs
0052e: 3140 fffd               mov.w   %d0,-3(%a0)           # set RGPIO_DIR register
00532: 3140 0001               mov.w   %d0,1(%a0)            # set RGPIO_ENB register
00536: 223c 0001 0000          mov.l   &0x10000,%d1          # d1[17:16] = {clk, cs}
0053c: 2001                    mov.l   %d1,%d0               # copy into temp reg
0053e: e6a8                    lsr.l   %d3,%d0               # align in d0[2:0]
00540: 5880                    addq.l  &4,%d0                # set clk = 1
00542: 1080                    mov.b   %d0,(%a0)             # initialize data
00544: 6002                    bra.b   L%1
                               align   4

                      L%1:
00548: 3202                    mov.w   %d2,%d1               # d1[17:15] = {clk, cs, data}
0054a: 2001                    mov.l   %d1,%d0               # copy into temp reg
0054c: e6a8                    lsr.l   %d3,%d0               # align in d0[2:0]
0054e: 1080                    mov.b   %d0,(%a0)             # transmit data with clk = 0
00550: 5880                    addq.l  &4,%d0                # force clk = 1
00552: e38a                    lsl.l   &1,%d2                # d2[15] = new message data bit
00554: 51fc                    tpf                           # preserve 50% duty cycle
00556: 51fc                    tpf
00558: 51fc                    tpf
0055a: 51fc                    tpf
0055c: 1080                    mov.b   %d0,(%a0)             # transmit data with clk = 1
0055e: 5387                    subq.l  &1,%d7                # decrement loop counter
00560: 66e6                    bne.b   L%1

00562: c0bc 0000 fff5          and.l   &0xfff5,%d0           # negate chip-select
00568: 1080                    mov.b   %d0,(%a0)             # update gpio

0056a: 4cd7 008c               movm.l  (%sp),&0x8c           # restore d2,d3,d7
0056e: 4fef 000c               lea     12(%sp),%sp           # deallocate stack space
00572: 4e75                    rts
```

The resulting SPI performance, as measured in the effective Mbps transmission rate for the 16-bit message, is shown in the following table.

Table 37-13. Emulated SPI Performance using GPIO Outputs

Peripheral Bus-mapped GPIO		RGPIO	
SPI Speed @ CPU *f* = 50 MHz	Relative Speed	SPI Speed @ CPU *f* = 50 MHz	Relative Speed
2.063 Mbps	1.00x	3.809 Mbps	1.29x

Chapter 38
Enhanced GPIO (EGPIO)

38.1 Introduction

This section explains software controls related to parallel input/output (I/O) and pin control. Refer to the device's data sheet for more information about pin assignments and external hardware considerations for these pins.

In addition to standard I/O port functionality, some ports have set, clear, and toggle functions that are integrated as part of the ColdFire core itself to improve edge resolution on those pins. For additional information, see the Functional description and the description of Rapid GPIO (RGPIO). Refer to Port Mux details to identify the specific ports that have this additional RGPIO functionality.

Many port pins are shared with on-chip peripherals such as timer systems, communication systems, or keyboard interrupts as shown in Figure 38-1. The peripheral modules have priority over general-purpose I/O functions. When a peripheral is enabled, the I/O functions associated with the shared pins may be disabled.

After reset, the shared peripheral functions are disabled and the pins are configured as inputs (each PTDD*n* bit is 0). The pin control functions for each pin are configured as follows: slew rate control is disabled (each PTSRE*n* bit is 0), low drive strength is selected (each PTDS*n* bit is 0), and internal pulls are disabled (each PTPUE*n* bit is 0).

Note

Not all general-purpose I/O pins are available on all packages. To avoid extra current drain from floating input pins, the user's reset initialization routine in the application program must either enable on-chip pullup devices or change the direction of unconnected pins to outputs, so the pins do not float.

38.2 Overview

The second generation parallel input/output, EGPIO, is a purely digital module (synthesizable for all technologies) present at both sides of the multiplex control module, as shown in the following figure.

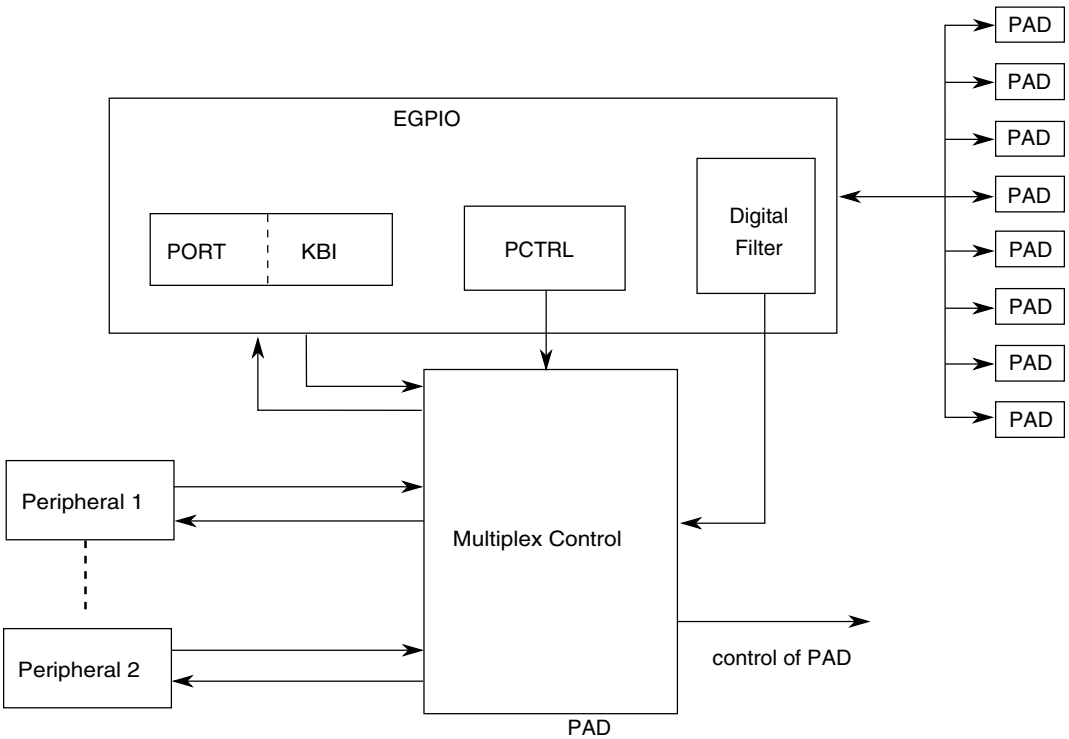


Figure 38-1. Top-level interface of EGPIO

38.2.1 Features

The EGPIO includes four main functions with these features:

- Port data logic
  - Pin input data mapped to, and output data controlled through, port data (D) register
  - Independent directional control per pin through data direction (DD) register
  - Independent pin value (PV) register to read logic level on digital pin
- Independent port control

- Independent internal input pulling resistor enable per pin through pulling enable (PUE) register
- Independent internal input pullup/pulldown select per pin through pullup/ pulldown select (PUS) register
- Independent output slew rate control per pin through slew rate enable (SRE) register
- Independent output drive strength control per pin through drive strength control (DS) register
- Independent input passive filter per pin through PFE register

- Pin interrupt
  - Pin interrupt for each pin with individual flag and individual enable bit
  - Each pin is programmable as falling edge (or rising edge) only, or both falling edge and low level (or both rising edge and high level) interrupt sensitivity.
  - One software-enabled interrupt to CPU
  - One asynchronous interrupt to wake the CPU from low-power mode
  - DMA interface to support transfer by DMA
- Digital filters
  - Digital filter for each pin if configured as input with individual enable bit in DFE register
  - Programmable digital filtering frequency through DFC register

38.2.2 Modes of operation

The section defines the EGPIO operation in wait, stop, and background debug modes.

38.2.2.1 Operation in wait mode

All EGPIO functions continue to operate in wait mode if enabled before execution of the WAIT instruction. If a pin DMA request is not enabled (the PTDMAEN bit is 0), an enabled interrupt pin (the corresponding PTIPEn bit is 1) can be used to bring the MCU out of wait mode if the interrupt of EGPIO is enabled (the PTIE bit is 1).

38.2.2.2 Operation in stop mode

The pin interrupt function operates asynchronously in stop mode if enabled before execution of the STOP instruction. Therefore, an enabled interrupt pin (the corresponding PTIPEn bit is 1) can be used to bring the MCU out of stop mode if the interrupt of EGPIO is enabled (the PTIE bit is 1) and either the digital filter on this pin is disabled (bypass mode) or the digital filter continues to operate on the LPO clock.

The digital filter continues to operate if it is enabled and the LPO clock is selected for digital filtering before entry to stop mode.

38.2.2.3 Operation in active background mode

When the MCU is in active background mode, all EGPIO functions continue to operate normally.

38.3 Memory Map and Registers

NOTE

Port D has only 5 pins available. As a result, in port D's PUE register, you can program only bits 4-0; reading bits 7-5 returns 0, and writing data to bits 7-5 has no effect. The reset value of port D's PUE register is 1Fh.

PCTL memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_8620	Port Pulling Enable Register (PCTLA_PUE)	8	R/W	00h	38.3.1/882
FFFF_8621	Port Pullup/Pulldown Select Register (PCTLA_PUS)	8	R/W	00h	38.3.2/882
FFFF_8622	Port Drive Strength Enable Register (PCTLA_DS)	8	R/W	00h	38.3.3/883
FFFF_8623	Port Slew Rate Enable Register (PCTLA_SRE)	8	R/W	00h	38.3.4/883
FFFF_8624	Port Passive Filter Enable Register (PCTLA_PFE)	8	R/W	FFh	38.3.5/884
FFFF_8625	Port Interrupt Control Register (PCTLA_IC)	8	R/W	00h	38.3.6/884
FFFF_8626	Port Interrupt Pin Enable Register (PCTLA_IPE)	8	R/W	00h	38.3.7/885
FFFF_8627	Port Interrupt Flag Register (PCTLA_IF)	8	R/W	00h	38.3.8/885
FFFF_8628	Interrupt Edge Select Register (PCTLA_IES)	8	R/W	00h	38.3.9/886

Table continues on the next page...

PCTL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_8629	Port Digital Filter Enable Register (PCTLA_DFE)	8	R/W	00h	38.3.10/ 886
FFFF_862A	Port Digital Filter Control Register (PCTLA_DFC)	8	R/W	00h	38.3.11/ 887
FFFF_8630	Port Pulling Enable Register (PCTLB_PUE)	8	R/W	00h	38.3.1/882
FFFF_8631	Port Pullup/Pulldown Select Register (PCTLB_PUS)	8	R/W	00h	38.3.2/882
FFFF_8632	Port Drive Strength Enable Register (PCTLB_DS)	8	R/W	00h	38.3.3/883
FFFF_8633	Port Slew Rate Enable Register (PCTLB_SRE)	8	R/W	00h	38.3.4/883
FFFF_8634	Port Passive Filter Enable Register (PCTLB_PFE)	8	R/W	FFh	38.3.5/884
FFFF_8635	Port Interrupt Control Register (PCTLB_IC)	8	R/W	00h	38.3.6/884
FFFF_8636	Port Interrupt Pin Enable Register (PCTLB_IPE)	8	R/W	00h	38.3.7/885
FFFF_8637	Port Interrupt Flag Register (PCTLB_IF)	8	R/W	00h	38.3.8/885
FFFF_8638	Interrupt Edge Select Register (PCTLB_IES)	8	R/W	00h	38.3.9/886
FFFF_8639	Port Digital Filter Enable Register (PCTLB_DFE)	8	R/W	00h	38.3.10/ 886
FFFF_863A	Port Digital Filter Control Register (PCTLB_DFC)	8	R/W	00h	38.3.11/ 887
FFFF_8640	Port Pulling Enable Register (PCTLC_PUE)	8	R/W	00h	38.3.1/882
FFFF_8641	Port Pullup/Pulldown Select Register (PCTLC_PUS)	8	R/W	00h	38.3.2/882
FFFF_8642	Port Drive Strength Enable Register (PCTLC_DS)	8	R/W	00h	38.3.3/883
FFFF_8643	Port Slew Rate Enable Register (PCTLC_SRE)	8	R/W	00h	38.3.4/883
FFFF_8644	Port Passive Filter Enable Register (PCTLC_PFE)	8	R/W	FFh	38.3.5/884
FFFF_8645	Port Interrupt Control Register (PCTLC_IC)	8	R/W	00h	38.3.6/884
FFFF_8646	Port Interrupt Pin Enable Register (PCTLC_IPE)	8	R/W	00h	38.3.7/885
FFFF_8647	Port Interrupt Flag Register (PCTLC_IF)	8	R/W	00h	38.3.8/885
FFFF_8648	Interrupt Edge Select Register (PCTLC_IES)	8	R/W	00h	38.3.9/886
FFFF_8649	Port Digital Filter Enable Register (PCTLC_DFE)	8	R/W	00h	38.3.10/ 886
FFFF_864A	Port Digital Filter Control Register (PCTLC_DFC)	8	R/W	00h	38.3.11/ 887
FFFF_8650	Port Pulling Enable Register (PCTLD_PUE)	8	R/W	00h	38.3.1/882
FFFF_8651	Port Pullup/Pulldown Select Register (PCTLD_PUS)	8	R/W	00h	38.3.2/882
FFFF_8652	Port Drive Strength Enable Register (PCTLD_DS)	8	R/W	00h	38.3.3/883
FFFF_8653	Port Slew Rate Enable Register (PCTLD_SRE)	8	R/W	00h	38.3.4/883
FFFF_8654	Port Passive Filter Enable Register (PCTLD_PFE)	8	R/W	FFh	38.3.5/884
FFFF_8655	Port Interrupt Control Register (PCTLD_IC)	8	R/W	00h	38.3.6/884
FFFF_8656	Port Interrupt Pin Enable Register (PCTLD_IPE)	8	R/W	00h	38.3.7/885
FFFF_8657	Port Interrupt Flag Register (PCTLD_IF)	8	R/W	00h	38.3.8/885
FFFF_8658	Interrupt Edge Select Register (PCTLD_IES)	8	R/W	00h	38.3.9/886

Table continues on the next page...

PCTL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_8659	Port Digital Filter Enable Register (PCTLD_DFE)	8	R/W	00h	38.3.10/ 886
FFFF_865A	Port Digital Filter Control Register (PCTLD_DFC)	8	R/W	00h	38.3.11/ 887

38.3.1 Port Pulling Enable Register (PCTLx_PUE)

Address: Base address + 0h offset

Bit	7	6	5	4	3	2	1	0
Read Write	PTPUE[7:0]							
Reset	0	0	0	0	0	0	0	0

PCTLx_PUE field descriptions

Field	Description
7–0 PTPUE[7:0]	Port internal pulling enable bits Each pin has a pullup and pulldown resistor associated with it. For port pins that are not configured as inputs, these bits have no effect and the internal pull resistors are disabled. If there is no special note or description for the module that controls the pin whether a pulling resistor is enabled for the module, it is decided by the associated PUE bit. 0 Pulling resistor is disabled. 1 Pulling resistor is enabled.

38.3.2 Port Pullup/Pulldown Select Register (PCTLx_PUS)

Address: Base address + 1h offset

Bit	7	6	5	4	3	2	1	0
Read Write	PTPUS[7:0]							
Reset	0	0	0	0	0	0	0	0

PCTLx_PUS field descriptions

Field	Description
7–0 PTPUS[7:0]	Port pullup/pulldown select bits Each bit selects the pullup or pulldown resistor enabled by the corresponding PUE bit. For port pins that are not configured as inputs, these bits have no effect. If there is no special note or description for the module that controls the pin, the selection of pullup/ pulldown is decided by the associated PUS bit.

PCTLx_PUS field descriptions (continued)

Field	Description
	0 Pulldown resistor is selected. 1 Pullup resistor is selected.

38.3.3 Port Drive Strength Enable Register (PCTLx_DS)

Address: Base address + 2h offset

Bit	7	6	5	4	3	2	1	0
Read Write	PTDS[7:0]							
Reset	0	0	0	0	0	0	0	0

PCTLx_DS field descriptions

Field	Description
7–0 PTDS[7:0]	Port output drive strength control bits Each of these control bits selects between low and high output drive for the associated port pin. For port pins that are configured as inputs, these bits have no effect. 0 Low output drive strength selected. 1 High output drive strength selected.

38.3.4 Port Slew Rate Enable Register (PCTLx_SRE)

Address: Base address + 3h offset

Bit	7	6	5	4	3	2	1	0
Read Write	PTSRE[7:0]							
Reset	0	0	0	0	0	0	0	0

PCTLx_SRE field descriptions

Field	Description
7–0 PTSRE[7:0]	Port output slow rate enable bits Each of these control bits determines whether the output slew rate control is enabled for the associated port pin. For port pins that are not configured as outputs, these bits have no effect. 0 Slew rate control disabled. 1 Slew rate control enabled.

38.3.5 Port Passive Filter Enable Register (PCTLx_PFE)

The PFE register enables control of the input low-pass filter on the pad. The filter is enabled by setting the bit corresponding to a given pin. When set high, a low pass filter (10 MHz to 30 MHz bandwidth) is enabled in the logic input path. When set low, the filter is bypassed.

The filter is enabled during and after reset by setting the associated PTPFE bit. The filter is disabled through software control by clearing the associated PTPFE bit.

Address: Base address + 4h offset

Bit	7	6	5	4	3	2	1	0
Read Write	PTPFE[7:0]							
Reset	1	1	1	1	1	1	1	1

PCTLx_PFE field descriptions

Field	Description
7–0 PTPFE[7:0]	Port passive input filter enable bits These bits enable control of input low-pass filters for port pins. For port pins not configured as inputs, these bits have no effect. 0 Input low-pass filter on pad disabled. 1 Input low-pass filter on pad enabled.

38.3.6 Port Interrupt Control Register (PCTLx_IC)

Address: Base address + 5h offset

Bit	7	6	5	4	3	2	1	0
Read Write	PTDMAEN	0					PTIE	PTMOD
Reset	0	0	0	0	0	0	0	0

PCTLx_IC field descriptions

Field	Description
7 PTDMAEN	DMA enable Determines whether the pin DMA request is enabled. If it is enabled, the pin DMA request is asserted when an interrupt flag of any pin is set (at least one bit of the IF register is set). Meanwhile, a synchronous interrupt from the IF register is disabled if this bit is set. 0 Pin DMA request is disabled and synchronous interrupt from IF is allowed. 1 Pin DMA request is enabled and synchronous interrupt from IF is disabled.

Table continues on the next page...

PCTLx_IC field descriptions (continued)

Field	Description
6–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 PTIE	Interrupt enable Determines whether a pin interrupt is requested to the CPU. 0 Pin interrupt request not enabled. 1 Pin interrupt request enabled.
0 PTMOD	Direction mode for pin interrupt This bit (along with the EDG bits) controls the detection mode of the pin interrupt for pins. 0 Pin interrupt detects edges only. 1 Pin interrupt detects both edges and levels.

38.3.7 Port Interrupt Pin Enable Register (PCTLx_IPE)

This register's bits enable control of pin interrupts.

Address: Base address + 6h offset

Bit	7	6	5	4	3	2	1	0
Read Write	PTIPE[7:0]							
Reset	0	0	0	0	0	0	0	0

PCTLx_IPE field descriptions

Field	Description
7–0 PTIPE[7:0]	Interrupt pin enables Each PTIPE bit enables the corresponding pin for a pin interrupt. **NOTE:** For PCTLA_IPE , bit 0 and bit 4 should be written as zero only . **NOTE:** For PCTLC_IPE , bit 6 should be wriiten as zero only . 0 Pin not enabled for pin interrupt. 1 Pin enabled for pin interrupt.

38.3.8 Port Interrupt Flag Register (PCTLx_IF)

This register contains the interrupt flag bits for pin interrupts.

Address: Base address + 7h offset

Bit	7	6	5	4	3	2	1	0
Read Write	PTIF[7:0]							
Reset	0	0	0	0	0	0	0	0

PCTLx_IF field descriptions

Field	Description
7–0 PTIF[7:0]	Interrupt flags Indicate whether a pin interrupt condition is detected on each input pin if the interrupt is enabled by the associated PTIPE bit. Writing 1 to one bit clears the associated PTIF bit if it is set. 0 No condition of pin interrupt detected. 1 Condition of pin interrupt detected.

38.3.9 Interrupt Edge Select Register (PCTLx_IES)

This register contains the edge select control bits.

Address: Base address + 8h offset

Bit	7	6	5	4	3	2	1	0
Read Write	PTEDG[7:0]							
Reset	0	0	0	0	0	0	0	0

PCTLx_IES field descriptions

Field	Description
7–0 PTEDG[7:0]	Edge selects of pin interrupt Each EDG bit selects the falling edge/low level or rising edge/high level function of the corresponding pin. 0 Falling edge/low level. 1 Rising edge/high level.

38.3.10 Port Digital Filter Enable Register (PCTLx_DFE)

This register contains the enable control bits for digital filters.

NOTE

Only ports B and C have this register.

Address: Base address + 9h offset

Bit	7	6	5	4	3	2	1	0
Read Write	PTDFE[7:0]							
Reset	0	0	0	0	0	0	0	0

PCTLx_DFE field descriptions

Field	Description
7–0 PTDFE[7:0]	Digital filter enables

PCTLx_DFE field descriptions (continued)

Field	Description
	Each PTDFE bit enables the digital filter on the pin when the pin is configured as an input. If the pin is not configured as an input, the digital filter circuit is not used. When enabled, the digital filter is included in the signal path to EGPIO and any module that gets control of the pin and configures it as an input. 0 Digital filter disabled (bypass mode). 1 Digital filter enabled.

38.3.11 Port Digital Filter Control Register (PCTLx_DFC)

This register contains the control bits to select clock and filter factors for all digital filters of the port.

NOTE

Only ports B and C have this register.

Address: Base address + Ah offset

Bit	7	6	5	4	3	2	1	0
Read	PTCLKS	0			PTFF[4:0]			
Write								
Reset	0	0	0	0	0	0	0	0

PCTLx_DFC field descriptions

Field	Description
7 PTCLKS	Clock select bit Selects the counting clock for digital filters. 0 Digital filters count on the bus clock. 1 Digital filters count on the LPO clock.
6–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–0 PTFF[4:0]	Filter factor bits Controls the width of the glitch (in terms of clock cycles) the filter should absorb; glitches less than or equal to this width setting are not allowed by the filter to pass. 00000 1 clock cycle 00001 2 clock cycles 00010 3 clock cycles 00011 4 clock cycles 00100 5 clock cycles 00101 6 clock cycles 00110 7 clock cycles 00111 8 clock cycles . . . . . . 11000 25 clock cycles

Table continues on the next page...

PCTLx_DFC field descriptions (continued)

Field	Description
	11001 26 clock cycles 11010 27 clock cycles 11011 28 clock cycles 11100 29 clock cycles 11101 30 clock cycles 11110 31 clock cycles 11111 32 clock cycles

38.4 Functional description

This section provides full descriptions for all EGPIO functions.

38.4.1 Port data logic

The following figure illustrates port data logic.

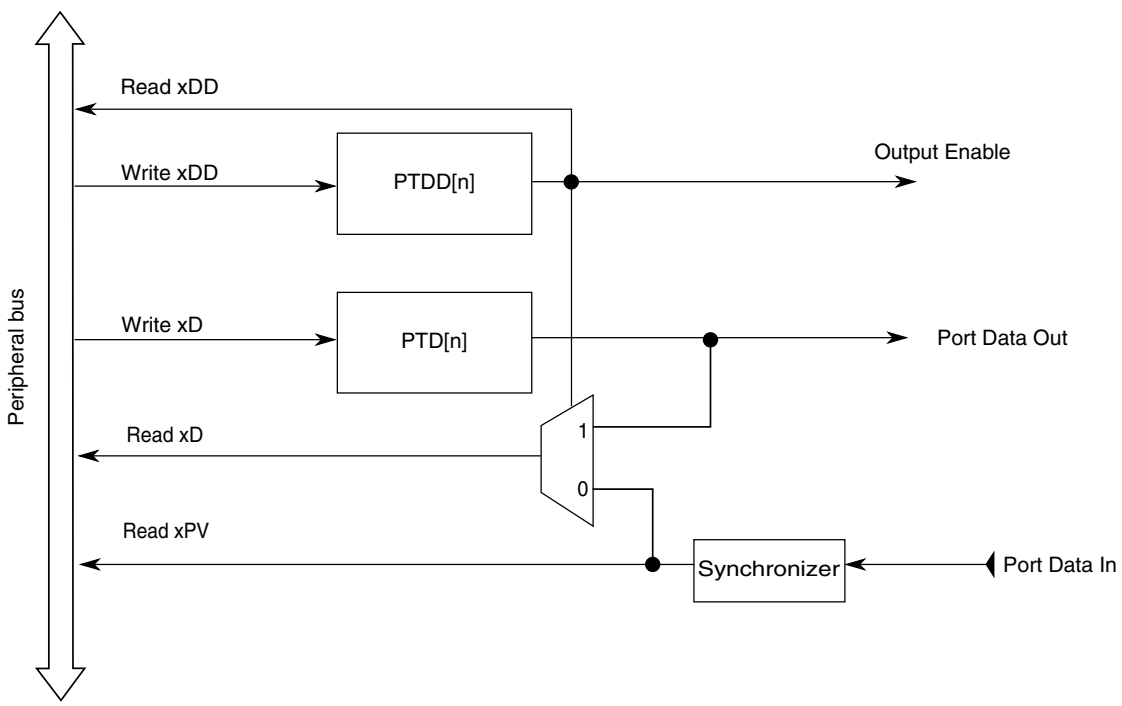


Figure 38-57. Diagram of port data logic function

Port data logic applies to the general purpose input/output function, which is the default function for a port pin when no shared on-chip module is enabled and the pin interrupt function is disabled.

Each port has a data register, a data direction register, and a pin value register. Besides the EGPIO pin interrupt function, an I/O port bit may or may not share an I/O pin with another on-chip module. Consult the Signal Multiplexing description for information about pin assignments.

38.4.1.1 Operation when EGPIO controls pin

If the pin interrupt for a pin is not enabled (the PTIPE bit is 0), the direction of the pin depends on the associated PTDD bit. When the PTDD bit is 1, data written to the port data (D) register is driven out to the corresponding pad.

If the pin interrupt for a pin is enabled (the PTIPE bit is 1), the direction of the pin is configured as an input and the associated PTDD bit has no effect. In this case, data written to the port data (D) register is not driven out to the corresponding pad. When the associated PTDD bit is cleared, a read of the port data register returns the logic level of the associated pin. When the PTDD bit is set, a read of the port data register returns the last value written to the associated bit of the port data register.

When a port pin is controlled by EGPIO, the port is configured for digital functionality, so reading the pin value (PV) register always returns the actual logic level on the pads.

38.4.1.2 Operation when another on-chip module controls pin

When a pin is controlled by another on-chip module, the PTDD bit associated with the pin does not affect the input or output direction of the pin. However, when the PTDD bit is set, reading the port data (D) register still returns the last value written to the port data register's associated bit. When the PTDD bit is cleared and the port pin is configured for digital functionality, a read of the port data register still returns the actual logic level on the pad.

38.4.1.3 Pin value register

For port pins that are configured as digital pins, no matter as input or output, pin value register read always reflects logic level on pads. For port pins that are controlled by analog functions, pin value register read returns zeros (off-value).

38.4.2 Port control

The following figure diagrams the PCTRL function.

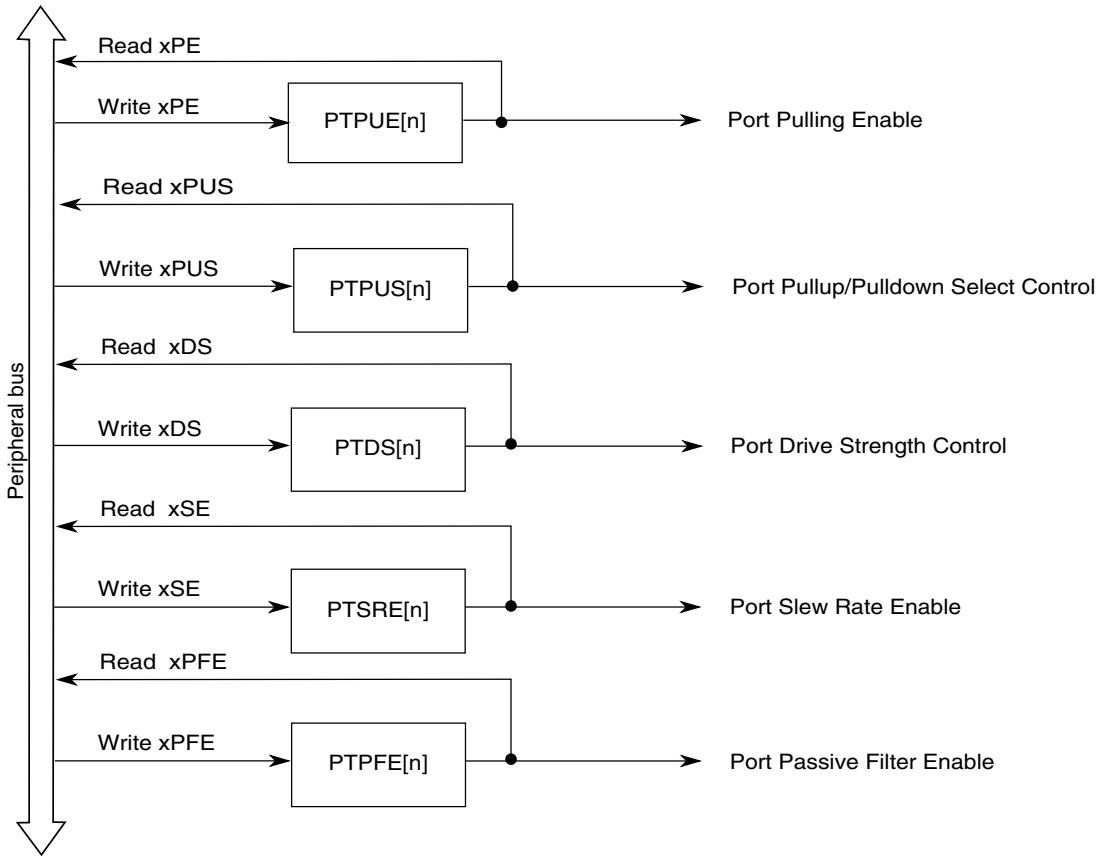


Figure 38-58. Diagram of PCTRL function

The pin control registers are CPU-accessible and operate as described in the Memory map and registers.

38.4.3 Pin interrupt

The pin interrupt function provides up to eight independently enabled external interrupt sources. The following figure is the block diagram for the pin interrupt function.

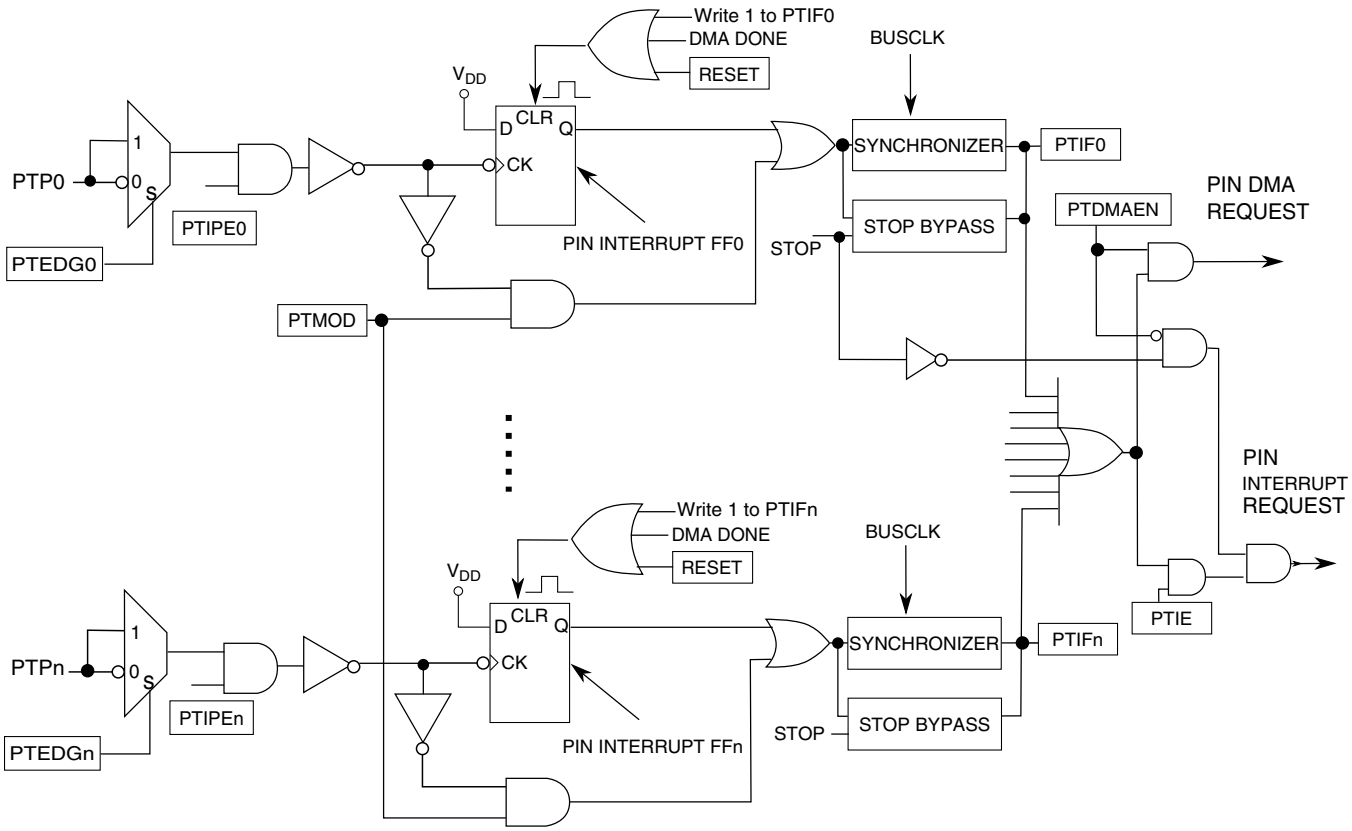


Figure 38-59. Diagram of pin interrupt function

The pin interrupt function is included in EGPIO to supersede an existing KBI module. The main difference is that the pin interrupt function is still able to operate only if a port pin is configured as a digital pin even when EGPIO does not control the pin. That means the pin interrupt, if enabled, can operate with another on-chip module at the same time.

When a port pin is controlled by EGPIO, if the pin interrupt is enabled for this pin (the port interrupt pin enable register's bit is 1), the pin is configured as an input. When a port pin is controlled by the digital function of another on-chip module, the pin interrupt, if enabled (the port interrupt pin enable register's bit is 1), still operates because the EGPIO always receives the actual logic level on external pads when the pin is configured for digital use. But special care must be taken when the pin interrupt operates at the same time with other on-chip modules, and it is strongly recommended to disable the pin interrupt function when another on-chip module controls the pin for output function. When a port pin is configured for analog function, the input for the pin interrupt function is zero and it is recommended to disable the pin interrupt function as well.

38.4.3.1 Edge only sensitivity

A valid edge on an enabled pin interrupt sets an associated PTIF bit. If the PTDMAEN bit in the IC register is set, a DMA request is asserted. If the PTDMAEN bit is not set and the PTIE bit is set, an interrupt request is presented to the CPU. Clearing a PTIF bit is accomplished by writing 1 to the same bit or when the DMA DONE signal for a pin DMA request is asserted.

38.4.3.2 Edge and level sensitivity

A valid edge or level on a pin enabled for pin interrupt sets an associated PTIF bit. If the PTDMAEN bit in the IC register is set, a DMA request is asserted. If the PTDMAEN bit is not set and the PTIE bit is set, an interrupt request is presented to the CPU. Clearing a PTIF bit is accomplished by writing 1 to the same bit or when the DMA DONE signal for a pin DMA request is asserted, provided the associated enabled input of the pin interrupt is at its negated level. During an attempt to clear a PTIF bit, if the associated pin enabled for pin interrupt is asserted, the PTIF bit remains set. Because the DMA process is automatic and negating all enabled inputs before the DMA DONE signal is done is very difficult, it is strongly recommended not to enable a pin DMA request when both edge and level are selected as valid conditions for pin interrupt (the corresponding PTMOD bit is 1).

38.4.3.3 Control of pullup/pulldown resistors

The enabled pin interrupt can be configured to use an internal pullup/pulldown resistor with the associated I/O port pulling enable register. When an internal pulling resistor is enabled (the pin is configured as an input and the PTPUE bit is 1), three situations can apply:

1. If the EGPIO gets control of the pin and the pin is enabled for pin interrupt, the associated PTEDG bit in the IES register selects whether the resistor is a pullup (the PTEDG bit is 0) or a pulldown (the PTEDG bit is 1).
2. If the EGPIO does not get control of the pin or if the pin is not enabled for pin interrupt, the PTEDG bits have no effect.
3. If the EGPIO does not get control of the pin and the pin is still enabled for pin interrupt, the PTEDG bits have no effect on selecting pullup/pulldown resistors. Nevertheless, the value of the associated PTEDG bit must match the pullup/pulldown selection on the pad as defined by the module that gets control of the pin: if the module's actual selection is an internal pullup, the PTEDG bit must be cleared, and if

the module's actual selection is an internal pulldown, the PTEDG bit must be set. Without these aligned settings, when there is no drive on the pad outside, false conditions may be detected as valid conditions for pin interrupt.

38.4.3.4 Asynchronous interrupt in stop mode

When the MCU enters stop mode, the synchronous edge-detection logic is bypassed (because clocks are stopped). In stop mode, enabled inputs of pin interrupt act as asynchronous level-sensitive inputs so they can wake the MCU from stop mode. In stop mode, the pin interrupt requests are not blocked by the PTDMAEN bit being set to 1.

38.4.3.5 Pin interrupt initialization

When an interrupt pin is first enabled or reconfigured, a false interrupt flag can result. To prevent a false interrupt request during pin interrupt initialization, you must perform the following steps:

1. Mask port interrupts by clearing the PTIE bit in the IC register.
2. Enable the polarity for pin interrupt by setting the appropriate PTEDGn bits in the IES register. If the pin interrupt function operates at the same time with another on-chip module that gets control of the pin, the associated PTEDG bit must be set according to the actual selection of internal pullup (the PTEDG bit must be cleared) or pulldown (the PTEDG bit must be set) on the pad by the module that gets control of the pin.
3. If using an internal pullup/pulldown device, configure the associated pulling enable bits in the PUE register.
4. Enable the pins for pin interrupt by setting the appropriate bits in the IPE register.
5. Execute three NOP instructions before the next step to avoid a timing conflict between setting the IPE register and clearing the flag.
6. Write FFh to the IF register to clear any false interrupt flags.
7. Enable the DMA request or interrupt request by configuring the IC register.

38.4.4 Digital filters

The passive input low-pass filter can filter only signals greater than 10 MHz . EGPIO provides programmable digital filters to filter signals much less than 10 MHz for low-speed applications.

The digital filters absorb glitches on digital pins. For the port pin configured as a digital pin, the digital filter is enabled for the pin if the associated bit is set in the port digital filter enable register. For a port pin that is not configured for analog function, the digital filter for the pin is disabled, and the associated bit in the port digital filter enable register has no effect.

The width of the glitch to absorb can be specified in terms of number of filter clock cycles. The bus clock or LPO clock can be selected as a filter clock that is configured by the PTCLKS bit in the port digital filter control register. The width of the glitch to absorb depends on the Filter Factor (FF) bits in the port digital filter control register. Effectively, any down-up-down or up-down-up transition on the digital input line that occurs within the number of clock cycles programmed by the filter factor is ignored by on-chip modules. For details, see the description of the port digital filter control register.

Because the configuration of the port digital filter control register is for all digital filters of the port, changing the port digital filter control register affects all digital filters of the port if enabled. Configuration of the port digital filter control register must occur when no digital filter is active (the port digital filter enable register is 00h).

The LPO clock can operate in stop mode; if the digital filter works in stop mode, the PTCLKS bit must be set before the entry to stop mode.

38.4.4.1 Initialization of digital filters

When a digital filter for a port pin is enabled from a disabled state or pin control is changed from one module to another module while the digital filter is active, some enabled modules may get a false input. To prevent a false input to on-chip modules and related errors during initialization of digital filters, you must do the following to enable the digital filter for input of the target module:

1. Write 0 to the PTDFE bit to disable the digital filter for the pin, if it is active.
2. Write 0 to the PTIPE bit to disable the pin interrupt function, if it is enabled.
3. Disable other on-chip modules that have higher priority for pin control than the target module.
4. Write 1 to the associated pin enable bit of the target module, if it is not enabled.

5. Follow the flow for initialization of the pin interrupt if you still want the pin interrupt function working with the target module.
6. Write 1 to the associated PTDFE bit to enable the digital filter for the pin.

The preceding flow assumes that selection of the filter clock and filter factor do not change and the target module is an on-chip module other than EGPIO.

If you enable the pin interrupt function of EGPIO on a port pin when the digital filter for the pin is active for the input of another on-chip module, you must follow the initialization of the pin interrupt.

If you enable only a digital filter for EGPIO input, you must perform the following steps:

1. Write 0 to the PTDFE bit to disable the digital filter for the pin, if it is active.
2. Disable other on-chip modules that have higher priority for pin control than EGPIO.
3. Follow initialization of the pin interrupt if you enable the pin interrupt function on this port pin.
4. Write 1 to the PTDFE bit to enable the digital filter for the pin.

38.5 Reset

The EGPIO module is reset automatically by any MCU reset.

The EGPIO module cannot cause an MCU reset.

Regarding EGPIO registers:

- The IFE register resets to FFh to disable passive low-pass filters on pads.
- The PV register is unaffected by reset. For details, refer to the register's description.
- A reset clears all other EGPIO registers.

Chapter 39
EGPIO Port Control

39.1 Introduction

Registers for EGPIO control are in two groups. Registers for general port (pin) control are described here. Refer to Introduction for more information about both sets of registers and about parallel I/O control in general.

39.2 Memory Map and Registers

PT memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFFF_8000	Port Data Register (PTA_D)	8	R/W	See section	39.2.1/898
FFFF_8001	Port Data Direction Register (PTA_DD)	8	R/W	00h	39.2.2/898
FFFF_8002	Port Pin Value Register (PTA_PV)	8	R	Undefined	39.2.3/899
FFFF_8010	Port Data Register (PTB_D)	8	R/W	See section	39.2.1/898
FFFF_8011	Port Data Direction Register (PTB_DD)	8	R/W	00h	39.2.2/898
FFFF_8012	Port Pin Value Register (PTB_PV)	8	R	Undefined	39.2.3/899
FFFF_8020	Port Data Register (PTC_D)	8	R/W	See section	39.2.1/898
FFFF_8021	Port Data Direction Register (PTC_DD)	8	R/W	00h	39.2.2/898
FFFF_8022	Port Pin Value Register (PTC_PV)	8	R	Undefined	39.2.3/899
FFFF_8030	Port Data Register (PTD_D)	8	R/W	See section	39.2.1/898
FFFF_8031	Port Data Direction Register (PTD_DD)	8	R/W	00h	39.2.2/898
FFFF_8032	Port Pin Value Register (PTD_PV)	8	R	Undefined	39.2.3/899

39.2.1 Port Data Register (PTx_D)

Address: Base address + 0h offset

Bit	7	6	5	4	3	2	1	0
Read Write	PTD[7:0]							
Reset	*	*	*	*	*	*	*	*

* Notes:
- PTD[7:0] field: The reset values of internal registers for D are zeros. However, a read of the D register after reset returns the actual logic level on external pins.

PTx_D field descriptions

Field	Description
7–0 PTD[7:0]	Port data bits Writes are latched into all bits of this register. When port data direction bits for port pins are set (each DD bit is 1), reads return the last value written to this register. When a port pin is controlled by EGPIO with the pin interrupt function disabled and the associated port data direction bit is set (the DD bit is 1), the logic level is driven out to the corresponding MCU pin. When the port data direction bits for port pins are cleared (each DD bit is 0): • For pins that are configured as digital pins, reads return the logic level on the pin. • For port pins that are controlled by analog functions, reads return zeros (off value).

39.2.2 Port Data Direction Register (PTx_DD)

Address: Base address + 1h offset

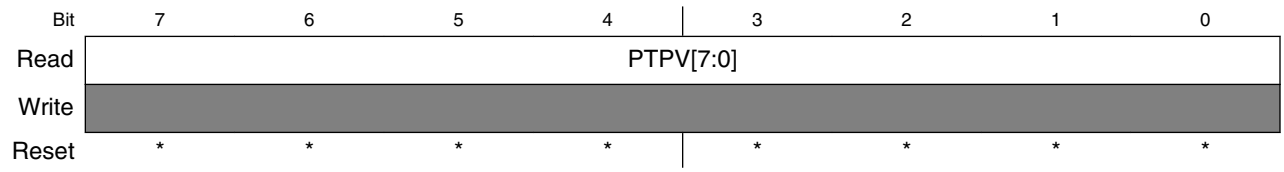
Bit	7	6	5	4	3	2	1	0
Read Write	PTDD[7:0]							
Reset	0	0	0	0	0	0	0	0

PTx_DD field descriptions

Field	Description
7–0 PTDD[7:0]	Port data direction bits Each bit of the port data direction register controls whether an associated port pin is an input or output when pin interrupt is disabled and no other module controls the pin. If the DD bit for a port pin is equal to logic one, and Port Data Logic has control of the pin, the port pin is defined as output and the logic value of an internal register for the D register is driven out to the corresponding MCU pin. 0 The port pin is defined only as an input. 1 The port pin is defined as an output.

39.2.3 Port Pin Value Register (PTx_PV)

Address: Base address + 2h offset



* Notes:
- x = Undefined at reset.
- A read of the PV register after reset returns the actual logic level on external pins because reset configures all port pins as high-impedance inputs with pullup/pulldown disabled.x = Undefined at reset.

PTx_PV field descriptions

Field	Description
7–0 PTPV[7:0]	Port pin value bits Each bit of port pin value register is mapped to one MCU pin. For pins that are configured as digital pins, this register always reflects the digital level on the actual PAD. For pins that are controlled by analog functions, reads return zeros (off-value).

Chapter 40
External Interrupt (IRQ)

40.1 Introduction

The IRQ (External Interrupt) module provides an interrupt input.

40.1.1 Features

The IRQ includes these distinctive features:

- IP Bus V2.0 compliant
- External interrupt pin (IRQ)
- IRQ pin can be selected as falling edge and low level or rising edge and high level
- Separate IRQ pin enable
- Software enabled interrupt
- Programmable falling edge (or rising edge) only, or both falling edge and low level (or both rising edge and high level) interrupt sensitivity
- Exit from low-power modes
- Software control of whether on-chip pullup/pulldown device is enabled on IRQ pin

40.1.2 Modes of Operation

The IRQ module is mode independent and will continue to operate in all user modes. In the low power STOP mode, the IRQ input becomes an asynchronous path.

40.1.3 Block Diagram

The following is a block diagram of the IRQ module.

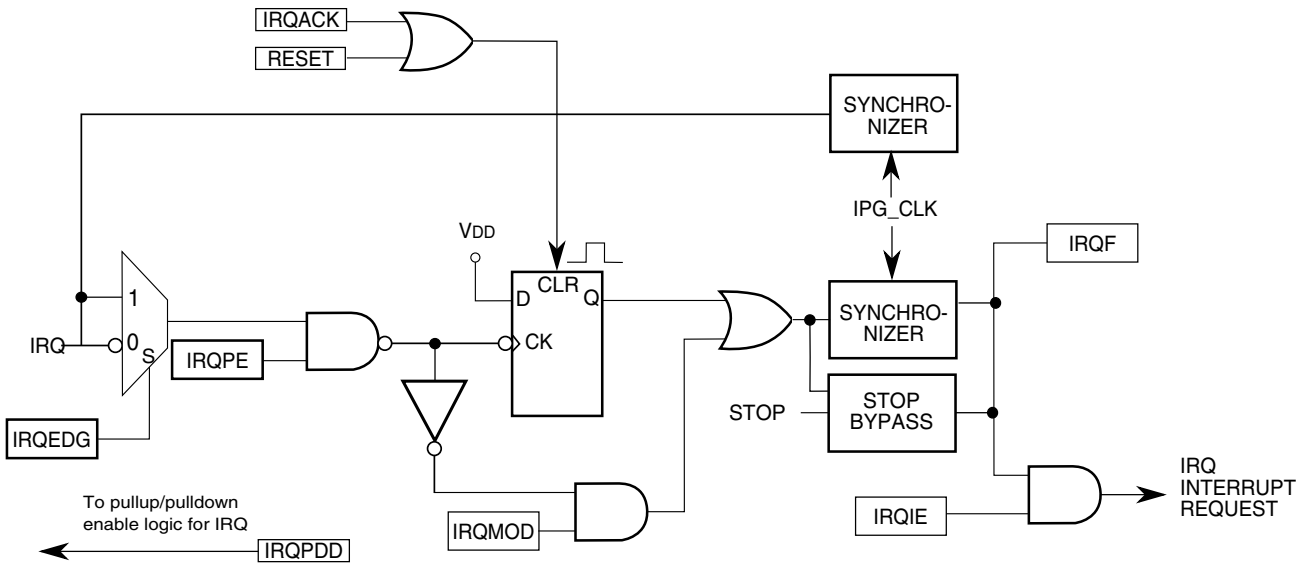


Figure 40-1. IRQ Block Diagram

40.2 Signal Description

The following table shows the user-accessible signal for the IRQ module.

Table 40-1. Signal Properties

Name	Function	Reset State
IRQ	External interrupt pin	input

40.2.1 Detailed Signal Descriptions

This section describes each user-accessible pin.

1. IRQ — External interrupt input pin

   This input pin is used to detect either falling edge, or both falling edge and low level interrupt requests. This input pin can also be used to detect either rising edge, or both rising edge and high level interrupt requests.

40.3 Memory Map and Register Description

This section provides a detailed description of the IRQ register that is accessible to the end user.

IRQ memory map

Address offset (hex)	Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	FFFF_8800	Interrupt status and control register (IRQ0_SC)	8	R/W	00h	40.3.1/903
0	FFFF_8810	Interrupt status and control register (IRQ1_SC)	8	R/W	00h	40.3.1/903
0	FFFF_8820	Interrupt status and control register (IRQ2_SC)	8	R/W	00h	40.3.1/903
0	FFFF_8830	Interrupt status and control register (IRQ3_SC)	8	R/W	00h	40.3.1/903

40.3.1 Interrupt status and control register (IRQx_SC)

Address: Base address + 0h offset

Bit	7	6	5	4	3	2	1	0
Read	0	IRQPDD	IRQEDG	IRQPE	IRQF	0	IRQIE	IRQMOD
Write						IRQACK		
Reset	0	0	0	0	0	0	0	0

IRQx_SC field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 IRQPDD	IRQ pull device disable Use this bit to disable the on-chip pullup/pulldown device on the IRQ pin. This allows users to have an external device if required for their application. 0 On-chip pullup/pulldown device is enabled 1 On-chip pullup/pulldown device is disabled
5 IRQEDG	IRQ edge select This bit selects the falling edge/low level or rising edge/high level function of the IRQ pin. 0 Falling edge/low level 1 Rising edge/high level
4 IRQPE	IRQ pin enable This bit determines whether the IRQ pin is enabled.

Table continues on the next page...

IRQx_SC field descriptions (continued)

Field	Description
	0 IRQ pin not enabled 1 IRQ pin enabled
3 IRQF	IRQ flag This bit indicates when an IRQ interrupt is detected. 0 No IRQ interrupt detected 1 IRQ interrupt detected
2 IRQACK	IRQ acknowledge Writing 1 to this bit is part of the flag clearing mechanism. For more information about flag clearing, see Clearing an IRQ Interrupt Request. This bit always reads as 0.
1 IRQIE	IRQ interrupt enable This bit determines whether an IRQ interrupt request is enabled. 0 IRQ interrupt requests not enabled 1 IRQ interrupt requests enabled
0 IRQMOD	IRQ detection mode This bit (along with the IRQEDG bit) controls the detection mode of the IRQ pin. 0 IRQ interrupt requests on falling edge only or on rising edge only 1 IRQ interrupt requests on falling edge and low level or on rising edge and high levels

40.4 Functional Description

This section provides a complete functional description of the IRQ module.

40.4.1 External Interrupt Pin

Writing to the IRQPE bit in the SC register, enables or disables the IRQ pin.

40.4.2 IRQ Edge Select

The IRQEDG bit in the SC register determines if the IRQ pin is either falling edge and low level or rising edge and high level sensitive.

40.4.3 IRQ Sensitivity

The IRQMOD bit in the SC register controls the detection mode of the IRQ module.

- If the IRQ interrupt is falling (or rising) edge sensitive only, a falling (or rising) edge on the enabled IRQ pin will set the IRQF bit.
- If the IRQ interrupt is both falling (or rising) edge and low (or high) level sensitive, a falling (or rising) edge on the enabled IRQ sets the IRQF bit. The IRQF bit remains set as long as the IRQ pin remains asserted.

40.4.4 IRQ Interrupts

The IRQ module can provide a source of interrupts. To cause an IRQ module interrupt request, the following must occur:

- The IRQIE bit in the SC register must be set.
- The IRQF bit in the SC register must become set by a triggered IRQ pin. The IRQF bit becomes set by the fifth clock cycle after the IRQ pin has become asserted.
- The IRQ pin must have been in an inactive state for at least one clock cycle before becoming active.
- Changing the IRQMOD or IRQEDG bit while the IRQPE bit is enabled may cause a spurious interrupt and the IRQF bit may be inadvertently cleared.

40.4.5 Clearing an IRQ Interrupt Request

If the IRQ module interrupt pin is either falling edge and low level sensitive or rising edge and high level sensitive, both of the following actions must occur to clear an IRQ interrupt request:

- Software provides an interrupt acknowledge by writing a logic 1 to the IRQACK bit in the SC register.
- And either of the following:
  - The IRQ pin returns to a deasserted logic state.
  - The IRQ pin is disabled using the IRQPE bit.

If the IRQ module interrupt pin is falling (or rising) edge sensitive only, writing a logic 1 to the IRQACK bit in the SC register immediately clears the IRQ interrupt request even if the enabled IRQ pin remains asserted.

40.4.6 Exit from Low-Power Modes

The IRQ interrupt, if enabled, can provide a means to exit CPU low-power modes. If the IRQ pin is enabled and asserted upon entering a low-power mode and the detection mode is set to both falling edge and low level sensitivity or both rising edge and high level sensitivity, an immediate exit from the low-power mode may occur depending on the specific chip implementation. If the detection mode is set to falling or rising edge sensitivity only, an edge must be seen on the enabled IRQ pin to exit the low-power mode.

40.4.6.1 Wait

The IRQ module remains active in WAIT mode. Setting the IRQIE bit in the SC register enables the IRQ interrupt request. Any detected IRQ interrupt will bring the CPU out of WAIT mode.

40.4.6.2 Stop modes

The IRQ module can remain active in stop modes, depending on the chip implementation. Setting the IRQIE bit in the SC register enables the IRQ interrupt request. Any detected IRQ interrupt brings the CPU out of the stop mode.

40.5 Resets

The IRQ interrupt is disabled after reset. The IRQ module cannot cause an MCU reset.

40.6 Interrupts

The IRQ module generates a single interrupt.

The IRQ interrupt is listed in the following table, which shows the interrupt name and the name of the local enable that can be used to disable a IRQ interrupt request.

Table 40-12. Interrupt Summary

Interrupt	Local Enable	Source	Description
IRQF	IRQIE	IRQ input	Software programmable for falling edge only (or rising edge only), or both falling edge and low level detection (or both rising edge and high level detection).

Chapter 41
Debug

41.1 Introduction

This chapter describes the capabilities defined by the Version 1 ColdFire debug architecture. The Version 1 ColdFire core supports BDM functionality using the HCS08's single-pin interface. The traditional 3-pin full-duplex ColdFire BDM serial communication protocol based on 17-bit data packets is replaced with the HCS08 debug protocol where all communication is based on an 8-bit data packet using a single package pin (BKGD).

An on-chip trace buffer allows a stream of compressed processor execution status packets to be recorded for subsequent retrieval to provide program (and partial data) trace capabilities.

The following sections in this chapter provide details on the BKGD pin, the background debug serial interface controller (BDC), a standard 6-pin BDM connector, the BDM command set as well as real-time debug and trace capabilities. The V1 definition supports revision B+ (DEBUG_B+) of the ColdFire debug architecture.

A simplified block diagram of the V1 core including the processor and debug module is shown in the following figure.

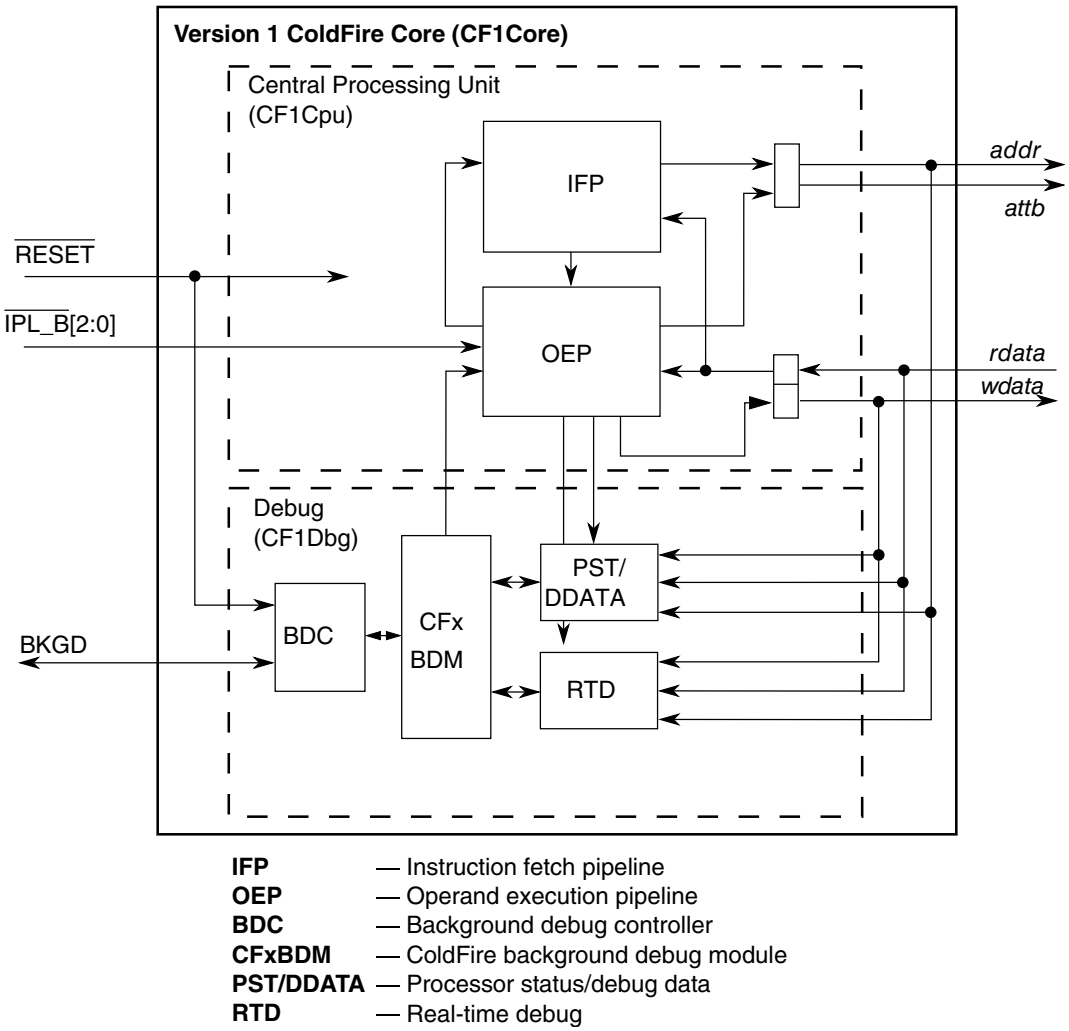


Figure 41-1. Simplified Version 1 ColdFire Core Block Diagram

41.1.1 Overview

Debug support is divided into three areas:

- Background debug mode (BDM)—Provides low-level debugging in the ColdFire processor core. In BDM, the processor core is halted and a variety of commands can be sent to the processor to access memory, registers, and peripherals. The external emulator uses a one-pin serial communication protocol. See Background Debug Mode (BDM).
- Real-time debug support—Use of the full BDM command set requires the processor to be halted, which many real-time embedded applications cannot support. The core includes a variety of internal breakpoint registers which can be configured to trigger and generate a special interrupt. The resulting debug interrupt lets real-time systems execute a unique service routine that can quickly save the contents of key registers

and variables and return the system to normal operation. The external development system can then access the saved data, because the hardware supports concurrent operation of the processor and BDM-initiated memory commands. In addition, the option is provided to allow interrupts to occur. See Real-Time Debug Support.

- Program trace support—The ability to determine the dynamic execution path through an application is fundamental for debugging. The V1 solution implements a trace buffer that records processor execution status and data, which can be subsequently accessed by the external emulator system to provide program (and optional partial data) trace information. See Trace Support .

There are two fields in debug registers which provide revision information: the hardware revision level in CSR and the 1-pin debug hardware revision level in CSR2. The following table summarizes the various debug revisions.

Table 41-1. Debug Revision Summary

Revision	CSR[HRL]	CSR2[D1HRL]	Enhancements
A	0000	N/A	Initial ColdFire debug definition
B	0001	N/A	BDM command execution does not affect hardware breakpoint logic Added BDM address attribute register (BAAR) $\overline{\text{BKPT}}$ configurable interrupt (CSR[BKD]) Level 1 and level 2 triggers on OR condition, in addition to AND SYNC_PC command to display the processor's current PC
B+	1001	N/A	Added 3 PC breakpoint registers PBR1–3
CF1_B+	1001	0001	Converted to HCS08 1-pin BDM serial interface Added PST compression and on-chip PST/DDATA buffer for program trace
CF1_B +_no_PSTB	1001	0010	Standard CF1 Debug_B+ without the PST trace buffer
CF1_B+ for 90 nm TFS	1001	0011	CF1 debug with DBGCR and DBGSR

41.1.2 Features

The Version 1 ColdFire debug definition supports the following features:

- Classic ColdFire DEBUG_B+ functionality mapped into the single-pin BDM interface
- Real time debug support, with 6 hardware breakpoints (4 PC, 1 address pair and 1 data) that can be configured into a 1- or 2-level trigger with a programmable response (processor halt or interrupt)

- Capture of compressed processor status and debug data into on-chip trace buffer provides program (and optional slave bus data) trace capabilities
- On-chip trace buffer provides programmable start/stop recording conditions plus support for obtrusive or PC-profiling modes
- Debug resources are accessible via single-pin BDM interface or the privileged WDEBUG instruction from the core

41.1.3 Modes of Operation

V1 ColdFire devices typically implement a number of modes of operation, including run, wait, and stop modes. Additionally, the operation of the core's debug module is highly dependent on a number of chip configurations that determine its operating state.

When operating in secure mode, as defined by a 2-bit field in the flash memory examined at reset, BDM access to debug resources is extremely restricted. It is possible to tell that the device has been secured and to clear security, which involves mass erasing the on-chip flash memory. No other debug access is allowed. Secure mode can be used in conjunction with each of the wait and stop low-power modes.

If the BDM interface is not enabled, access to the debug resources is limited in the same manner as a secure device.

If the device is not secure and the BDM interface is enabled (XCSR[ENBDM] is set), the device is operating in debug mode and additional resources are available via the BDM interface. In this mode, the status of the processor (running, stopped, or halted) determines which BDM commands may be used.

Debug mode functions are managed through the background debug controller (BDC) in the Version 1 ColdFire core. The BDC provides the means for analyzing MCU operation during software development.

BDM commands can be classified into three types as shown in the following table.

Table 41-2. BDM Command Types

Command Type	Flash Secure?	BDM?	Core Status	Command Set
Always-available	Secure or Unsecure	Enabled or Disabled	—	• Read/write access to XCSR[31–24], CSR2[31–24], CSR3[31–24]

Table continues on the next page...

Table 41-2. BDM Command Types (continued)

Command Type	Flash Secure?	BDM?	Core Status	Command Set
Non-intrusive	Unsecure	Enabled	Run, Halt	• Memory access • Memory access with status • Debug register access • BACKGROUND
Active background	Unsecure	Enabled	Halt	• Read or write CPU registers (also available in stop mode) • Single-step the application • Exit halt mode to return to the application program (GO)

For more information on these three BDM command classifications, see BDM Command Set Summary.

The core's halt mode is entered in a number of ways:

- The BKGD pin is low during POR
- The BKGD pin is low immediately after a BDM-initiated force reset (see CSR2[BDFR] in Configuration/Status Register 2 (CSR2), for details)
- A background debug force reset occurs (CSR2[BDFR] is set) and CSR2[BFHBR] is set
- An illegal operand reset occurs and CSR2[IOPHR] is set
- An illegal address reset occurs and CSR2[IADHR] is set
- A computer operating properly reset occurs and CSR2[COPHR] is set
- A BACKGROUND command is received through the BKGD pin. If necessary, this wakes the device from STOP/WAIT modes.
- A properly-enabled (XCSR[ENBDM] is set) HALT instruction is executed
- Encountering a BDM breakpoint and the trigger response is programmed to generate a halt
- Reaching a PSTB trace buffer full condition when operating in an obtrusive recording mode (CSR2[PSTBRM] is set to 01 or 11)

While in halt mode, the core waits for serial background commands rather than executing instructions from the application program.

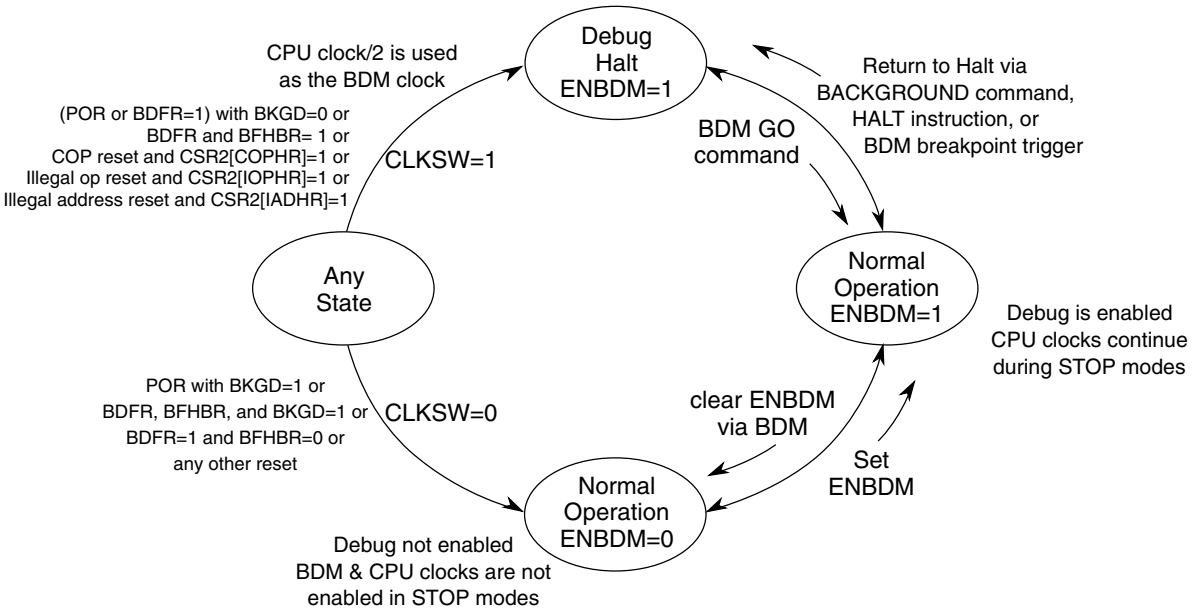


Figure 41-2. Debug Modes State Transition Diagram

The preceding figure contains a simplified view of the V1 ColdFire debug mode states. The XCSR[CLKSW] bit controls the BDC clock source. When CLKSW is set, the BDC serial clock frequency is the same as the synchronous bus clock. When CLKSW is cleared, the BDC serial clock is supplied from an alternate clock source.

The ENBDM bit determines if the device can be placed in halt mode, if the core and BDC serial clocks continue to run in STOP modes, and if the regulator can be placed into standby mode. Again, if booting to halt mode, XCSR[ENBDM, CLKSW] are automatically set.

If ENBDM is cleared, the ColdFire core treats the HALT instruction as an illegal instruction and generates a reset (if CPUCR[IRD] is cleared) or an exception (if CPUCR[IRD] is set) if execution is attempted.

If XCSR[ENBDM] is set, the device can be restarted from STOP/WAIT via the BDM interface.

41.2 External Signal Descriptions

The following table describes the debug module's 1-pin external signal (BKGD). A standard 6-pin debug connector is shown in Freescale-Recommended BDM Pinout.

Table 41-3. Debug Module Signals

Signal	Description
Background Debug (BKGD)	Single-wire background debug interface pin. The primary function of this pin is for bidirectional serial communication of background debug mode commands and data. During reset, this pin selects between starting in active background (halt) mode or starting the application program. This pin also requests a timed sync response pulse to allow a host development tool to determine the correct clock frequency for background debug serial communications.

41.3 Memory Map and Register Descriptions

In addition to the BDM commands that provide access to the processor's registers and the memory subsystem, the debug module contains a number of registers. Most of these registers (all except the PST/DDATA trace buffer) are also accessible (write-only) from the processor's supervisor programming model by executing the WDEBUG instruction. Thus, the breakpoint hardware in the debug module can be read (certain registers) or written by the external development system using the serial debug interface or written by the operating system running on the processor core. Software is responsible for guaranteeing that accesses to these resources are serialized and logically consistent. The hardware provides a locking mechanism in the CSR to allow the external development system to disable any attempted writes by the processor to the breakpoint registers (setting CSR[IPW]). BDM commands must not be issued during the processor's execution of the WDEBUG instruction to configure debug module registers or the resulting behavior is undefined.

These registers are treated as 32-bit quantities regardless of the number of implemented bits. Unimplemented bits are reserved and must be cleared. These registers are also accessed through the BDM port by the commands WRITE_DREG and READ_DREG described in BDM Command Set Summary. These commands contain a 5-bit field, DRc, that specifies the register, as shown in the following table.

Note

Most debug control registers can be written either by the external development system or by the CPU through the WDEBUG instruction. These control registers are write-only from the programming model and they can be written through the BDM port using the WRITE_DREG command. In addition, the four configuration/status registers (CSR, XCSR, CSR2, CSR3) can be read through the BDM port using the READ_DREG command.

The ColdFire debug architecture supports a number of hardware breakpoint registers that can be configured into single- or double-level triggers based on the PC or operand address ranges with an optional inclusion of specific data values. The triggers can be configured to halt the processor or generate a debug interrupt exception. Additionally, these same breakpoint registers can be used to specify start/stop conditions for recording in the PST trace buffer.

The core includes four PC breakpoint triggers and a set of operand address breakpoint triggers with two independent address registers (to allow specification of a range) and an optional data breakpoint with masking capabilities. Core breakpoint triggers are accessible through the serial BDM interface or written through the supervisor programming model using the WDEBUG instruction.

Table 41-4. Debug Module Memory Map

DRc[4:0]	Register	Width (bits)	Access	Reset Value
0x00	Configuration/Status Register (CSR)	32	R/W (BDM), W (CPU)	0x0090_0000
0x01	Extended Configuration/Status Register (XCSR)	32	R/W[1] (BDM), W (CPU)	0x0000_0000
0x02	Configuration/Status Register 2 (CSR2)	32	R/W[1](BDM), W (CPU)	See section
0x03	Configuration/Status Register 3 (CSR3)	32[2]	R/W[1] (BDM), W (CPU)	0x0000_0000
—	Debug Control Register (DBGCR)	32	Indirect W via CSR3 (BDM)	0x0000_0000
—	Debug Status Register (DBGSR)	32	Indirect R via CSR3 (BDM)	0x0000_0000
0x05	BDM Address Attribute Register (BAAR)	32[2]	W	0x0000_0005
0x06	Address Attribute Trigger Register (AATR)	32[2]	W	0x0000_0005
0x07	Trigger Definition Register (TDR)	32	W	0x0000_0000
0x08	Program Counter Breakpoint Register 0 (PBR0)	32	W	Undefined, unaffected[3]
0x09	Program Counter Mask Register (PBMR)	32	W	Undefined, unaffected[3]
0x0C	Address Breakpoint High Register (ABHR)	32	W	Undefined, unaffected[3]
0x0D	Address Breakpoint Low Register (ABLR)	32	W	0x0000_0000
0x0E	Data Breakpoint Register (DBR)	32	W	0x0000_0000
0x0F	Data Breakpoint Mask Register (DBMR)	32	W	0x0000_0000
0x18	Program Counter Breakpoint Register 1 (PBR1)	32	W	PBR1[0] = 0
0x1A	Program Counter Breakpoint Register 2 (PBR2)	32	W	PBR2[0] = 0
0x1B	Program Counter Breakpoint Register 3 (PBR3)	32	W	PBR3[0] = 0
—	PST Trace Buffer*n* (PSTB*n*); *n* = 0–11 (0xB)	32	R (BDM)[4]	Undefined, unaffected[3]

1. The most significant bytes of the XCSR, CSR2, and CSR3 registers support special control functions and are writeable via BDM using the WRITE_XCSR_BYTE, WRITE_CSR2_BYTE, and WRITE_CSR3_BYTE commands. They can be read from BDM using the READ_XCSR_BYTE, READ_CSR2_BYTE, and READ_CSR3_BYTE commands. These 3 registers, along with the CSR, can also be referenced as 32-bit quantities using the BDM READ_DREG and WRITE_DREG commands, but the WRITE_DREG command only writes bits 23–0 of these three registers.
2. Each debug register is accessed as a 32-bit value; undefined fields are reserved and must be cleared.
3. The register's reset value is undefined after POR and unaffected by any other reset type. After POR, any value written to the register is retained.
4. The contents of the PST trace buffer is only read from BDM (32 bits per access) using READ_PSTB commands.

41.3.1 Configuration/Status Register (CSR)

CSR defines the debug configuration for the processor and memory subsystem and contains status information from the breakpoint logic. CSR is accessible from the programming model using the WDEBUG instruction and through the BDM port using the READ_DREG and WRITE_DREG commands.

DRc: 0x00 (CSR) Access: Supervisor write-only

BDM read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BSTAT				FOF	TRG	HALT	BKPT	HRL				0	BKD	VBD	IPW
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TRO	0	DDC		UHE	BTB		0	NPL	IPI	SSM	0	0	FID	DDH
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 41-6. CSR Field Descriptions

Field	Description
31–28 BSTAT	Breakpoint status Provides read-only status (from the BDM port only) information concerning hardware breakpoints. BSTAT is cleared by a TDR write, by a CSR read when a level-2 breakpoint is triggered, or a level-1 breakpoint is triggered and the level-2 breakpoint is disabled. The PSTB value that follows the PSTB entry of 0x1B is 0x20 + (2 x BSTAT). 0000 No breakpoints enabled 0001 Waiting for level-1 breakpoint 0010 Level-1 breakpoint triggered 0101 Waiting for level-2 breakpointB 0110 Level-2 breakpoint triggered
27 FOF	Fault-on-fault Indicates a catastrophic halt occurred and forced entry into BDM. FOF is cleared by reset or when CSR is read (from the BDM port only).

Table continues on the next page...

Table 41-6. CSR Field Descriptions (continued)

Field	Description
26 TRG	Hardware breakpoint trigger Indicates a hardware breakpoint halted the processor core and forced entry into BDM. Reset, the debug GO command, or reading CSR (from the BDM port only) clears TRG.
25 HALT	Processor halt Indicates the processor executed a HALT and forced entry into BDM. Reset, the debug go command, or reading CSR (from the BDM port only) clears HALT.
24 BKPT	Breakpoint assert Indicates when either: • The $\overline{\text{BKPT}}$ input was asserted, • BDM BACKGROUND command received, or • The PSTB halt on full condition, CSR2[PSTBH], sets. This forces the processor into a BDM halt. Reset, the debug go command, or reading CSR (from the BDM port only) clears BKPT.
23–20 HRL	Hardware revision level Indicates, from the BDM port only, the level of debug module functionality. An emulator can use this information to identify the level of functionality supported. 0000 Revision A 0001 Revision B 0010 Revision C 0011 Revision D 1001 Revision B+ (The value used for this device) 0110 Revision D+
19	Reserved; must be cleared.
18 BKD	Breakpoint disable Disables the BACKGROUND command functionality, and allows the execution of the BACKGROUND command to generate a debug interrupt. 0 Normal operation 1 The receipt of a BDM BACKGROUND command signals a debug interrupt to the ColdFire core. The processor makes this interrupt request pending until the next sample point occurs, when the exception is initiated. In the ColdFire architecture, the interrupt sample point occurs once per instruction. There is no support for nesting debug interrupts.
17	Reserved; must be cleared.
16 IPW	Inhibit processor writes. Inhibits processor-initiated writes to the debug module's programming model registers. IPW can be modified only by commands from the BDM interface.
15	Reserved; must be cleared.
14 TRC	Force emulation mode on trace exception 0 Processor enters supervisor mode (default behavior). 1 Processor enters emulator mode when a trace exception occurs.
13	Reserved; must be cleared.

Table continues on the next page...

Table 41-6. CSR Field Descriptions (continued)

Field	Description	
12–11 DDC	Debug data control. Controls peripheral bus operand data capture for DDATA, which displays the number of bytes defined by the operand reference size (a marker) before the actual data; byte displays 8 bits, word displays 16 bits, and long displays 32 bits (one nibble at a time across multiple PSTCLK clock cycles). See Table 41-41. A non-zero value enables partial data trace capabilities.	
	00	No operand data is displayed.
	01	Capture all write data.
	10	Capture all read data.
	11	Capture all read and write data.
10 UHE	User halt enable Selects the CPU privilege level required to execute the HALT instruction. The core must be operating with XCSR[ENBDM] set to execute any HALT instruction, else the instruction is treated as an illegal opcode.	
	0	HALT is a supervisor-only instruction.
	1	HALT is a supervisor/user instruction.
9–8 BTB	Branch target bytes Defines the number of bytes of branch target address DDATA displays.	
	00	No target address capture.
	01	Lower 2 bytes of the target address
	1x	Lower 3 bytes of the target address
7	Reserved; must be cleared.	
6 NPL	Non-pipelined mode Determines if the core operates in pipelined mode.	
	0	Pipelined mode
	1	Non-pipelined mode. The processor effectively executes one instruction at a time with no overlap. This typically adds five cycles to the execution time of each instruction. Given an average execution latency of ~2 cycles per instruction, throughput in non-pipeline mode would be ~7 cycles per instruction, approximately 25% - 33% of pipelined performance.
	Regardless of the NPL state, a triggered PC breakpoint is always reported before the triggering instruction executes. In normal pipeline operation, the occurrence of an address and/or data breakpoint trigger is imprecise. In non-pipeline mode, these triggers are always reported before the next instruction begins execution and trigger reporting can be considered precise.	
5 IPI	Ignore pending interrupts when in single-step mode	
	0	Core services any pending interrupt requests signalled while in single-step mode.
	1	Core ignores any pending interrupt requests signalled while in single-step mode.
4 SSM	Single-step mode enable	
	0	Normal mode.
	1	Single-step mode. The processor halts after execution of each instruction. While halted, any BDM command can be executed. On receipt of the GO command, the processor executes the next instruction and halts again. This process continues until SSM is cleared.
3–2	Reserved; must be cleared.	

Table continues on the next page...

Table 41-6. CSR Field Descriptions (continued)

Field	Description	
1 FID	Force ipg_debug The core generates this output to the device, signaling it is in debug mode.	
	0	Do not force the assertion of ipg_debug.
	1	Force the assertion of ipg_debug.
0 DDH	Disable ipg_debug due to a halt condition. The core generates an output to the other modules in the device, signaling it is in debug mode. By default, this output signal is asserted when the core halts.	
	0	Assert ipg_debug if the core is halted.
	1	Negate ipg_debug due to the core being halted.

41.3.2 Extended Configuration/Status Register (XCSR)

The 32-bit XCSR is partitioned into two sections: the upper byte contains status and command bits always accessible to the BDM interface, even if debug mode is disabled. This status byte is also known as XCSR_SB. The lower 24 bits contain fields related to the generation of automatic SYNC_PC commands, which can be used to periodically capture and display the current program counter (PC) in the PST trace buffer (if properly configured).

This table summarizes the methods for accessing the XCSR.

Table 41-7. XCSR Access Summary

Reference method	Details
READ_XCSR_BYTE	Reads bits 31–24 from the BDM interface. Available in all modes.
WRITE_XCSR_BYTE	Writes bits 31–24 from the BDM interface. Available in all modes.
READ_DREG	Reads bits 31–0 from the BDM interface. Classified as a non-intrusive BDM command. Available in all modes.
WRITE_DREG	Writes bits 23–0 from the BDM interface. Classified as a non-intrusive BDM command. Available in all modes.
WDEBUG instruction	Writes bits 23–0 during execution of the core WDEBUG instruction. This instruction is a privileged supervisor-mode instruction.

DRc: 0x01 (XCSR)

Access: Supervisor write-only

BDM read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CPUHALT	CPUSTOP	CSTAT			CLKSW	SEC	ENBDM	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Table 41-9. XCSR Field Descriptions

Field	Description
31 CPUHALT	CPU Halt Indicates that the CPU is in the halt state. The CPU state may be running, stopped, or halted, as indicated by the CPUHALT and CPUSTOP bits. If both CPUHALT and CPUSTOP are 0, then the CPU is running.
30 CPUSTOP	CPU Stop Indicates that the CPU is in the stop state. The CPU state may be running, stopped, or halted, as indicated by the CPUHALT and CPUSTOP bits. If both CPUHALT and CPUSTOP are 0, then the CPU is running.
29–27 CSTAT	BDM Command Status Indicates the BDM command status. 000 Command done, no errors 001 Command done, data invalid 01x Command done, illegal 1xx Command busy, overrun If an overrun is detected (CSTAT = 1xx), the following sequence is suggested to clear the source of the error: 1. Issue a SYNC command to reset the BDC channel. 2. The host issues a BDM NOP command. 3. The host checks the channel status using a READ_XCSR_BYTE command. 4. If CSTAT = 000, then you can proceed; else: a. Halt the CPU with a BDM BACKGROUND command. b. Repeat steps 1 through 3. c. If CSTAT != 000, then reset device.

Table continues on the next page...

Table 41-9. XCSR Field Descriptions (continued)

Field	Description
26 CLKSW	Clock Select Select source for serial BDC communication clock. 0 Alternate, asynchronous BDC clock 1 Synchronous bus clock (CPU clock divided by 2) The initial state of the CLKSW bit is loaded by the hardware in response to certain reset events and the state of the BKGD pin, as described in Figure 41-2.
25 SEC	Flash Security Status This bit's read value typically indicates the status of the flash security field. 0 Flash security disabled 1 Flash security enabled If the user programs a mass erase via BDM, then SEC acts as a flash busy flag and its values have the following meanings. You can poll SEC to track the progress of the mass erase sequence. This bit is cleared immediately after the erase operation is complete, and then it returns to indicating the status of flash security. 0 Flash is not busy performing a BDM mass erase sequence 1 Flash is busy performing a BDM mass erase sequence
24 ENBDM	Enable BDM 0 Disable BDM 1 Enable BDM (assuming the flash is not secure, as indicated in SEC)
23–3 Reserved	Reserved for future use by the debug module. These bits must all equal 0b.
2–1 APCSC	Automatic PC Synchronization Control If APCENB is 1b, determines the periodic interval of PC address captures. When the selected interval is reached, a SYNC_PC command is sent to the CPU. For more information on the SYNC_PC operation, see the APCENB description. The chosen frequency depends on the setting of CSR2[APCDIV16], as shown in Table 41-10.
0 APCENB	Automatic PC Synchronization Enable Enables the periodic output of the PC, which can be used for PST/DDATA trace synchronization and code profiling. As described in APCSC, when the enabled periodic timer expires, a SYNC_PC command is sent to the CPU that generates a forced instruction fetch of the next instruction. The PST/DDATA module captures the target address as defined by CSR[9] (two bytes if CSR[9] is 0b, three bytes if CSR[9] is 1b). This produces a PST sequence of the PST marker indicating a 2- or 3-byte address, followed by the captured instruction address. 0 Disable automatic PC synchronization 1 Enable automatic PC synchronization

This table shows the selected PC address capture period as determined by the XCSR[APCENB], CSR2[APCDIV16], and XCSR[APCSC] fields.

Table 41-10. PC address capture period (SYNC_PC interval)

XCSR[APCENB]	CSR2[APCDIV16]	XCSR[APCSC]	SYNC_PC interval (cycles)
1	1	00	128
1	1	01	256
1	1	10	512
1	1	11	1024
1	0	00	2048
1	0	01	4096
1	0	10	8092
1	0	11	16384

41.3.3 Configuration/Status Register 2 (CSR2)

The 32-bit CSR2 is partitioned into two sections. The upper byte contains status and configuration bits always accessible to the BDM interface, even if debug mode is disabled. The lower 24 bits contain fields related to the configuration of the PST trace buffer (PSTB).

This table summarizes the methods for accessing CSR2.

Table 41-11. CSR2 Access Summary

Reference method	Details
READ_CSR2_BYTE	Reads bits 31–24 from the BDM interface. Available in all modes.
WRITE_CSR2_BYTE	Writes bits 31–24 from the BDM interface. Available in all modes.
READ_DREG	Reads bits 31–0 from the BDM interface. Classified as a non-intrusive BDM command.
WRITE_DREG	Writes bits 23–0 from the BDM interface. Classified as a non-intrusive BDM command.
WDEBUG instruction	Writes bits 23–0 during execution of the core WDEBUG instruction. This instruction is a privileged supervisor-mode instruction.

DRc: 0x02 (CSR2)

Access: Supervisor read-only

BDM read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PSTBP	0	COPHR	IOPHR	IADHR	0	BFHBR	0	PSTBH	PSTBST		0	D1HRL			
W								BDFR								
POR	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
Other Reset	0	0	u	u	u	0	u	0	0	0	0	0	0	0	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PSTBWA								0	APCDIV16	0	PSTBRM		PSTBSS		
W									PSTBR							
Reset	Undefined and Unaffected								0	0	0	0	0	0	0	0

Table 41-12. CSR2 Field Descriptions

Field	Description
31 PSTBP	PST Buffer Stop Signals if a PST buffer stop condition has been reached. 0 Stop condition has not been reached 1 Stop condition has been reached
30	Reserved. Must write 0.
29 COPHR	Computer Operating Properly Halt After Reset Specifies operation of the device after a COP reset. This bit is 0b after a power-on reset and is unaffected by any other reset. **NOTE:** This bit can be changed only if XCSR[ENBDM] is 1b and the system is not secure. 0 The device immediately enters normal operation mode. 1 The device halts (as if the BKGD pin was held low after a power-on reset).
28 IOPHR	Illegal Operation Halt After Reset Specifies operation of the device after an illegal operation reset. This bit is 0b after a power-on reset and is unaffected by any other reset. **NOTE:** This bit can be changed only if XCSR[ENBDM] is 1b and the flash is not secure. 0 The device immediately enters normal operation mode. 1 The device halts (as if the BKGD pin was held low after a power-on reset).
27 IADHR	Illegal Address Halt After Reset Specifies operation of the device after an illegal address reset. This bit is 0b after a power-on reset and is unaffected by any other reset. **NOTE:** This bit can be changed only if XCSR[ENBDM] is 1b and the system is not secure. 0 The device immediately enters normal operation mode. 1 The device halts (as if the BKGD pin was held low after a power-on reset).

Table continues on the next page...

Table 41-12. CSR2 Field Descriptions (continued)

Field	Description
26	Reserved. Must write 0.
25 BFHBR	BDM Force Halt on BDM Reset Specifies operation of the device after a BDM reset (on any reset based on external BKGD logic implementation). This bit is cleared after a power-on reset and is unaffected by any other reset. **NOTE:** This bit can be changed only if XCSR[ENBDM] is 1b and the system is not secure. 0 The device immediately enters normal operation mode. 1 The device halts (as if the BKGD pin was held low after a power-on reset).
24 BDFR	Background Debug Force Reset Forces a BDM reset to the device. This bit always reads as 0 after the reset has been initiated. 0 No reset 1 Force a BDM reset
23 PSTBH	PST Trace Buffer Halt Indicates if the processor is halted due to the PST trace buffer being full when recording in obtrusive mode. 0 Not halted 1 Halted
22–21 PSTBST	PST Trace Buffer State Indicates the current state of PST trace buffer recording. 00 Disabled 01 Enabled and waiting for the start condition 10 Enabled, recording, and waiting for the stop condition 11 Enabled, completed recording after the stop condition was reached
20	Reserved. Must write 0.
19–16 D1HRL	Debug 1-pin Hardware Revision Level Indicates the hardware revision level of the 1-pin debug module implemented in the core. For this device, this field is 1h3h.
15–8 PSTBWA	PST Trace Buffer Write Address Indicates the current write address of the PST trace buffer. The most significant bit of this field is sticky; if set, it remains set until a PST/DDATA reset event occurs. As the core inserts PST and DDATA packets into the trace buffer, this field is incremented. The value of the write address defines the next location in the PST trace buffer to be loaded. In other words, the contents of PSTB[PSTBWA – 1] is the last valid entry in the trace buffer. The most-significant bit of this field can be used to determine if the entire PST trace buffer has been loaded with valid data. The PSTBWA is unaffected when a buffer stop condition has been reached, the buffer is disabled, or a system reset occurs. This allows the contents of the PST trace buffer to be retrieved after these events to assist in debug. **NOTE:** Because this device contains a 64-entry trace buffer, PSTBWA[6] is always zero. Bit 7 PSTB valid data locations (oldest to newest) 0 0, 1, ..., PSTBWA – 1 1 PSTBWA, PSTBWA + 1, ..., 0, 1, ..., PSTBWA – 1

Table continues on the next page...

Table 41-12. CSR2 Field Descriptions (continued)

Field	Description
7 PSTBR	PST Trace Buffer Reset Generates a reset of the PST trace buffer logic, which clears PSTBWA and PSTBST. The same resources are reset when a disabled trace buffer becomes enabled and upon the receipt of a BDM GO command when operating in obtrusive trace mode. These reset events also clear any accumulation of PSTs. This bit always reads as 0b. 0 — No reset 1 — Force a PST trace buffer reset
6 APCDIV16	Automatic PC Synchronization Divide Cycle Counts by 16 Divides the cycle counts for automatic SYNC_PC command insertion by 16. See the XCSR[APCSC] and XCSR[APCENB] fields.
5	Reserved. Must write 0.
4–3 PSTBRM	PST Trace Buffer Recording Mode Specifies the trace buffer recording mode. The start and stop recording conditions are defined by the PSTBSS field. The terms obtrusive and non-obtrusive are defined as: • Non-obtrusive—The core is not halted. The PST trace buffer is overwritten unless a PSTB start/stop combination results in less than or equal to 64 PSTB captures. • Obtrusive—The core is halted when the PSTB trace buffer reaches its full level (full before overwriting). The PSTB trace buffer contents are available by the BDM PSTB_READ commands. The PSTB trace buffer write address resets and the CPU resumes upon a BDM GO command. 00 — Non-obtrusive, normal recording mode 01 — Obtrusive, normal recording 10 — Non-obtrusive, PC profile recording. Automatic PC synchronization must be enabled (see XCSR[APCSC, APCENB], CSR2[APCDIV16], and CSR[BTB]). 11 — Obtrusive, PC profile recording. Automatic PC synchronization must be enabled (see XCSR[APCSC, APCENB], CSR2[APCDIV16], and CSR[BTB]).
2–0 PSTBSS	PST Trace Buffer Start/Stop Definition Specifies the start and stop conditions for PST trace buffer recording. In certain cases, the start and stop conditions are defined by the breakpoint registers. The remaining breakpoint registers are available for trigger configurations. See Table 41-13.

This table shows the start and stop recording conditions as specified by the PSTBSS field.

Table 41-13. PST trace buffer start and stop recording conditions (CSR2[PSTBSS])

PSTBSS	Start condition	Stop condition
000	Trace buffer disabled, no recording	
001	Unconditional recording	
010	ABxR{& DBR/DBMR}	PBR0/PBMR
011		PBR1
100	PBR0/PBMR	ABxR{& DBR/DBMR}
101		PBR1

Table continues on the next page...

Table 41-13. PST trace buffer start and stop recording conditions (CSR2[PSTBSS]) (continued)

PSTBSS	Start condition	Stop condition
110	PBR1	ABxR{& DBR/DBMR}
111		PBR0/PBMR

41.3.4 Configuration/Status Register 3 (CSR3)

The CSR3 contains fields enabling indirect writes to the DBGCR. Writes to these CSR3 fields execute 4-bit nibble writes to the DBGCR. Thus, writing to the entire 32-bit DBGCR would require eight "write CSR3" commands be processed via the single-pin BDM interface.

When read, CSR3 enables indirect reads of the DBGSR. When a "read CSR3" command is transmitted on the single-pin BDM interface, one byte of DBGSR is read and returned. The debug logic maintains a 2-bit counter that selects the byte of the DBGSR register to be read, starting with DBGSR[7:0]. The 2-bit counter is cleared by POR and by any write to the CSR3. It is incremented by any read of the CSR3. Thus, a read of the entire 32-bit DBGSR would require four "read CSR3" commands be processed on the single-pin BDM interface, and the sequential commands would return data in the following order: DBGSR[7:0], DBGSR[15:8], DBGSR[23:16], and DBGSR[31:24].

This table summarizes the methods for accessing CSR3.

Table 41-14. CSR3 Reference Summary

Method	Reference Details
READ_CSR3_BYTE	Reads bits CSR3[31–24] from the BDM interface. Available in all modes.
WRITE_CSR3_BYTE	Writes bits CSR3[31–24] from the BDM interface. Available in all modes.
READ_DREG	Reads bits CSR3[31–0] from the BDM interface. Classified as a non-intrusive BDM command.
WRITE_DREG	Writes bits CSR3[23–0] from the BDM interface. Classified as a non-intrusive BDM command.
WDEBUG instruction	No operation while the core executes a WDEBUG instruction.

DRc: 0x03 (CSR3) Access: Supervisor write-only

BDM read/write

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

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

Table 41-15. CSR3 Field Descriptions

Field	Description
31 UI	Update indicator for write to DBGCR When set to 1, this bit signals the chip that an update has been completed and the content of the DBGCR is valid.
30–28 NS	Nibble select for write to DBGCR This field selects which nibble in the DBGCR is written. 000 Write DBGCR[31:28] 001 Write DBGCR[27:24] 010 Write DBGCR[23:20] ... ... 111 Write DBGCR[3:0]
27–24 WD	Write data to DBGCR This field defines the data to be written into the selected nibble of the DBGCR. The selected nibble is determined by the value of CSR3[30:28].
23–0	Reserved for future use by the debug module; must be cleared.

41.3.5 Debug Control Register (DBGCR) and Debug Status Register (DBGSR)

The 32-bit DBGCR and 32-bit DBGSR are accessible only via the BDM port. They provide functionality for mass erase operations and for monitoring when the chip is or was in low power modes.

The DBGCR and DBGSR are accessed indirectly via fields in the CSR3.

The CSR3 contains fields enabling indirect writes to the DBGCR. Writes to these CSR3 fields execute 4-bit nibble writes to the DBGCR. Thus, writing to the entire 32-bit DBGCR would require eight "write CSR3" commands be processed via the single-pin BDM interface.

When read, CSR3 enables indirect reads of the DBGSR. When a "read CSR3" command is transmitted on the single-pin BDM interface, one byte of DBGSR is read and returned. The debug logic maintains a 2-bit counter that selects the byte of the DBGSR register to be read, starting with DBGSR[7:0]. The 2-bit counter is cleared by POR and by any write to the CSR3. It is incremented by any read of the CSR3. Thus, a read of the entire 32-bit DBGSR would require four "read CSR3" commands be processed on the single-pin BDM interface, and the sequential commands would return data in the following order: DBGSR[7:0], DBGSR[15:8], DBGSR[23:16], and DBGSR[31:24].

Using these registers, the flash memory mass erase procedure consists of these steps:

1. Initiate the operation: Write the CSR3 to set DBGCR[0] to 1.
2. Confirm that the operation is complete: Read the CSR3 to poll the DBGSR[0] flag, which reflects the operation's status. The operation is complete when DBGSR[0] is 1.

Table 41-16. DBGCR Definition

Bit Number	Bit Name	Description
0	Flash Mass Erase Command	Set this bit to 1 to initiate a mass erase. This bit is cleared by hardware after the launch of the mass erase operation. **NOTE:** When the mass erase capability is disabled (due to the values of the flash memory module's FSEC[MEEN] and FSEC[SEC] fields), the Erase Command request is still issued and acknowledged, but no erase operation occurs.
1	VLLSx Status Acknowledge	Set this bit to 1 to acknowledge that a read of the DBGSR's VLLSx Mode Exit status bit has occurred. This acknowledgement automatically clears the status bit.

Table 41-17. DBGSR Definition

Bit Number	Bit Name	Description
0	ERASE STATUS	This bit is cleared after any system reset. The bit is also cleared at the launch of a mass erase command due to a write of the DBGCR[0] bit. The bit is set at the completion of the mass erase sequence. **NOTE:** If the mass erase function is disabled, the mass erase sequence is completed, but no erase operation occurs. The value of DBGSR[2] reflects whether the flash memory module's FSEC[MEEN] field is set to disable the mass erase function. 0: Status cleared or mass erase not done 1: Mass erase sequence done

Table continues on the next page...

Table 41-17. DBGSR Definition (continued)

Bit Number	Bit Name	Description
1	Freescale Factory Access	This bit indicates whether or not Freescale's factory testing can access the chip's flash memory contents. 0: Freescale factory access is denied (FSEC[FSLACC] is 01 or 10) 1: Freescale factory access is granted (FSEC[FSLACC] is 00 or 11)
2	Mass Erase Enable	This bit indicates whether or not the flash memory can be mass erased. 0: Mass erase is disabled (FSEC[MEEN] is 10) 1: Mass erase is enabled (FSEC[MEEN] is 00, 01, or 11)
3	Backdoor Key Access Enable	This bit indicates whether or not the flash memory has backdoor key access enabled. 0: Backdoor key access is disabled (FSEC[KEYEN] is 00, 01, or 11) 1: Backdoor key access is enabled (FSEC[KEYEN] is 10)
4	Very Low Power Modes	This bit always indicates whether the current power mode is one of the VLPx modes. This bit is used to throttle debugger frequency up/down.
6	VLLSx Modes Exit	This bit indicates that an exit from one of the VLLSx modes has occurred. The debugger loses communication (including access to this register) while the system is in a VLLSx mode. When communication is re-established, this bit indicates that the system had been in a VLLSx mode. The debug modules lose their state during VLLSx modes, so they must be reconfigured after an exit from a VLLSx mode. The VLLSx Mode Exit bit is held until the debugger recognizes that a VLLSx mode was exited. The bit is cleared by a write of 1 to the DBGCR[1] bit (VLLSx Status Acknowledge bit).

41.3.6 BDM Address Attribute Register (BAAR)

BAAR defines the address space for memory-referencing BDM commands. BAAR[R, SZ] are loaded directly from the BDM command, while the lower five bits can be programmed from the external development system. BAAR is loaded any time AATR is written and is initialized to a value of 0x05, setting supervisor data as the default address space. The upper 24 bits of this register are reserved for future use and any attempted write of these bits has no effect.

DRc: 0x05 (BAAR)

Access: Supervisor write-only
BDM write-only

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

Table 41-19. BAAR Field Descriptions

Field	Description	
31–8	Reserved for future use by the debug module; must be cleared.	
7 R	Read/Write	
	0	Write.
	1	Read.
6–5 SZ	Size	
	00	Longword
	01	Byte
	10	Word
	11	Reserved
4–3 TT	Transfer type See the TT definition in the AATR description.	
2–0 TM	Transfer modifier See the TM definition in the AATR description.	

41.3.7 Address Attribute Trigger Register (AATR)

AATR defines address attributes and a mask to be matched in the trigger. The register value is compared with address attribute signals from the processor's high-speed local bus, as defined by the setting of the trigger definition register (TDR). AATR is accessible in supervisor mode as debug control register 0x06 using the WDEBUG instruction and through the BDM port using the WRITE_DREG command.

DRc: 0x06 (AATR) Access: Supervisor write-only

BDM write-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	RM	SZM		TTM		TMM			R	SZ		TT		TM		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Table 41-21. AATR Field Descriptions

Field	Description
31–16	Reserved; must be cleared.
15 RM	Read/write mask Masks the R bit in address comparisons.
14–13 SZM	Size mask Masks the corresponding SZ bit in address comparisons.
12–11 TTM	Transfer type mask Masks the corresponding TT bit in address comparisons.
10–8 TMM	Transfer modifier mask Masks the corresponding TM bit in address comparisons.
7 R	Read/Write R is compared with the R/$\overline{W}$ signal of the processor's local bus.
6–5 SZ	Size Compared to the processor's local bus size signals. 00 Longword 01 Byte 10 Word 11 Reserved
4–3 TT	Transfer type Compared with the local bus transfer type signals. These bits also define the TT encoding for BDM memory commands. 00 Normal processor access Else Reserved
2–0 TM	Transfer modifier Compared with the local bus transfer modifier signals, which give supplemental information for each transfer type. These bits also define the TM encoding for BDM memory commands (for backward compatibility). 000 Reserved 001 User-mode data access 010 User-mode code access 011 Reserved 100 Reserved 101 Supervisor-mode data access 110 Supervisor-mode code access 111 Reserved

41.3.8 Trigger Definition Register (TDR)

TDR configures the operation of the hardware breakpoint logic that corresponds with the ABHR/ABLR/AATR, PBR/PBR1/PBR2/PBR3/PBMR, and DBR/DBMR registers within the debug module. TDR controls the actions taken under the defined conditions. Breakpoint logic may be configured as one- or two-level trigger. TDR[31–16] defines the second-level trigger, and TDR[15–0] defines the first-level trigger.

NOTE

The debug module has no hardware interlocks. To prevent spurious breakpoint triggers while the breakpoint registers are being loaded, disable TDR (write 0 to L2EBL and L1EBL) before defining triggers.

A write to TDR clears the CSR trigger status bits, CSR[BSTAT]. TDR is accessible in supervisor mode as debug control register 0x07 using the WDEBUG instruction and through the BDM port using the WRITE_DREG command.

DRc: 0x07 (TDR) Access: Supervisor write-only, BDM write-only

Second Level Trigger

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	TRC		L2EBL	L2ED							L2DI	L2EA			L2EPC	L2PCI
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

First Level Trigger

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	L2T	L1T	L1EBL	L1ED							L1DI	L1EA			L1EPC	L1PCI
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 41-23. TDR Field Descriptions

Field	Description	
31–30	Trigger Response Control	
TRC	Determines how the processor responds to a completed trigger condition. The trigger response is displayed on PST.	
	00	Display on PST only
	01	Processor halt
	10	Debug interrupt
	11	Reserved

Table continues on the next page...

Table 41-23. TDR Field Descriptions (continued)

Field	Description
29 L2EBL	Enable Level 2 Breakpoints Global enable for the level 2 breakpoint triggers. 0 Disable all level 2 breakpoints 1 Enable all level 2 breakpoints
28–22 L2ED	Enable Level 2 Data Breakpoint Setting an L2ED bit enables the corresponding data breakpoint condition based on the size and placement on the processor's local data bus. Clearing all ED bits disables data breakpoints. Bit — Description 28 — Data longword. Entire processor's local data bus 27 — Lower data word 26 — Upper data word 25 — Lower lower data byte. Low-order byte of the low-order word 24 — Lower middle data byte. High-order byte of the low-order word 23 — Upper middle data byte. Low-order byte of the high-order word 22 — Upper upper data byte. High-order byte of the high-order word
21 L2DI	Level 2 Data Breakpoint Invert Inverts the logical sense of all the data breakpoint comparators. This can develop a trigger based on the occurrence of a data value other than the DBR contents. 0 Do not invert 1 Invert
20–18 L2EA	Enable Level 2 Address Breakpoint Setting an L2EA bit enables the corresponding address breakpoint. Clearing all three bits disables the breakpoint. Bit — Description 20 — Address breakpoint inverted. Breakpoint is based outside the range between ABLR and ABHR. 19 — Address breakpoint range. The breakpoint is based on the inclusive range defined by ABLR and ABHR. 18 — Address breakpoint low. The breakpoint is based on the address in the ABLR.
17 L2EPC	Enable Level 2 PC Breakpoint 0 Disable 1 Enable
16 L2PCI	Level 2 PC Breakpoint Invert 0 Do not invert 1 Invert
15 L2T	Level 2 Trigger Determines the logic operation for the trigger between the PC_condition and the (Address_range and Data_condition) condition where the inclusion of a Data_condition is optional. The debug architecture supports the creation of single- or double-level triggers. 0 Trigger = PC_condition && (Address_range && Data_condition) 1 Trigger = PC_condition || (Address_range && Data_condition)

Table continues on the next page...

Table 41-23. TDR Field Descriptions (continued)

Field	Description
14 L1T	Level 1 Trigger Determines the logic operation for the trigger between the PC_condition and the (Address_range and Data_condition) condition where the inclusion of a Data_condition is optional. The debug architecture supports the creation of single- or double-level triggers. 0 Trigger = PC_condition && (Address_range && Data_condition) 1 Trigger = PC_condition || (Address_range && Data_condition)
13 L1EBL	Enable Level 1 Breakpoints Global enable for the level 1 breakpoint triggers. 0 Disable all level 1 breakpoints 1 Enable all level 1 breakpoints
12–6 L1ED	Enable Level 1 Data Breakpoint Setting an L1ED bit enables the corresponding data breakpoint condition based on the size and placement on the processor's local data bus. Clearing all ED bits disables data breakpoints. Bit Description 28 Data longword. Entire processor's local data bus 27 Lower data word 26 Upper data word 25 Lower lower data byte. Low-order byte of the low-order word 24 Lower middle data byte. High-order byte of the low-order word 23 Upper middle data byte. Low-order byte of the high-order word 22 Upper upper data byte. High-order byte of the high-order word
5 L1DI	Level 1 Data Breakpoint Invert Inverts the logical sense of all the data breakpoint comparators. This can develop a trigger based on the occurrence of a data value other than the DBR contents. 0 Do not invert 1 Invert
4–2 L1EA	Enable Level 1 Address Breakpoint Setting an L1EA bit enables the corresponding address breakpoint. Clearing all three bits disables the breakpoint. Bit Description 20 Address breakpoint inverted. Breakpoint is based outside the range between ABLR and ABHR. 19 Address breakpoint range. The breakpoint is based on the inclusive range defined by ABLR and ABHR. 18 Address breakpoint low. The breakpoint is based on the address in the ABLR.
1 L1EPC	Enable Level 1 PC Breakpoint 0 Disable 1 Enable
0 L1PCI	Level 1 PC Breakpoint Invert 0 Do not invert 1 Invert

41.3.9 Program Counter Breakpoint/Mask Registers (PBR0–3, PBMR)

The PBRn registers define instruction addresses for use as part of the trigger. These registers' contents are compared with the processor's program counter register when the appropriate valid bit is set (for PBR1–3) and TDR is configured appropriately. PBR0 bits are masked by setting corresponding PBMR bits (PBMR has no effect on PBR1–3). Results are compared with the processor's program counter register, as defined in TDR. The PC breakpoint registers, PBR1–3, have no masking associated with them, but do include a valid bit. These registers' contents are compared with the processor's program counter register when TDR is configured appropriately.

The PC breakpoint registers are accessible in supervisor mode using the WDEBUG instruction and through the BDM port using the WRITE_DREG command using values shown in BDM Command Set Descriptions.

NOTE

Version 1 ColdFire core devices implement a 24-bit, 16 MB address map. When programming these registers with a 32-bit address, the upper byte must be zero-filled.

DRc: 0x08 (PBR0) Access: Supervisor write-only

BDM write-only

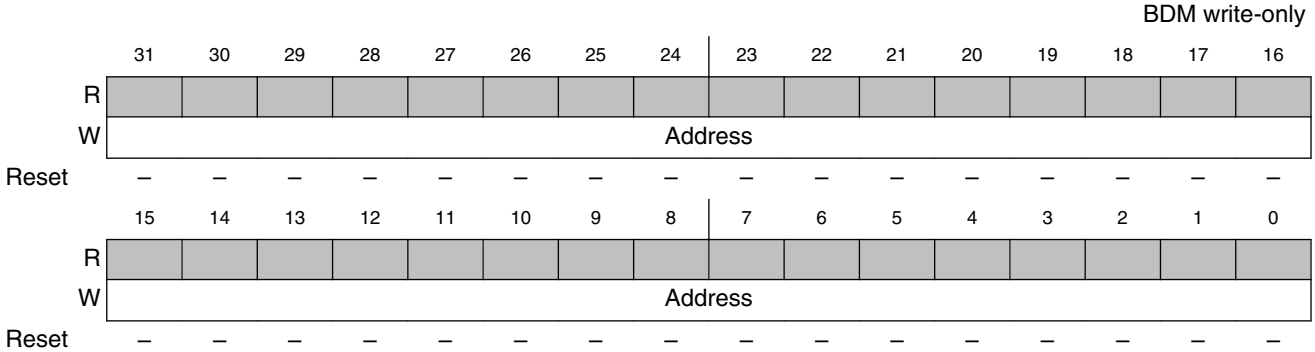


Table 41-25. PBR0 Field Descriptions

Field	Description
31–0 Address	PC breakpoint address The address to be compared with the PC as a breakpoint trigger. Because all instruction sizes are multiples of 2 bytes, bit 0 of the address should always be zero.

DRc: 0x18 (PBR1)
0x1A (PBR2)
0x1C (PBR3)

Access: Supervisor write-only
BDM write-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	Address															
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	Address															V
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0

Table 41-27. PBRn Field Descriptions

Field	Description	
31–1 Address	PC breakpoint address The 31-bit address to be compared with the PC as a breakpoint trigger.	
0 V	Valid bit This bit must be set for the PC breakpoint to occur at the address specified in the Address field.	
	0	PBR is disabled.
	1	PBR is enabled.

The following display shows PBMR. PBMR is accessible in supervisor mode using the WDEBUG instruction and via the BDM port using the WRITE_DREG command. PBMR only masks PBR0.

DRc: 0x09

Access: Supervisor write-only
BDM write-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	Mask															
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

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

Table 41-29. PBMR Field Descriptions

Field	Description	
31–1 Mask	PC breakpoint mask If using PBR0, this register must be initialized since it is undefined after reset.	
	0	The corresponding PBR0 bit is compared to the appropriate PC bit.
	1	The corresponding PBR0 bit is ignored.

41.3.10 Address Breakpoint Registers (ABLR, ABHR)

The ABLR and ABHR define regions in the processor's data address space that can be used as part of the trigger. These register values are compared with the address for each transfer on the processor's high-speed local bus. The trigger definition register (TDR) identifies the trigger as one of three cases:

- Identical to the value in ABLR
- Inside the range bound by ABLR and ABHR inclusive
- Outside that same range

The address breakpoint registers are accessible in supervisor mode using the WDEBUG instruction and through the BDM port using the WRITE_DREG command using values shown in BDM Command Set Descriptions.

NOTE

Version 1 ColdFire core devices implement a 24-bit, 16 MB address map. When programming these registers with a 32-bit address, the upper byte should be zero-filled when referencing

the flash, RAM, and RGPIO regions, and set to 0xFF when referencing any of the slave peripheral devices.

DRc: 0x0C (ABHR) Access: Supervisor write-only
0x0D (ABLR) BDM write-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	Address															
ABHR Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ABLR Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	Address															
ABHR Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ABLR Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 41-31. ABLR Field Descriptions

Field	Description
31–0 Address	Low address Holds the 32-bit address marking the lower bound of the address breakpoint range. Breakpoints for specific addresses are programmed into ABLR.

Table 41-32. ABHR Field Descriptions

Field	Description
31–0 Address	High address Holds the 32-bit address marking the upper bound of the address breakpoint range.

41.3.11 Data Breakpoint and Mask Registers (DBR, DBMR)

DBR specifies data patterns used as part of the trigger into debug mode. DBR bits are masked by setting corresponding DBMR bits, as defined in TDR.

DBR and DBMR are accessible in supervisor mode using the WDEBUG instruction and through the BDM port using the WRITE_DREG commands.

DRc: 0x0E (DBR) Access: Supervisor write-only
0x0F (DBMR) BDM write-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	Data (DBR); Mask (DBMR)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	Data (DBR); Mask (DBMR)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 41-34. DBR Field Descriptions

Field	Description
31–0 Data	Data breakpoint value. Contains the value to be compared with the data value from the processor's local bus as a breakpoint trigger.

Table 41-35. DBMR Field Descriptions

Field	Description	
31–0 Mask	Data breakpoint mask The 32-bit mask for the data breakpoint trigger.	
	0	The corresponding DBR bit is compared to the appropriate bit of the processor's local data bus.
	1	The corresponding DBR bit is ignored

The DBR supports aligned and misaligned references. This table shows the relationships between processor address, access size, and location within the 32-bit data bus.

Table 41-36. Access Size and Operand Data Location

Address[1–0]	Access Size	Operand Location
00	Byte	D[31–24]
01	Byte	D[23–16]
10	Byte	D[15–8]
11	Byte	D[7–0]
0x	Word	D[31–16]
1x	Word	D[15–0]
xx	Longword	D[31–0]

41.3.12 Resulting Set of Possible Trigger Combinations

The resulting set of possible breakpoint trigger combinations consists of the following options where || denotes logical OR, && denotes logical AND, and {} denotes an optional additional trigger term:

One-level triggers of the form:

```
if        (PC_breakpoint)
if        (PC_breakpoint || Address_breakpoint{&& Data_breakpoint})
if        (Address_breakpoint {&& Data_breakpoint})
```

Two-level triggers of the form:

```
if        (PC_breakpoint)
          then if (Address_breakpoint {&& Data_breakpoint} )

if        (Address_breakpoint {&& Data_breakpoint} )
          then if (PC_breakpoint)
```

In these examples, PC_breakpoint is the logical summation of the PBR0/PBMR, PBR1, PBR2, and PBR3 breakpoint registers; Address_breakpoint is a function of ABHR, ABLR, and AATR; Data_breakpoint is a function of DBR and DBMR. In all cases, the data breakpoints can be included with an address breakpoint to further qualify a trigger event as an option.

The breakpoint registers can also be used to define the start and stop recording conditions for the PST trace buffer.

41.3.13 PST Buffer (PSTB)

The PST trace buffer contains 64 six-bit entries, packed consecutively into 12 longword locations. The following table illustrates how the buffer entries are packed.

The write pointer for the trace buffer is available as CSR2[PSTBWA]. Using this pointer, it is possible to determine the oldest-to-newest entries in the trace buffer.

Table 41-37. PST Trace Buffer Entries and Locations

Core register number (CRN)	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
0x10	TB #00						TB #01						TB #02						TB #03						TB #04						05[5:4]	
0x11	TB #05[3:0]				TB #06				TB #07				TB #08						TB #09						TB #10[5:2]							
0x12	10[1:0]		TB #11						TB #12						TB #13						TB #14						TB #15					
0x13	TB #16						TB #17						TB #18						TB #19						TB #20						21[5:4]	
0x14	TB #21[3:0]				TB #22				TB #23				TB #24						TB #25						TB #26[5:2]							
0x15	26[1:0]		TB #27						TB #28						TB #29						TB #30						TB #31					
0x16	TB #32						TB #33						TB #34						TB #35						TB #36						37[5:4]	
0x17	TB #37[3:0]				TB #38				TB #39				TB #40						TB #41						TB #42[5:2]							
0x18	42[1:0]		TB #43						TB #44						TB #45						TB #46						TB #47					
0x19	TB #48						TB #49						TB #50						TB #51						TB #52						53[5:4]	

Table continues on the next page...

Table 41-37. PST Trace Buffer Entries and Locations (continued)

Core register number (CRN)	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
0x1A	TB #53[3:0]				TB #54						TB #55						TB #56						TB #57						TB #58[5:2]			
0x1B	58[1:0]		TB #59						TB #60						TB #61						TB #62						TB #63					

41.4 Functional Description

The following sections describe functional details of the debug module.

41.4.1 Background Debug Mode (BDM)

This section provides details on the background debug serial interface controller (BDC) and the BDM command set.

The BDC provides a single-wire debug interface to the target MCU. As shown in the Version 1 ColdFire core block diagram of Figure 41-1, the BDC module interfaces between the single-pin (BKGD) interface and the remaining debug modules, including the ColdFire background debug logic, the real-time debug hardware, and the PST/DDATA trace logic. This interface provides a convenient means for programming the on-chip flash and othernon-volatile memories. The BDC is the primary debug interface for development and allows non-intrusive access to memory data and traditional debug features such as run/halt control, read/write of core registers, breakpoints, and single instruction step.

Features of the background debug controller (BDC) include:

- Single dedicated pin for mode selection and background communications
- Special BDC registers not located in system memory map
- SYNC command to determine target communications rate
- Non-intrusive commands for memory access
- Active background (halt) mode commands for core register access
- GO command to resume execution
- BACKGROUND command to halt core or wake CPU from low-power modes
- Oscillator runs in stop mode, if BDM enabled

Based on these features, BDM is useful for the following reasons:

- In-circuit emulation is not needed, so physical and electrical characteristics of the system are not affected.
- BDM is always available for debugging the system and provides a communication link for upgrading firmware in existing systems.
- Provides high-speed memory downloading, especially useful for flash programming
- Provides absolute control of the processor, and thus the system. This feature allows quick hardware debugging with the same tool set used for firmware development.

41.4.1.1 CPU Halt

Although certain BDM operations can occur in parallel with CPU operations, unrestricted BDM operation requires the CPU to be halted. The sources that can cause the CPU to halt are listed below in order of priority. Recall that the default configuration of the Version 1 ColdFire core (CF1Core) defines the occurrence of certain exception types to automatically generate a system reset. Some of these fault types include illegal instructions, privilege errors, address errors, and bus error terminations, with the response under control of the processor's CPUCR[ARD, IRD] bits.

Table 41-38. CPU Halt Sources

Halt Source	Halt Timing	Description		
Fault-on-fault	Immediate	Refers to the occurrence of any fault while exception processing. For example, a bus error is signaled during exception stack frame writes or while fetching the first instruction in the exception service routine.		
		CPUCR[ARD] = 1	Immediately enters halt.	
		CPUCR[ARD] = 0	Reset event is initiated.	
Hardware breakpoint trigger	Pending	Halt is made pending in the processor. The processor samples for pending halt and interrupt conditions once per instruction. When a pending condition is asserted, the processor halts execution at the next sample point.		
HALT instruction	Immediate	BDM disabled	CPUCR[IRD] = 0	A reset is initiated since attempted execution of an illegal instruction
			CPUCR[IRD] = 1	An illegal instruction exception is generated.
		BDM enabled, supervisor mode	Processor immediately halts execution at the next instruction sample point. The processor can be restarted by a BDM GO command. Execution continues at the instruction after HALT.	
		BDM enabled, user mode	CSR[UHE] = 0 CPUCR[IRD] = 0	A reset event is initiated, because a privileged instruction was attempted in user mode.
			CSR[UHE] = 0 CPUCR[IRD] = 1	A privilege violation exception is generated.
			CSR[UHE] = 1	Processor immediately halts execution at the next instruction sample point. The processor can be restarted by a BDM GO command. Execution continues at the instruction after HALT.
BACKGROUND command	Pending	BDM disabled or flash secure	Illegal command response and BACKGROUND command is ignored.	
		BDM enabled and flash unsecure	Processor is running	Halt is made pending in the processor. The processor samples for pending halt and interrupt conditions once per instruction. When a pending condition is asserted, the processor halts execution at the next sample point.
			Processor is stopped	Processing of the BACKGROUND command is treated in a special manner. The processor exits the stopped mode and enters the halted state, at which point all BDM commands may be exercised. When restarted, the processor continues by executing the next sequential instruction (the instruction following STOP).
PSTB full condition	Pending	PSTB	PSTB obtrusive recording mode pends halt in the processor if the trace buffer reaches its full threshold (full is defined as before the buffer is overwritten). When a pending condition is asserted, the processor halts at the next sample point.	

Table continues on the next page...

Table 41-38. CPU Halt Sources (continued)

Halt Source	Halt Timing	Description	
BKGD held low for ≥2 bus clocks after reset negated for POR or BDM reset	Immediate	Flash unsecure	Enters debug mode with XCSR[ENBDM, CLKSW] set. The full set of BDM commands is available and debug can proceed. If the core is reset into a debug halt condition, the processor's response to the GO command depends on the BDM command(s) performed while it was halted. Specifically, if the PC register was loaded, the GO command causes the processor to exit halted state and pass control to the instruction address in the PC, bypassing normal reset exception processing. If the PC was not loaded, the GO command causes the processor to exit halted state and continue reset exception processing.
		Flash secure	Enters debug mode with XCSR[ENBDM, CLKSW] set. The allowable commands are limited to the always-available group. A GO command to start the processor is not allowed. The only recovery actions in this mode are: • Issue a BDM reset setting CSR2[BDFR] with CSR2[BDHBR] cleared and the BKGD pin held high to reset into normal operating mode • Erase the flash to unsecure the memory and then proceed with debug • Power cycle the device with the BKGD pin held high to reset into the normal operating mode

The processor's run/stop/halt status is always accessible in XCSR[CPUHALT,CPUSTOP]. Additionally, CSR[27–24] indicate the halt source, showing the highest priority source for multiple halt conditions. This field is cleared by a read of the CSR. A processor halt due to the PSTB full condition as indicated by CSR2[PSTH] is also reflected in CSR[BKPT]. The debug GO command clears CSR[26–24] and CSR2[PSTBH].

41.4.1.2 Background Debug Serial Interface Controller (BDC)

BDC serial communications use a custom serial protocol first introduced on the M68HC12 Family of microcontrollers and later used in the M68HCS08 family. This protocol assumes that the host knows the communication clock rate determined by the target BDC clock rate. The BDC clock rate may be the system bus clock frequency or an alternate frequency source depending on the state of XCSR[CLKSW]. All communication is initiated and controlled by the host which drives a high-to-low edge to signal the beginning of each bit time. Commands and data are sent most significant bit (msb) first. For a detailed description of the communications protocol, refer to BDM Communication Details.

If a host is attempting to communicate with a target MCU that has an unknown BDC clock rate, a SYNC command may be sent to the target MCU to request a timed synchronization response signal from which the host can determine the correct communication speed. After establishing communications, the host can read XCSR and write the clock switch (CLKSW) control bit to change the source of the BDC clock for further serial communications if necessary.

BKGD is a pseudo-open-drain pin and there is an on-chip pullup so no external pullup resistor is required. Unlike typical open-drain pins, the external RC time constant on this pin, which is influenced by external capacitance, plays almost no role in signal rise time. The custom protocol provides for brief, actively driven speed-up pulses to force rapid rise times on this pin without risking harmful drive level conflicts. Refer to BDM Communication Details, for more details.

When no debugger pod is connected to the standard 6-pin BDM interface connector (Freescale-Recommended BDM Pinout), the internal pullup on BKGD chooses normal operating mode. When a development system is connected, it can pull BKGD and $\overline{\text{RESET}}$ low, release $\overline{\text{RESET}}$ to select active background (halt) mode rather than normal operating mode, and then release BKGD. It is not necessary to reset the target MCU to communicate with it through the background debug interface. There is also a mechanism to generate a reset event in response to setting CSR2[BDFR].

41.4.1.3 BDM Communication Details

The BDC serial interface requires the external host controller to generate a falling edge on the BKGD pin to indicate the start of each bit time. The external controller provides this falling edge whether data is transmitted or received.

BKGD is a pseudo-open-drain pin that can be driven by an external controller or by the MCU. Data is transferred msb first at 16 BDC clock cycles per bit (nominal speed). The interface times-out if 512 BDC clock cycles occur between falling edges from the host. If a time-out occurs, the status of any command in progress must be determined before new commands can be sent from the host. To check the status of the command, follow the steps detailed in the bit description of XCSR[CSTAT].

The custom serial protocol requires the debug pod to know the target BDC communication clock speed. The clock switch (CLKSW) control bit in the XCSR[31–24] register allows you to select the BDC clock source. The BDC clock source can be the bus clock or the alternate BDC clock source. When the MCU is reset in normal user mode, CLKSW is cleared and that selects the alternate clock source. This clock source is a fixed frequency independent of the bus frequency so it does change if the user modifies clock generator settings. This is the preferred clock source for general debugging.

When the MCU is reset in active background (halt) mode, CLKSW is set which selects the bus clock as the source of the BDC clock. This CLKSW setting is most commonly used during flash memory programming because the bus clock can usually be configured to operate at the highest allowed bus frequency to ensure the fastest possible flash programming times. Because the host system is in control of changes to clock generator settings, it knows when a different BDC communication speed should be used. The host programmer also knows that no unexpected change in bus frequency could occur to disrupt BDC communications.

Normally, setting CLKSW should not be used for general debugging because there is no way to ensure the application program does not change the clock generator settings. This is especially true in the case of application programs that are not yet fully debugged.

After any reset (or at any other time), the host system can issue a SYNC command to determine the speed of the BDC clock. CLKSW may be written using the serial WRITE_XCSR_BYTE command through the BDC interface. CLKSW is located in the special XCSR byte register in the BDC module and it is not accessible in the normal memory map of the ColdFire core. This means that no program running on the processor can modify this register (intentionally or unintentionally).

The BKGD pin can receive a high- or low-level or transmit a high- or low-level. The following diagrams show timing for each of these cases. Interface timing is synchronous to clocks in the target BDC, but asynchronous to the external host. The internal BDC clock signal is shown for reference in counting cycles.

The following figure shows an external host transmitting a logic 1 or 0 to the BKGD pin of a target MCU. The host is asynchronous to the target so there is a 0–1 cycle delay from the host-generated falling edge to where the target perceives the beginning of the bit time. Ten target BDC clock cycles later, the target senses the bit level on the BKGD pin. Typically, the host actively drives the pseudo-open-drain BKGD pin during host-to-target transmissions to speed up rising edges. Because the target does not drive the BKGD pin during the host-to-target transmission period, there is no need to treat the line as an open-drain signal during this period.

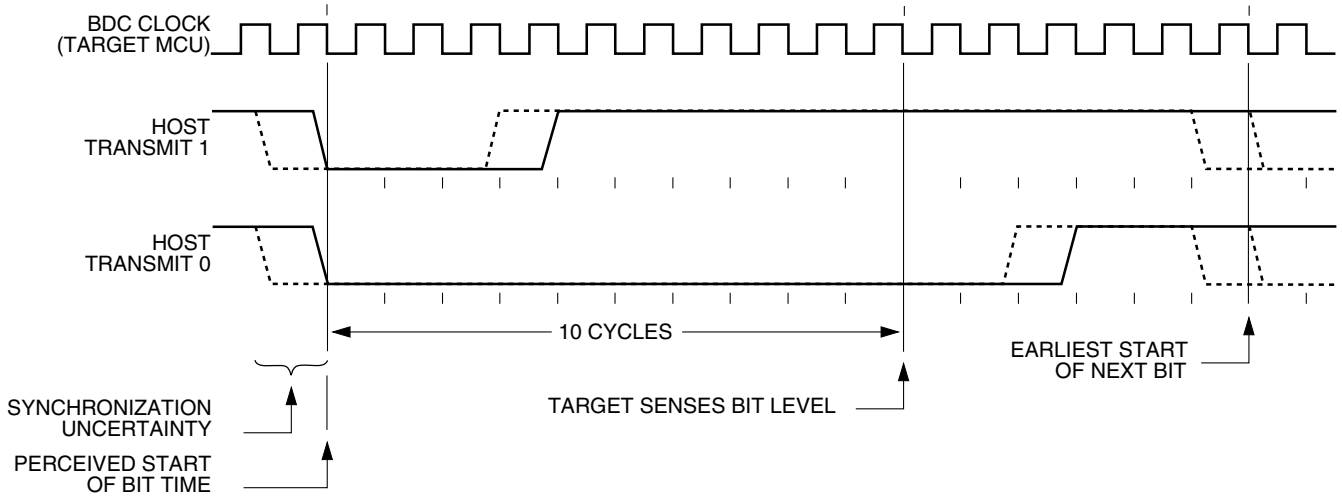


Figure 41-3. BDC Host-to-Target Serial Bit Timing

The following figure shows the host receiving a logic 1 from the target MCU. Because the host is asynchronous to the target MCU, there is a 0–1 cycle delay from the host-generated falling edge on BKGD to the perceived start of the bit time in the target MCU. The host holds the BKGD pin low long enough for the target to recognize it (at least two target BDC cycles). The host must release the low drive before the target MCU drives a brief active-high speedup pulse seven cycles after the perceived start of the bit time. The host should sample the bit level about 10 cycles after it started the bit time.

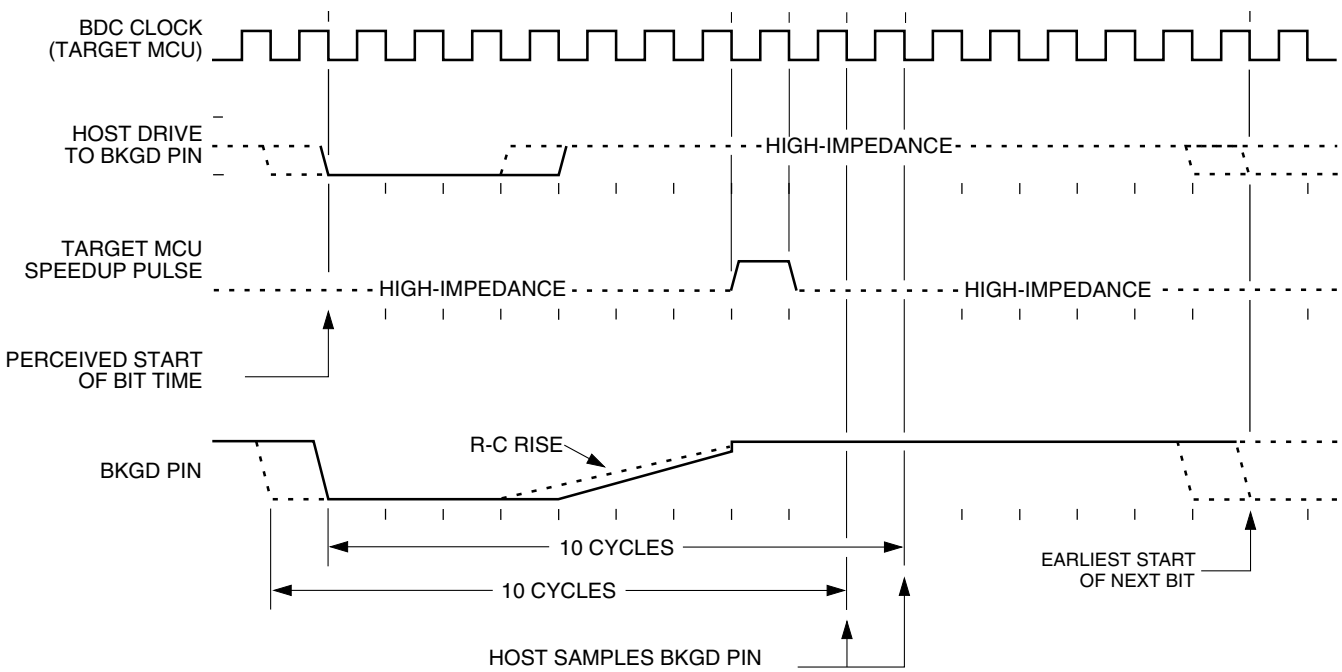


Figure 41-4. BDC Target-to-Host Serial Bit Timing (Logic 1)

The following figure shows the host receiving a logic 0 from the target MCU. Because the host is asynchronous to the target MCU, there is a 0–1 cycle delay from the host-generated falling edge on BKGD to the start of the bit time as perceived by the target

MCU. The host initiates the bit time, but the target MCU finishes it. Because the target wants the host to receive a logic 0, it drives the BKGD pin low for 13 BDC clock cycles, then briefly drives it high to speed up the rising edge. The host samples the bit level about 10 cycles after starting the bit time.

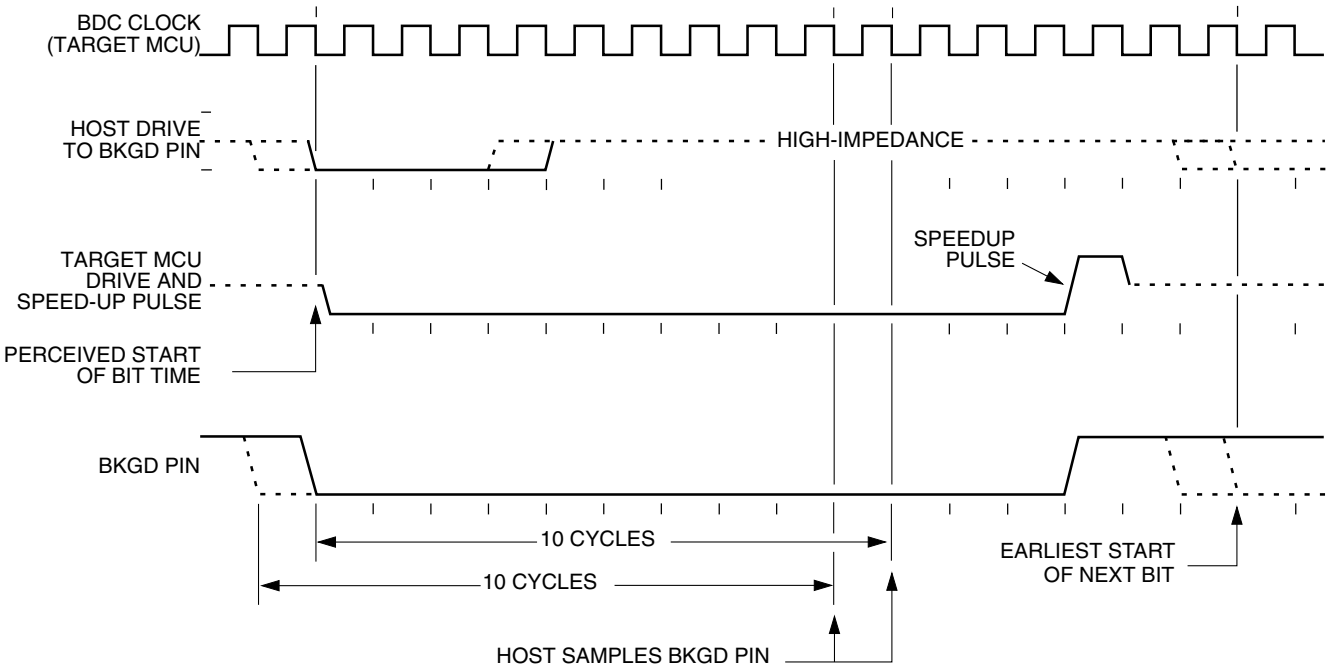


Figure 41-5. BDM Target-to-Host Serial Bit Timing (Logic 0)

41.4.1.4 BDM Command Set Descriptions

This section presents detailed descriptions of the BDM commands.

The V1 BDM command set is based on transmission of one or more 8-bit data packets per operation. Each operation begins with a host-to-target transmission of an 8-bit command code packet. The command code definition broadly maps the operations into four formats as shown in the following figure.

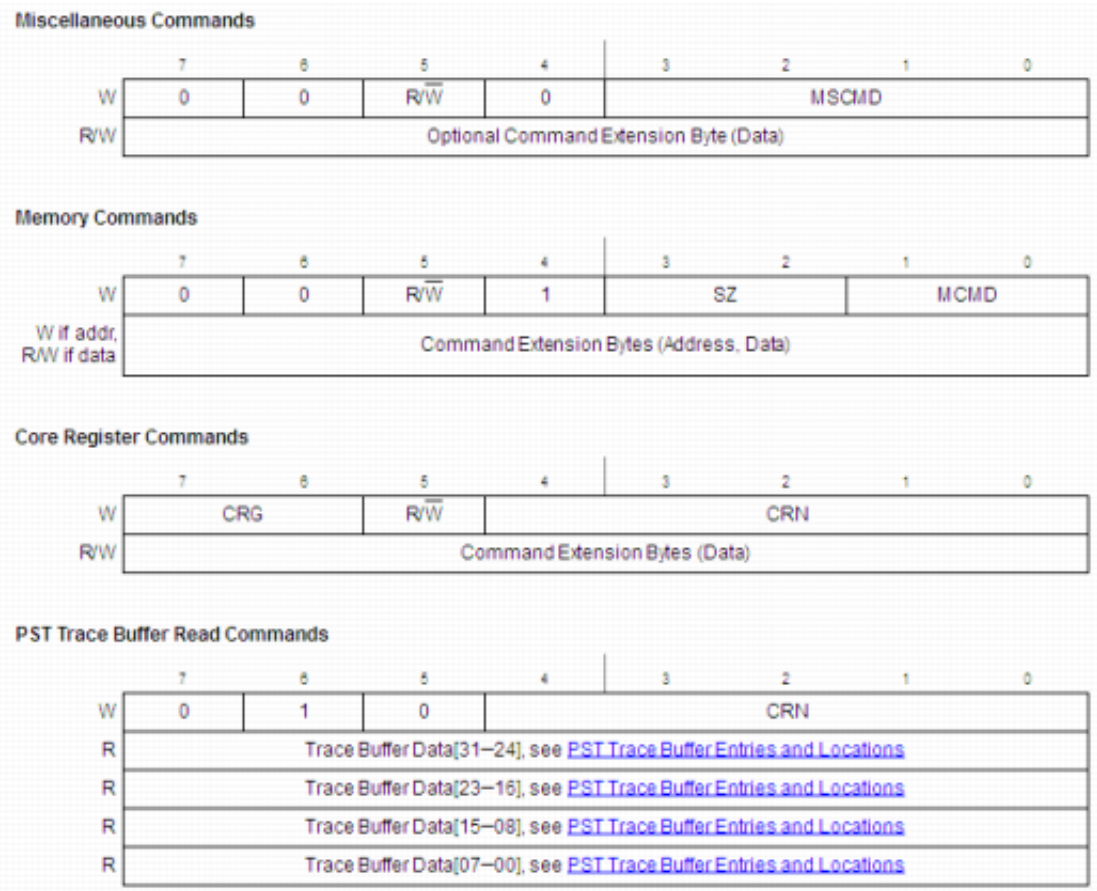


Figure 41-6. BDM Command Encodings

Table 41-39. BDM Command Field Descriptions

Field	Description
5 R/W	Read/Write. 0 Command is performing a write operation. 1 Command is performing a read operation.
3–0 MSCMD	Miscellaneous command. Defines the miscellaneous command to be performed. 0000 No operation 0001 Display the CPU's program counter (PC) plus optional capture in the PST trace buffer 0010 Enable the BDM acknowledge communication mode 0011 Disable the BDM acknowledge communication mode 0100 Force a CPU halt (background) 1000 Resume CPU execution (go) 1101 Read/write of the debug XCSR most significant byte 1110 Read/write of the debug CSR2 most significant byte 1111 Read/write of the debug CSR3 most significant byte

Table continues on the next page...

Table 41-39. BDM Command Field Descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>3–2
SZ</td><td>Memory operand size. Defines the size of the memory reference.
00 8-bit byte
01 16-bit word
10 32-bit long</td></tr>
<tr><td>1–0
MCMD</td><td>Memory command. Defines the type of the memory reference to be performed.
00 Simple write if R/W = 0; simple read if R/W = 1
01 Write + status if R/W = 0; read + status if R/W = 1
10 Fill if R/W = 0; dump if R/W = 1
11 Fill + status if R/W = 0; dump + status if R/W = 1</td></tr>
<tr><td>7–6
CRG</td><td>Core register group. Defines the core register group to be referenced.
01 CPU's general-purpose registers (An, Dn) or PST trace buffer
10 Debug's control registers
11 CPU's control registers (PC, SR, VBR, CPUCR,...)</td></tr>
<tr><td>4–0
CRN</td><td>Core register number. Defines the specific core register (its number) to be referenced. All other CRN values are reserved.
<table>
<tr><th>CRG</th><th>CRN</th><th>Register</th></tr>
<tr><td rowspan="3">01</td><td>0x00–0x07</td><td>D0–7</td></tr>
<tr><td>0x08–0x0F</td><td>A0–7</td></tr>
<tr><td>0x10–0x1B</td><td>PST Buffer 0–11</td></tr>
<tr><td>10</td><td colspan="2">DRc[4:0] as described in CSR</td></tr>
<tr><td rowspan="8">11</td><td>0x00</td><td>OTHER_A7</td></tr>
<tr><td>0x01</td><td>VBR</td></tr>
<tr><td>0x02</td><td>CPUCR</td></tr>
<tr><td>0x04</td><td>MACSR</td></tr>
<tr><td>0x05</td><td>MASK</td></tr>
<tr><td>0x06</td><td>ACC</td></tr>
<tr><td>0x0E</td><td>SR</td></tr>
<tr><td>0x0F</td><td>PC</td></tr>
</table></td></tr>
</table>

41.4.1.5 BDM Command Set Summary

The following table summarizes the BDM command set. Subsequent paragraphs contain detailed descriptions of each command. The nomenclature below is used in the following table to describe the structure of the BDM commands. Commands begin with an 8-bit hexadecimal command code in the host-to-target direction (most significant bit first)

```
/       =       separates parts of the command
d       =       delay 32 target BDC clock cycles
```

```
        ad24      =       24-bit memory address in the host-to-target direction
        rd8       =       8 bits of read data in the target-to-host direction
        rd16      =       16 bits of read data in the target-to-host direction
        rd32      =       32 bits of read data in the target-to-host direction
        rd.sz     =       read data, size defined by sz, in the target-to-host direction
        wd8       =       8 bits of write data in the host-to-target direction
        wd16      =       16 bits of write data in the host-to-target direction
        wd32      =       32 bits of write data in the host-to-target direction
        wd.sz     =       write data, size defined by sz, in the host-to-target direction
        ss        =       the contents of XCSR[31:24] in the target-to-host direction
(STATUS)
        sz        =       memory operand size (0b00 = byte, 0b01 = word, 0b10 = long)
        crn       =       core register number
        WS        =       command suffix signaling the operation is with status
```

Table 41-40. BDM Command Summary

Command Mnemonic	Command Classification	ACK if Enb? [1]	Command Structure	Description
SYNC	Always Available	N/A	N/A[2]	Request a timed reference pulse to determine the target BDC communication speed
ACK_DISABLE	Always Available	No	0x03/d	Disable the communication handshake. This command does not issue an ACK pulse.
ACK_ENABLE	Always Available	Yes	0x02/d	Enable the communication handshake. Issues an ACK pulse after the command is executed.
BACKGROUND	Non-Intrusive	Yes	0x04/d	Halt the CPU if ENBDM is set. Otherwise, ignore as illegal command.
DUMP_MEM.sz	Non-Intrusive	Yes	(0x32+4 x sz)/d/rd.sz	Dump (read) memory based on operand size (sz). Used with READ_MEM to dump large blocks of memory. An initial READ_MEM is executed to set up the starting address of the block and to retrieve the first result. Subsequent DUMP_MEM commands retrieve sequential operands.
DUMP_MEM.sz_WS	Non-Intrusive	No	(0x33+4 x sz)/d/ss/rd.sz	Dump (read) memory based on operand size (sz) and report status. Used with READ_MEM{_WS} to dump large blocks of memory. An initial READ_MEM{_WS} is executed to set up the starting address of the block and to retrieve the first result. Subsequent DUMP_MEM{_WS} commands retrieve sequential operands.

Table continues on the next page...

Table 41-40. BDM Command Summary (continued)

Command Mnemonic	Command Classification	ACK if Enb? [1]	Command Structure	Description
FILL_MEM.sz	Non-Intrusive	Yes	(0x12+4 x sz)/wd.sz/d	Fill (write) memory based on operand size (sz). Used with WRITE_MEM to fill large blocks of memory. An initial WRITE_MEM is executed to set up the starting address of the block and to write the first operand. Subsequent FILL_MEM commands write sequential operands.
FILL_MEM.sz_WS	Non-Intrusive	No	(0x13+4 x sz)/wd.sz/d/ss	Fill (write) memory based on operand size (sz) and report status. Used with WRITE_MEM{_WS} to fill large blocks of memory. An initial WRITE_MEM{_WS} is executed to set up the starting address of the block and to write the first operand. Subsequent FILL_MEM{_WS} commands write sequential operands.
GO	Non-Intrusive	Yes	0x08/d	Resume the CPU's execution[3]
NOP	Non-Intrusive	Yes	0x00/d	No operation
READ_CREG	Active Background	Yes	(0xE0+CRN)/d/rd32	Read one of the CPU's control registers
READ_DREG	Non-Intrusive	Yes	(0xA0+CRN)/d/rd32	Read one of the debug module's control registers
READ_MEM.sz	Non-Intrusive	Yes	(0x30+4 x sz)/ad24/d/rd.sz	Read the appropriately-sized (sz) memory value from the location specified by the 24-bit address
READ_MEM.sz_WS	Non-Intrusive	No	(0x31+4 x sz)/ad24/d/ss/ rd.sz	Read the appropriately-sized (sz) memory value from the location specified by the 24-bit address and report status
READ_PSTB	Non-Intrusive	Yes	(0x40+CRN)/d/rd32	Read the requested longword location from the PST trace buffer
READ_Rn	Active Background	Yes	(0x60+CRN)/d/rd32	Read the requested general-purpose register (An, Dn) from the CPU
READ_XCSR_BYTE	Always Available	No	0x2D/rd8	Read the most significant byte of the debug module's XCSR
READ_CSR2_BYTE	Always Available	No	0x2E/rd8	Read the most significant byte of the debug module's CSR2
READ_CSR3_BYTE	Always Available	No	0x2F/rd8	Read the most significant byte of the debug module's CSR3
SYNC_PC	Non-Intrusive	Yes	0x01/d	Display the CPU's current PC and capture it in the PST trace buffer
WRITE_CREG	Active Background	Yes	(0xC0+CRN)/wd32/d	Write one of the CPU's control registers

Table continues on the next page...

Table 41-40. BDM Command Summary (continued)

Command Mnemonic	Command Classification	ACK if Enb? [1]	Command Structure	Description
WRITE_DREG	Non-Intrusive	Yes	(0x80+CRN)/wd32/d	Write one of the debug module's control registers
WRITE_MEM.sz	Non-Intrusive	Yes	(0x10+4 x sz)/ad24/wd.sz/d	Write the appropriately-sized (sz) memory value to the location specified by the 24-bit address
WRITE_MEM.sz_WS	Non-Intrusive	No	(0x11+4 x sz)/ad24/wd.sz/d/ss	Write the appropriately-sized (sz) memory value to the location specified by the 24-bit address and report status
WRITE_Rn	Active Background	Yes	(0x40+CRN)/wd32/d	Write the requested general-purpose register (An, Dn) of the CPU
WRITE_XCSR_BYTE	Always Available	No	0x0D/wd8	Write the most significant byte of the debug module's XCSR
WRITE_CSR2_BYTE	Always Available	No	0x0E/wd8	Write the most significant byte of the debug module's CSR2
WRITE_CSR3_BYTE	Always Available	No	0x0F/wd8	Write the most significant byte of the debug module's CSR3

1. This column identifies if the command generates an ACK pulse if operating with acknowledge mode enabled. See ACK_ENABLE , for addition information.
2. The SYNC command is a special operation which does not have a command code.
3. If a GO command is received while the processor is not halted, it performs no operation.

41.4.1.5.1 SYNC

The SYNC command is unlike other BDC commands because the host does not necessarily know the correct speed to use for serial communications until after it has analyzed the response to the SYNC command.

To issue a SYNC command, the host:

1. Drives the BKGD pin low for at least 128 cycles of the slowest possible BDC clock (bus clock or device-specific alternate clock source).
2. Drives BKGD high for a brief speed-up pulse to get a fast rise time. (This speedup pulse is typically one cycle of the host clock which is as fast as the maximum target BDC clock.)
3. Removes all drive to the BKGD pin so it reverts to high impedance.
4. Listens to the BKGD pin for the sync response pulse.

Upon detecting the sync request from the host (which is a much longer low time than would ever occur during normal BDC communications), the target:

1. Waits for BKGD to return to a logic high.
2. Delays 16 cycles to allow the host to stop driving the high speed-up pulse.
3. Drives BKGD low for 128 BDC clock cycles.
4. Drives a 1-cycle high speed-up pulse to force a fast rise time on BKGD.
5. Removes all drive to the BKGD pin so it reverts to high impedance.

The host measures the low time of this 128-cycle sync response pulse and determines the correct speed for subsequent BDC communications. Typically, the host can determine the correct communication speed within a few percent of the actual target speed and the serial protocol can easily tolerate this speed error.

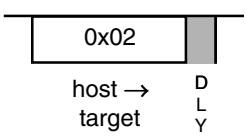
41.4.1.5.2 ACK_DISABLE

Disable host/target handshake protocol **Always Available**

Disables the serial communication handshake protocol. The subsequent commands, issued after the ACK_DISABLE command, do not execute the hardware handshake protocol. This command is not followed by an ACK pulse.

41.4.1.5.3 ACK_ENABLE

Enable host/target handshake protocol **Always Available**



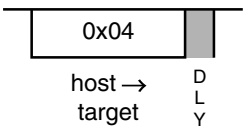
Enables the hardware handshake protocol in the serial communication. The hardware handshake is implemented by an acknowledge (ACK) pulse issued by the target MCU in response to a host command. The ACK_ENABLE command is interpreted and executed in the BDC logic without the need to interface with the CPU. However, an acknowledge (ACK) pulse is issued by the target device after this command is executed. This feature can be used by the host to evaluate if the target supports the hardware handshake

protocol. If the target supports the hardware handshake protocol, subsequent commands are enabled to execute the hardware handshake protocol; otherwise, this command is ignored by the target.

For additional information about the hardware handshake protocol, refer to Serial Interface Hardware Handshake Protocol and Hardware Handshake Abort Procedure.

41.4.1.5.4 BACKGROUND

Enter active background mode (if enabled) **Non-intrusive**



Provided XCSR[ENBDM] is set (BDM enabled), the BACKGROUND command causes the target MCU to enter active background (halt) mode as soon as the current CPU instruction finishes. If ENBDM is cleared (its default value), the BACKGROUND command is ignored.

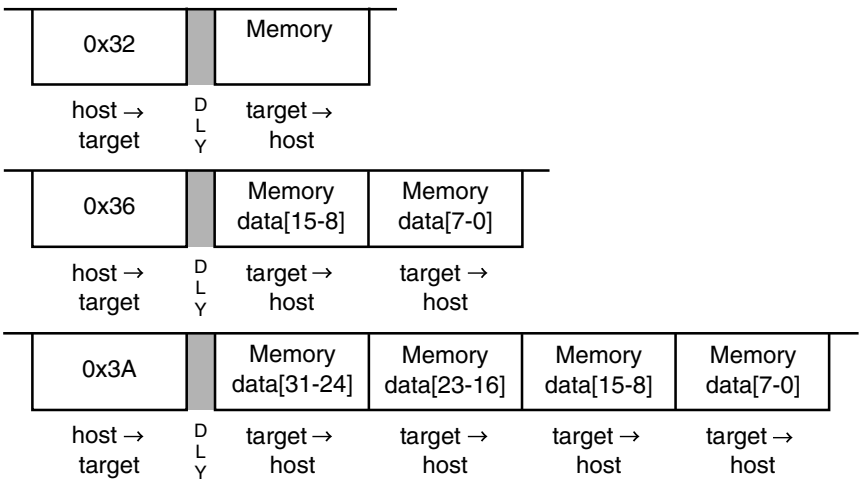
A delay of 32 BDC clock cycles is required after the BACKGROUND command to allow the target MCU to finish its current CPU instruction and enter active background mode before a new BDC command can be accepted.

After the target MCU is reset into a normal operating mode, the host debugger must send a WRITE_XCSR_BYTE command to set ENBDM before attempting to send the BACKGROUND command the first time. Normally, the development host sets ENBDM once at the beginning of a debug session or after a target system reset, and then it leaves the ENBDM bit set during debugging operations. During debugging, the host uses GO commands to move from active background mode to normal user program execution and uses BACKGROUND commands or breakpoints to return to active background mode.

41.4.1.5.5 DUMP_MEM.sz, DUMP_MEM.sz_WS

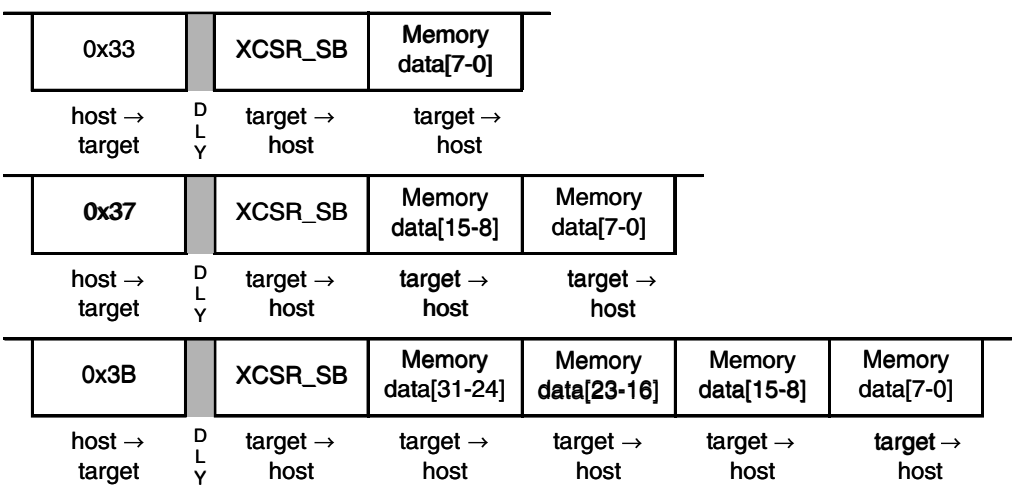
DUMP_MEM.sz

Read memory specified by debug address register, then increment address **Non-intrusive**



DUMP_MEM.sz_WS

Read memory specified by debug address register with status, then increment address **Non-intrusive**



DUMP_MEM{_WS} is used with the READ_MEM{_WS} command to access large blocks of memory. An initial READ_MEM{_WS} is executed to set-up the starting address of the block and to retrieve the first result. If an initial READ_MEM{_WS} is not executed before the first DUMP_MEM{_WS}, an illegal command response is returned. The DUMP_MEM{_WS} command retrieves subsequent operands. The initial address is incremented by the operand size (1, 2, or 4) and saved in a temporary register. Subsequent DUMP_MEM{_WS} commands use this address, perform the memory read,

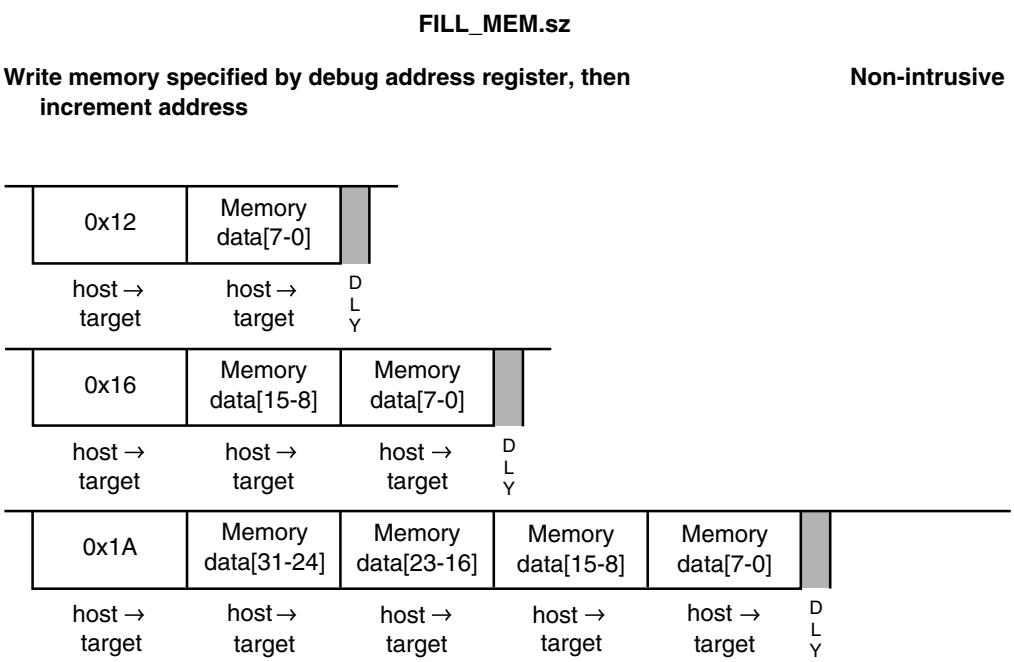
increment it by the current operand size, and store the updated address in the temporary register. If the with-status option is specified, the core status byte (XCSR_SB) contained in XCSR[31–24] is returned before the read data. XCSR_SB reflects the state after the memory read was performed.

Note

DUMP_MEM{_WS} does not check for a valid address; it is a valid command only when preceded by NOP, READ_MEM{_WS}, or another DUMP_MEM{_WS} command. Otherwise, an illegal command response is returned. NOP can be used for inter-command padding without corrupting the address pointer.

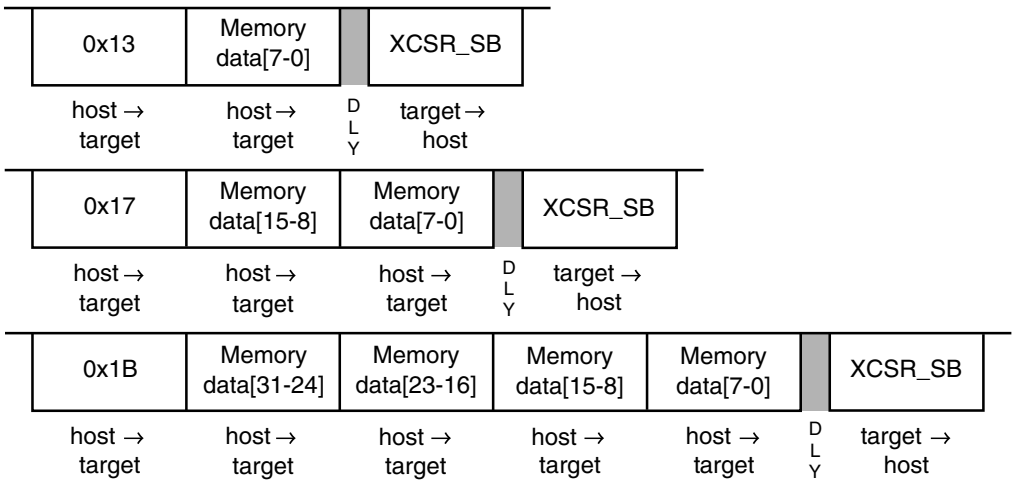
The size field (sz) is examined each time a DUMP_MEM{_WS} command is processed, allowing the operand size to be dynamically altered. The examples show the DUMP_MEM.B{_WS}, DUMP_MEM.W{_WS} and DUMP_MEM.L{_WS} commands.

41.4.1.5.6 FILL_MEM.sz, FILL_MEM.sz_WS



FILL_MEM.sz_WS

Write memory specified by debug address register with status, then increment address **Non-intrusive**



FILL_MEM{_WS} is used with the WRITE_MEM{_WS} command to access large blocks of memory. An initial WRITE_MEM{_WS} is executed to set up the starting address of the block and write the first datum. If an initial WRITE_MEM{_WS} is not executed before the first FILL_MEM{_WS}, an illegal command response is returned. The FILL_MEM{_WS} command stores subsequent operands. The initial address is incremented by the operand size (1, 2, or 4) and saved in a temporary register. Subsequent FILL_MEM{_WS} commands use this address, perform the memory write, increment it by the current operand size, and store the updated address in the temporary register. If the with-status option is specified, the core status byte (XCSR_SB) contained in XCSR[31–24] is returned after the write data. XCSR_SB reflects the state after the memory write was performed.

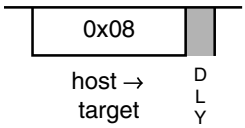
Note

FILL_MEM{_WS} does not check for a valid address; it is a valid command only when preceded by NOP, WRITE_MEM{_WS}, or another FILL_MEM{_WS} command. Otherwise, an illegal command response is returned. NOP can be used for intercommand padding without corrupting the address pointer.

The size field (sz) is examined each time a FILL_MEM{_WS} command is processed, allowing the operand size to be dynamically altered. The examples show the FILL_MEM.B{_WS}, FILL_MEM.W{_WS} and FILL_MEM.L{_WS} commands.

41.4.1.5.7 GO

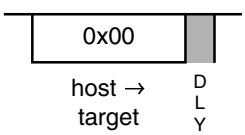
Go **Non-intrusive**



This command is used to exit active background (halt) mode and begin (or resume) execution of the application's instructions. The CPU's pipeline is flushed and refilled before normal instruction execution resumes. Prefetching begins at the current address in the PC and at the current privilege level. If any register (such as the PC or SR) is altered by a BDM command while the processor is halted, the updated value is used when prefetching resumes. If a GO command is issued and the CPU is not halted, the command is ignored.

41.4.1.5.8 NOP

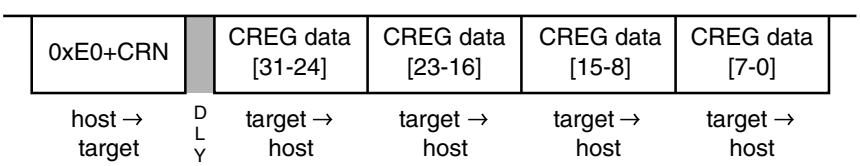
No operation **Non-intrusive**



NOP performs no operation and may be used as a null command where required.

41.4.1.5.9 READ_CREG

Read CPU control register **Active Background**

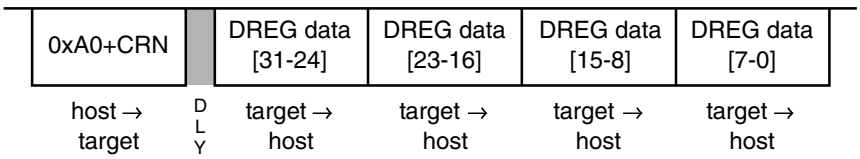


If the processor is halted, this command reads the selected control register and returns the 32-bit result. This register grouping includes the PC, SR, CPUCR, MACSR, MASK, ACC, VBR, and OTHER_A7. Accesses to processor control registers are always 32-bits wide, regardless of implemented register width. The register is addressed through the core register number (CRN). See Table 41-39 for the CRN details when CRG is 11.

If the processor is not halted, this command is rejected as an illegal operation and no operation is performed.

41.4.1.5.10 READ_DREG

Read debug control register **Non-intrusive**

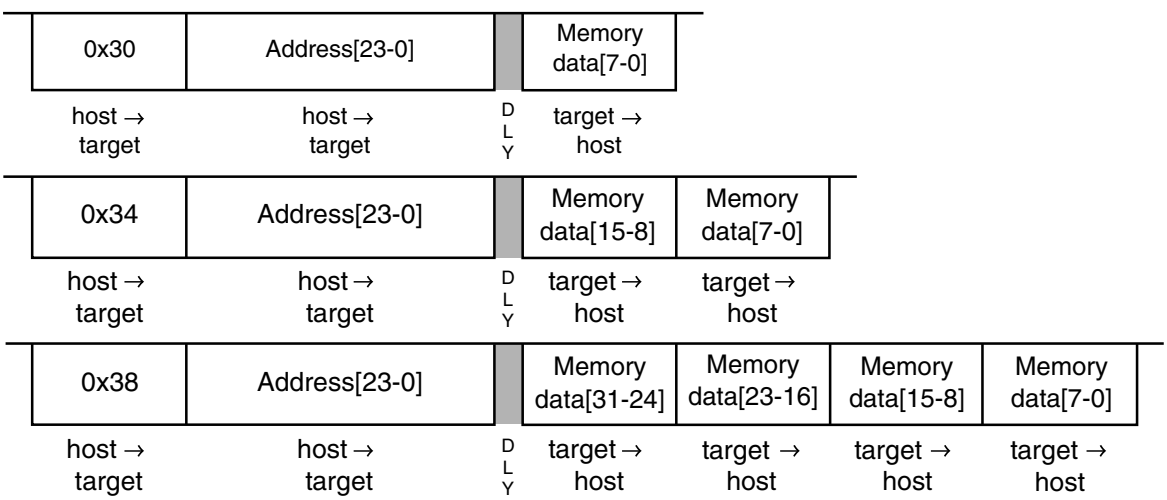


This command reads the selected debug control register and returns the 32-bit result. This register grouping includes the CSR, XCSR, CSR2, and CSR3. Accesses to debug control registers are always 32-bits wide, regardless of implemented register width. The register is addressed through the core register number (CRN). See Table 41-4 for CRN details.

41.4.1.5.11 READ_MEM.sz, READ_MEM.sz_WS

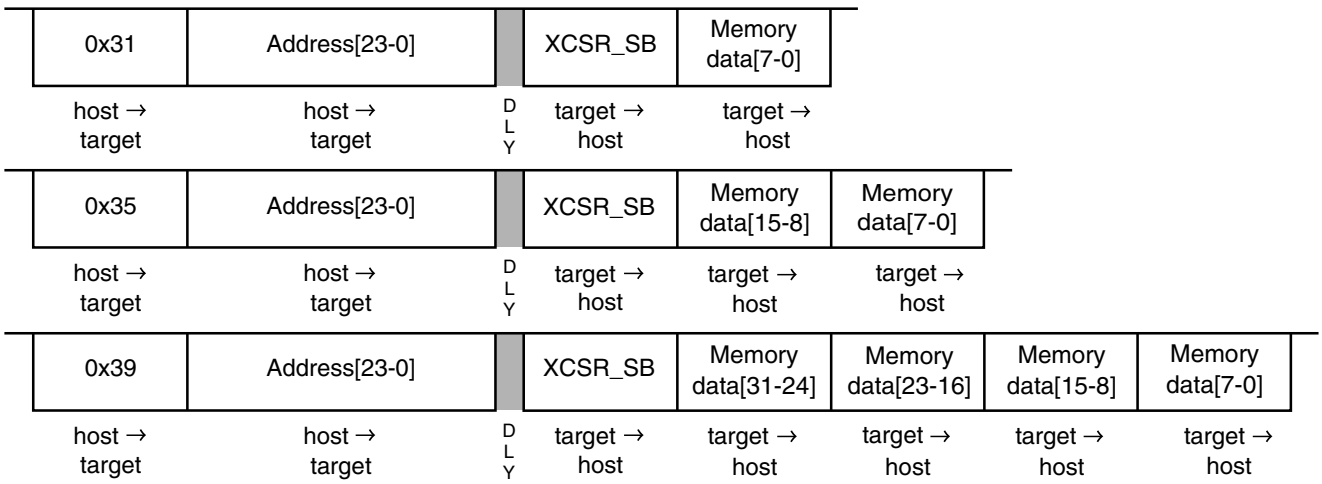
READ_MEM.sz

Read memory at the specified address **Non-intrusive**



READ_MEM.sz_WS

Read memory at the specified address with status **Non-intrusive**

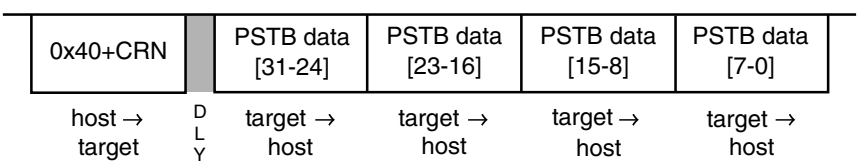


Read data at the specified memory address. The reference address is transmitted as three 8-bit packets (msb to lsb) immediately after the command packet. The access attributes are defined by BAAR[TT,TM]. The hardware forces low-order address bits to zeros for word and longword accesses to ensure these accesses are on 0-modulo-size alignments. If the with-status option is specified, the core status byte (XCSR_SB) contained in XCSR[31–24] is returned before the read data. XCSR_SB reflects the state after the memory read was performed.

The examples show the READ_MEM.B{_WS}, READ_MEM.W{_WS} and READ_MEM.L{_WS} commands.

41.4.1.5.12 READ_PSTB

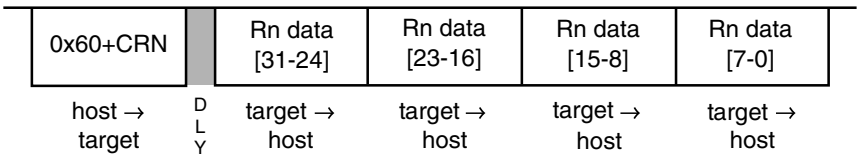
Read PST trace buffer at the specified address **Non-intrusive**



Read 32 bits of captured PST/DDATA values from the trace buffer at the specified address. The PST trace buffer contains 64 six-bit entries, packed consecutively into 12 longword locations. See Table 41-37 for an illustration of how the buffer entries are packed.

41.4.1.5.13 READ_Rn

Read general-purpose CPU register **Active Background**

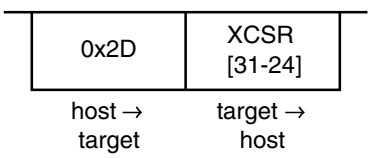


If the processor is halted, this command reads the selected CPU general-purpose register (An, Dn) and returns the 32-bit result. See Table 41-39 for the CRN details when CRG is 01.

If the processor is not halted, this command is rejected as an illegal operation and no operation is performed.

41.4.1.5.14 READ_XCSR_BYTE

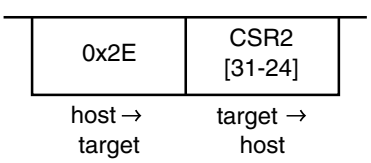
Read XCSR Status Byte **Always Available**



Read the special status byte of XCSR (XCSR[31–24]). This command can be executed in any mode.

41.4.1.5.15 READ_CSR2_BYTE

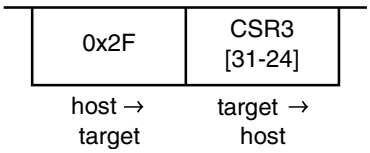
Read CSR2 Status Byte **Always Available**



Read the most significant byte of CSR2 (CSR2[31–24]). This command can be executed in any mode.

41.4.1.5.16 READ_CSR3_BYTE

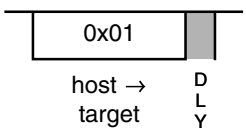
Read CSR2 Status Byte **Always Available**



Read the most significant byte of the CSR3 (CSR3[31–24]). This command can be executed in any mode.

41.4.1.5.17 SYNC_PC

Synchronize PC to PST/DDATA Signals **Non-intrusive**



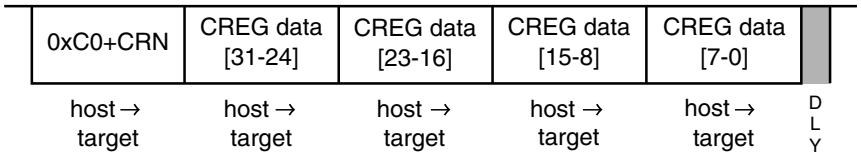
Capture the processor's current PC (program counter) and display it on the PST/DDATA signals. After the debug module receives the command, it sends a signal to the ColdFire core that the current PC must be displayed. The core responds by forcing an instruction fetch to the next PC with the address being captured by the DDATA logic. The DDATA logic captures a 2- or 3-byte instruction address, based on CSR[9]. If CSR[9] is cleared, then a 2-byte address is captured, else a 3-byte address is captured. The specific sequence of PST and DDATA values is defined as:

1. Debug signals a SYNC_PC command is pending.
2. CPU completes the current instruction.
3. CPU forces an instruction fetch to the next PC, generating a PST = 0x5 value indicating a taken branch. DDATA captures the instruction address corresponding to the PC. DDATA generates a PST marker signalling a 2- or 3-byte address as defined by CSR[9] (CSR[9] = 0, 2-byte; CSR[9] = 1, 3-byte) and displays the captured PC address.

This command can be used to provide a PC synchronization point between the core's execution and the application code in the PST trace buffer. It can also be used to dynamically access the PC for performance monitoring as the execution of this command is considerably less intrusive to the real-time operation of an application than a BACKGROUND/read-PC/GO command sequence.

41.4.1.5.18 WRITE_CREG

Write CPU control register **Active Background**

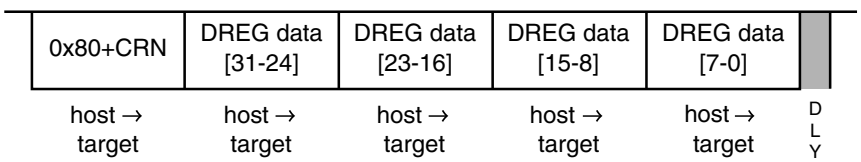


If the processor is halted, this command writes the 32-bit operand to the selected control register. This register grouping includes the PC, SR, CPUCR, MACSR, MASK, ACC, VBR, and OTHER_A7. Accesses to processor control registers are always 32-bits wide, regardless of implemented register width. The register is addressed through the core register number (CRN). See Table 41-39 for the CRN details when CRG is 11.

If the processor is not halted, this command is rejected as an illegal operation and no operation is performed.

41.4.1.5.19 WRITE_DREG

Write debug control register **Non-intrusive**



This command writes the 32-bit operand to the selected debug control register. This grouping includes all the debug control registers ({X}CSR*n*, BAAR, AATR, TDR, PBR*n*, PBMR, AB*x*R, DBR, DBMR). Accesses to debug control registers are always 32-bits wide, regardless of implemented register width. The register is addressed through the core register number (CRN).

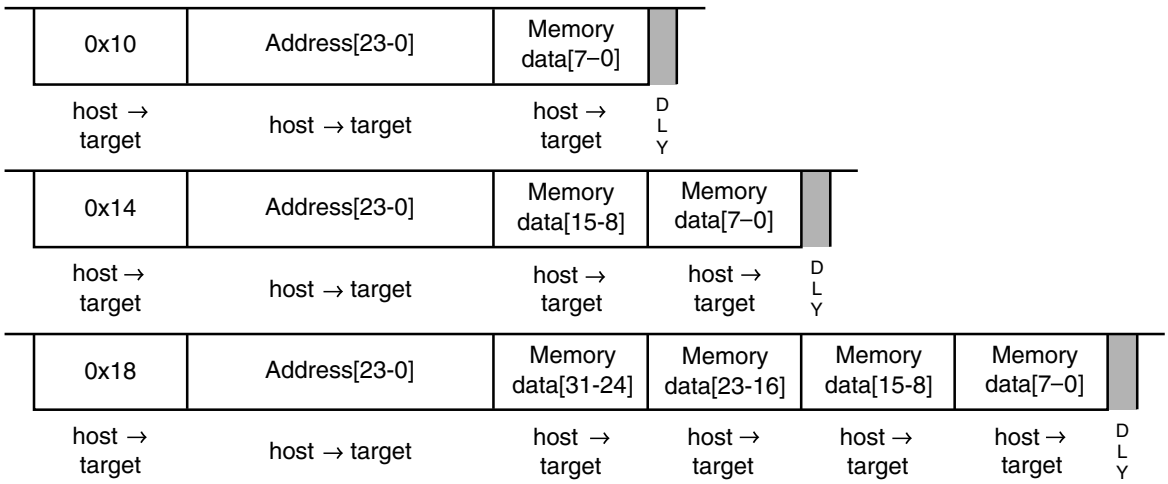
Note

When writing XCSR, CSR2, or CSR3, WRITE_DREG only writes bits 23–0. The upper byte of these debug registers is only written with the special WRITE_XCSR_BYTE, WRITE_CSR2_BYTE, and WRITE_CSR3_BYTE commands.

41.4.1.5.20 WRITE_MEM.sz, WRITE_MEM.sz_WS

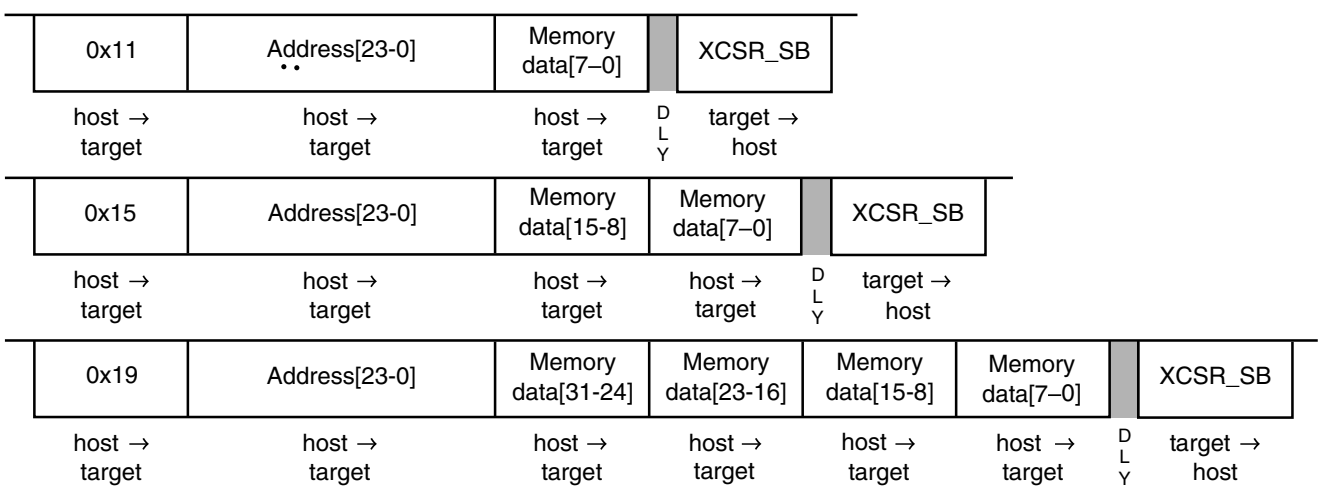
WRITE_MEM.sz

Write memory at the specified address **Non-intrusive**



WRITE_MEM.sz_WS

Write memory at the specified address with status **Non-intrusive**

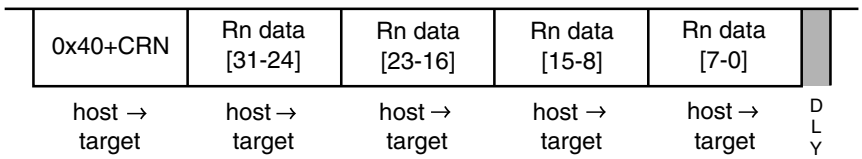


Write data at the specified memory address. The reference address is transmitted as three 8-bit packets (msb to lsb) immediately after the command packet. The access attributes are defined by BAAR[TT,TM]. The hardware forces low-order address bits to zeros for word and longword accesses to ensure these accesses are on 0-modulo-size alignments. If the with-status option is specified, the core status byte (XCSR_SB) contained in XCSR[31–24] is returned after the read data. XCSR_SB reflects the state after the memory write was performed.

The examples show the WRITE_MEM.B{_WS}, WRITE_MEM.W{_WS}, and WRITE_MEM.L{_WS} commands.

41.4.1.5.21 WRITE_Rn

Write general-purpose CPU register **Active Background**

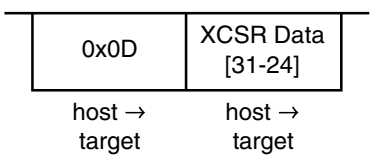


If the processor is halted, this command writes the 32-bit operand to the selected CPU general-purpose register (An, Dn). See Table 41-39 for the CRN details when CRG is 01.

If the processor is not halted, this command is rejected as an illegal operation and no operation is performed.

41.4.1.5.22 WRITE_XCSR_BYTE

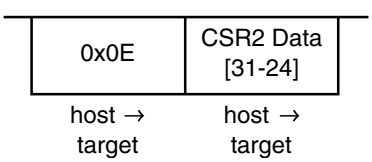
Write XCSR Status Byte **Always Available**



Write the special status byte of XCSR (XCSR[31–24]). This command can be executed in any mode.

41.4.1.5.23 WRITE_CSR2_BYTE

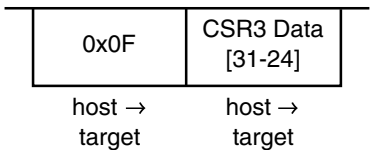
Write CSR2 Status Byte **Always Available**



Write the most significant byte of CSR2 (CSR2[31–24]). This command can be executed in any mode.

41.4.1.5.24 WRITE_CSR3_BYTE

Write CSR3 Status Byte **Always Available**



Write the most significant byte of CSR3 (CSR3[31–24]). This command can be executed in any mode.

41.4.1.5.25 BDM Accesses of the MAC Registers

The presence of rounding logic in the output datapath of the MAC requires special care for BDM-initiated reads and writes of its programming model. In particular, any result rounding modes must be disabled during the read/write process so the exact bit-wise MAC register contents are accessed.

For example, a BDM read of the accumulator (ACC) must be preceded by two commands accessing the MAC status register, as shown in the following sequence:

```
BdmReadACC (
        rcreg    macsr;             // read current macsr contents and save
        wcreg    #0,macsr;          // disable all rounding modes
        rcreg    ACC;               // read the desired accumulator
        wcreg    #saved_data,macsr; // restore the original macsr
)
```

Likewise, to write an accumulator register, the following BDM sequence is needed:

```
BdmWriteACC (
        rcreg     macsr;             // read current macsr contents and save
        wcreg     #0,macsr;          // disable all rounding modes
        wcreg     #data,ACC;         // write the desired accumulator
        wcreg     #saved_data,macsr; // restore the original macsr
```

For more information on saving and restoring the complete MAC programming model, see <<create reference to "Saving and Restoring the EMAC Programming Model" section>>.

41.4.1.6 Serial Interface Hardware Handshake Protocol

BDC commands that require CPU execution are ultimately treated at the core clock rate. Because the BDC clock source can be asynchronous relative to the bus frequency when CLKSW is cleared, it is necessary to provide a handshake protocol so the host can determine when an issued command is executed by the CPU. This section describes this protocol.

The hardware handshake protocol signals to the host controller when an issued command was successfully executed by the target. This protocol is implemented by a low pulse (16 BDC clock cycles) followed by a brief speedup pulse on the BKGD pin, generated by the target MCU when a command, issued by the host, has been successfully executed. See Figure 41-7. This pulse is referred to as the ACK pulse. After the ACK pulse is finished, the host can start the data-read portion of the command if the last-issued command was a read command, or start a new command if the last command was a write command or a control command (BACKGROUND, GO, NOP, SYNC_PC). The ACK pulse is not issued earlier than 32 BDC clock cycles after the BDC command was issued. The end of the BDC command is assumed to be the 16th BDC clock cycle of the last bit. This minimum delay assures enough time for the host to recognize the ACK pulse. There is no upper limit for the delay between the command and the related ACK pulse, because the command execution depends on the CPU bus frequency, which in some cases could be slow compared to the serial communication rate. This protocol allows great flexibility for pod designers, because it does not rely on any accurate time measurement or short response time to any event in the serial communication.

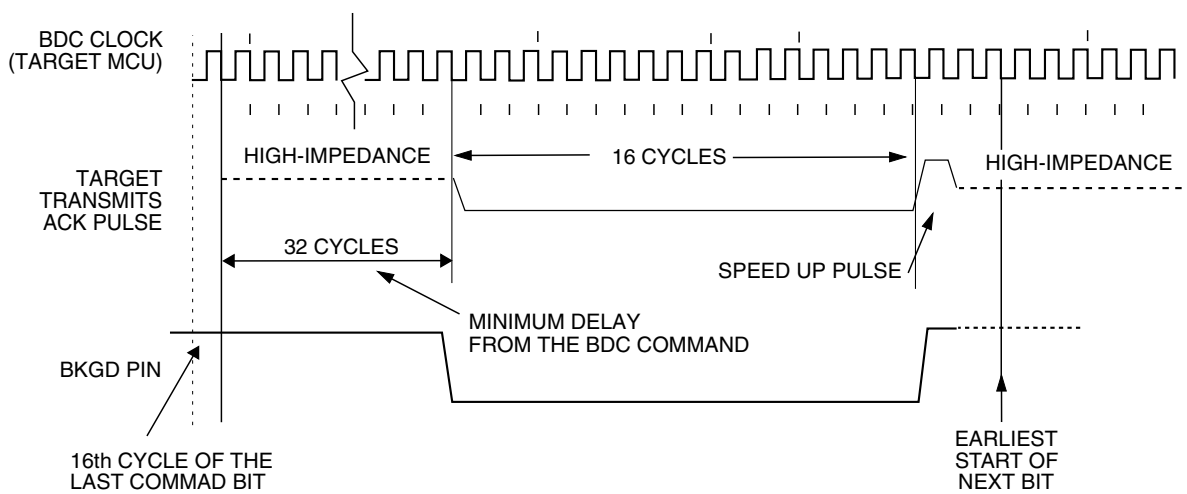


Figure 41-7. Target Acknowledge Pulse (ACK)

Note

If the ACK pulse was issued by the target, the host assumes the previous command was executed. If the CPU enters a stop mode prior to executing a non-intrusive command, the

command is discarded and the ACK pulse is not issued. After entering a stop mode, the BDC command is no longer pending and the XCSR[CSTAT] value of 001 is kept until the next command is successfully executed.

Figure 41-8 shows the ACK handshake protocol in a command level timing diagram. A READ_MEM.B command is used as an example:

1. The 8-bit command code is sent by the host, followed by the address of the memory location to be read.
2. The target BDC decodes the command and sends it to the CPU.
3. Upon receiving the BDC command request, the CPU schedules a execution slot for the command.
4. The CPU temporarily stalls the instruction stream at the scheduled point, executes the READ_MEM.B command and then continues.

This process is referred to as cycle stealing. The READ_MEM.B appears as a single-cycle operation to the processor, even though the pipelined nature of the Operand Execution Pipeline requires multiple CPU clock cycles for it to actually complete. After that, the debug module tracks the execution of the READ_MEM.b command as the processor resumes the normal flow of the application program. After detecting the READ_MEM.B command is done, the BDC issues an ACK pulse to the host controller, indicating that the addressed byte is ready to be retrieved. After detecting the ACK pulse, the host initiates the data-read portion of the command.

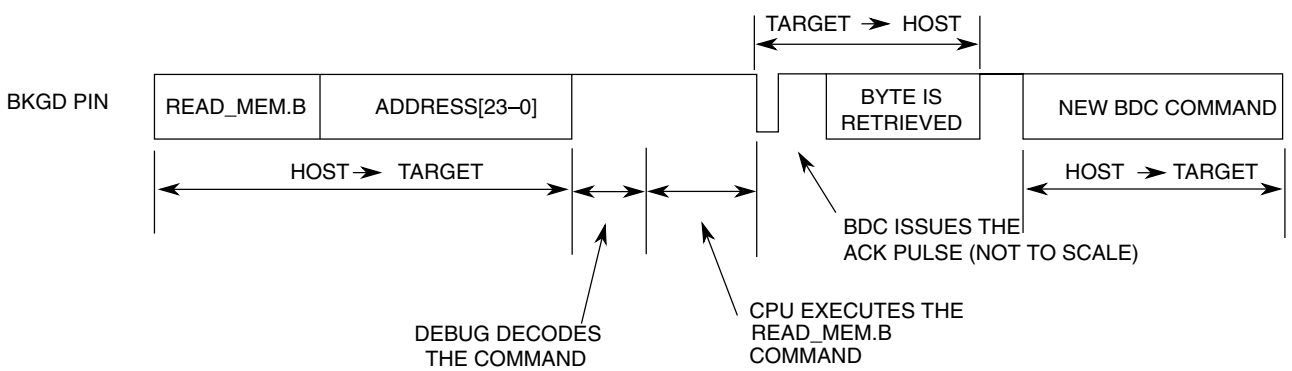


Figure 41-8. Handshake Protocol at Command Level

Unlike a normal bit transfer, where the host initiates the transmission by issuing a negative edge in the BKGD pin, the serial interface ACK handshake pulse is initiated by the target MCU. The hardware handshake protocol in Figure 41-8 specifies the timing when the BKGD pin is being driven, so the host should follow these timing relationships to avoid the risks of an electrical conflict at the BKGD pin.

The ACK handshake protocol does not support nested ACK pulses. If a BDC command is not acknowledged by an ACK pulse, the host first needs to abort the pending command before issuing a new BDC command. When the CPU enters a stop mode at about the same time the host issues a command that requires CPU execution, the target discards the incoming command. Therefore, the command is not acknowledged by the target, meaning that the ACK pulse is not issued in this case. After a certain time, the host could decide to abort the ACK protocol to allow a new command. Therefore, the protocol provides a mechanism where a command (a pending ACK) could be aborted. Unlike a regular BDC command, the ACK pulse does not provide a timeout. In the case of a STOP instruction where the ACK is prevented from being issued, it would remain pending indefinitely if not aborted. See the handshake abort procedure described in Hardware Handshake Abort Procedure.

41.4.1.7 Hardware Handshake Abort Procedure

The abort procedure is based on the SYNC command. To abort a command that has not responded with an ACK pulse, the host controller generates a sync request (by driving BKGD low for at least 128 serial clock cycles and then driving it high for one serial clock cycle as a speedup pulse). By detecting this long low pulse on the BKGD pin, the target executes the sync protocol (see SYNC), and assumes that the pending command and therefore the related ACK pulse, are being aborted. Therefore, after the sync protocol completes, the host is free to issue new BDC commands.

Because the host knows the target BDC clock frequency, the SYNC command does not need to consider the lowest possible target frequency. In this case, the host could issue a SYNC close to the 128 serial clock cycles length, providing a small overhead on the pulse length to assure the sync pulse is not misinterpreted by the target.

It is important to notice that any issued BDC command that requires CPU execution is scheduled for execution by the pipeline based on the dynamic state of the machine, provided the processor does not enter any of the stop modes. If the host aborts a command by sending the sync pulse, it should then read XCSR[CSTAT] after the sync response is issued by the target, checking for CSTAT cleared, before attempting to send any new command that requires CPU execution. This prevents the new command from being discarded at the debug/CPU interface, due to the pending command being executed by the CPU. Any new command should be issued only after XCSR[CSTAT] is cleared.

There are multiple reasons that could cause a command to take too long to execute, measured in terms of the serial communication rate: The BDC clock frequency could be much faster than the CPU clock frequency, or the CPU could be accessing a slow memory, which would cause pipeline stall cycles to occur. All commands referencing the CPU registers or memory require access to the processor's local bus to complete. If the

processor is executing a tight loop contained within a single aligned longword, the processor may never successfully grant the internal bus to the debug command. For example:

```
          align     4
label1:   nop
          bra.b     label1
or
          align     4
label2:   bra.w     label2
```

These two examples of tight loops exhibit the BDM lockout behavior. If the loop spans across two longwords, there are no issues, so the recommended construct is:

```
          align     4
label3:   bra.l     label3
```

The hardware handshake protocol is appropriate for these situations, but the host could also decide to use the software handshake protocol instead. In this case, if XCSR[CSTAT] is 001, there is a BDC command pending at the debug/CPU interface. The host controller should monitor XCSR[CSTAT] and wait until it is 000 to be able to issue a new command that requires CPU execution. However, if the XCSR[CSTAT] is 1*xx*, the host should assume the last command failed to execute. To recover from this condition, the following sequence is suggested:

1. Issue a SYNC command to reset the BDC communication channel.
2. The host issues a BDM NOP command.
3. The host reads the channel status using a READ_XCSR_BYTE command.
4. If XCSR[CSTAT] is 000

   then the status is okay; proceed

   else

   Halt the CPU using a BDM BACKGROUND command
   Repeat steps 1,2,3
   If XCSR[CSTAT] is 000, then proceed, else reset the device

The following figure shows a SYNC command aborting a READ_MEM.B. After the command is aborted, a new command could be issued by the host.

Note

In the following figure, signal timing is not drawn to scale.

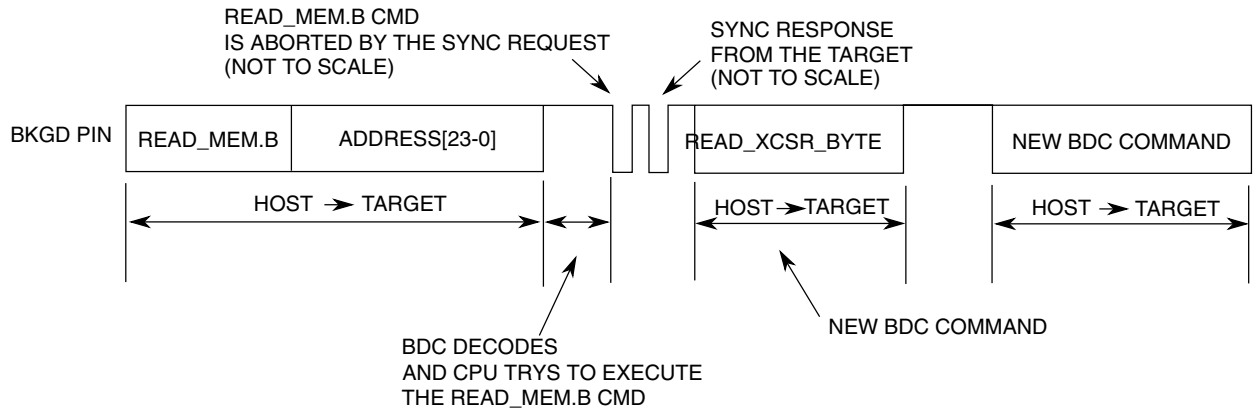


Figure 41-9. ACK Abort Procedure at the Command Level

The following figure shows a conflict between the ACK pulse and the sync request pulse. This conflict could occur if a pod device is connected to the target BKGD pin and the target is already executing a BDC command. Consider that the target CPU is executing a pending BDC command at the exact moment the pod is being connected to the BKGD pin. In this case, an ACK pulse is issued at the same time as the SYNC command. In this case there is an electrical conflict between the ACK speedup pulse and the sync pulse. Because this is not a probable situation, the protocol does not prevent this conflict from happening.

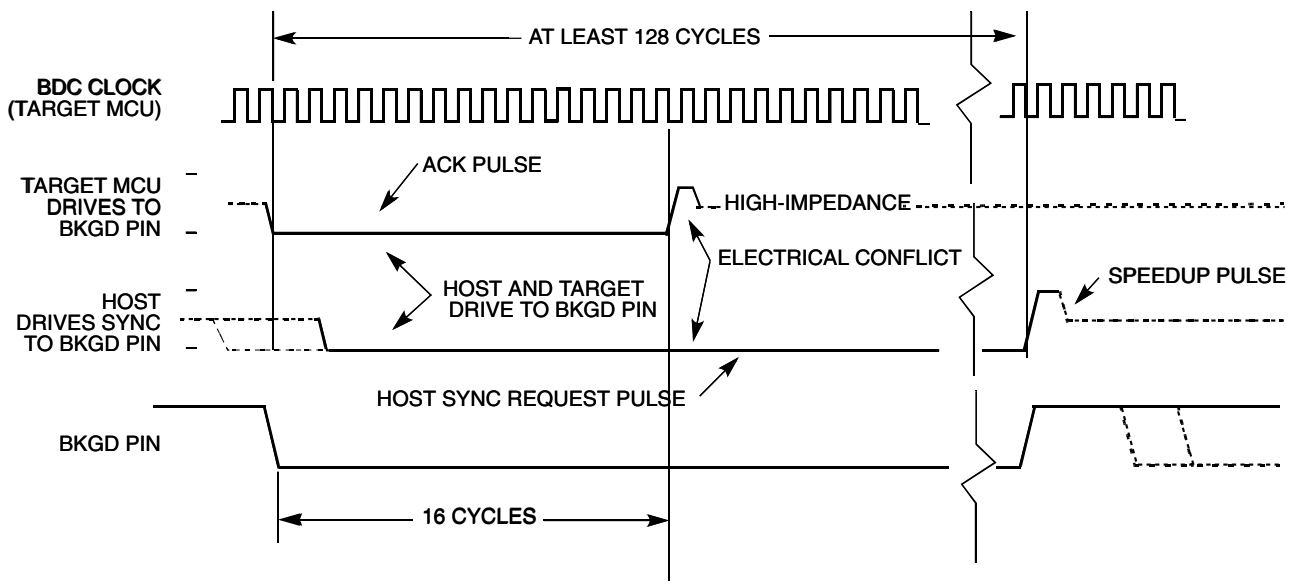


Figure 41-10. ACK Pulse and SYNC Request Conflict

The hardware handshake protocol is enabled by the ACK_ENABLE command and disabled by the ACK_DISABLE command. It also allows for pod devices to choose between the hardware handshake protocol or the software protocol that monitors the XCSR status byte. The ACK_ENABLE and ACK_DISABLE commands are:

- ACK_ENABLE — Enables the hardware handshake protocol. The target issues the ACK pulse when a CPU command is executed. The ACK_ENABLE command itself also has the ACK pulse as a response.
- ACK_DISABLE — Disables the ACK pulse protocol. In this case, the host should verify the state of XCSR[CSTAT] to evaluate if there are pending commands and to check if the CPU's operating state has changed to or from active background mode via XCSR[31–30].

The default state of the protocol, after reset, is hardware handshake protocol disabled.

The commands that do not require CPU execution, or that have the status register included in the retrieved bit stream, do not perform the hardware handshake protocol. Therefore, the target does not respond with an ACK pulse for those commands even if the hardware protocol is enabled. Conversely, only commands that require CPU execution and do not include the status byte perform the hardware handshake protocol. See the third column in Table 41-40 for the complete enumeration of this function.

An exception is the ACK_ENABLE command, which does not require CPU execution but responds with the ACK pulse. This feature can be used by the host to evaluate if the target supports the hardware handshake protocol. If an ACK pulse is issued in response to this command, the host knows that the target supports the hardware handshake protocol. If the target does not support the hardware handshake protocol the ACK pulse is not issued. In this case, the ACK_ENABLE command is ignored by the target, because it is not recognized as a valid command.

41.4.2 Real-Time Debug Support

The ColdFire family supports debugging real-time applications. For these types of embedded systems, the processor must continue to operate during debug. The foundation of this area of debug support is that while the processor cannot be halted to allow debugging, the system can generally tolerate the small intrusions with minimal effect on real-time operation.

Note

The details regarding real-time debug support will be supplied at a later time.

41.4.3 Trace Support

For the baseline V1 ColdFire core and its single debug signal, support for trace functionality is completely redefined. The V1 solution provides an on-chip PST/DDATA trace buffer (known as the PSTB) to record the stream of PST and DDATA values.

As a review, the classic ColdFire debug architecture supports real-time trace via the PST/DDATA output signals. For this functionality, the following apply:

- One (or more) PST value is generated for each executed instruction
- Branch target instruction address information is displayed on all non-PC-relative change-of-flow instructions, where the user selects a programmable number of bytes of target address
  - Displayed information includes PST marker plus target instruction address as DDATA
  - Captured address creates the appropriate number of DDATA entries, each with 4 bits of address
- Optional data trace capabilities are provided for accesses mapped to the slave peripheral bus
  - Displayed information includes PST marker plus captured operand value as DDATA
  - Captured operand creates the appropriate number of DDATA entries, each with 4 bits of data

The resulting PST/DDATA output stream, with the application program memory image, provides an instruction-by-instruction dynamic trace of the execution path.

Even with the application of a PST trace buffer, problems associated with the PST bandwidth and associated fill rate of the buffer remain. Given that there is one (or more) PST entry per instruction, the PSTB would fill rapidly without some type of data compression.

Consider the following example to illustrate the PST compression algorithm. Most sequential instructions generate a single PST = 1 value. Without compression, the execution of ten sequential instructions generates a stream of ten PST = 1 values. With PST compression, the reporting of any PST = 1 value is delayed so that consecutive PST = 1 values can be accumulated. When a PST ≠ 1 value is reported, the maximum accumulation count is reached, or a debug data value is captured, a single accumulated

PST value is generated. Returning to the example with compression enabled, the execution of ten sequential instructions generates a single PST value indicating ten sequential instructions have been executed.

This technique has proven to be effective at significantly reducing the average PST entries per instruction and PST entries per machine cycle. The application of this compression technique makes the application of a useful PST trace buffer for the V1 ColdFire core realizable. The resulting 5-bit PST definitions are shown in the following table.

Table 41-41. CF1 Debug Processor Status Encodings

PST[4:0]	Definition
0x00	Continue execution. Many instructions execute in one processor cycle. If an instruction requires more processor clock cycles, subsequent clock cycles are indicated by driving PST with this encoding.
0x01	Begin execution of one instruction. For most instructions, this encoding signals the first processor clock cycle of an instruction's execution. Certain change-of-flow opcodes, plus the PULSE and WDDATA instructions, generate different encodings.
0x02	Reserved
0x03	Entry into user-mode. Signaled after execution of the instruction that caused the ColdFire processor to enter user mode.
0x04	Begin execution of PULSE and WDDATA instructions. PULSE defines triggers or markers for debug and/or performance analysis. WDDATA lets the core write any operand (byte, word, or longword) directly to the DDATA port, independent of debug module configuration. When WDDATA is executed, a value of 0x04 is signaled on the PST port, followed by the appropriate marker, and then the data transfer on the DDATA port. The number of captured data bytes depends on the WDDATA operand size.
0x05	Begin execution of taken branch or SYNC_PC BDM command. For some opcodes, a branch target address may be displayed on DDATA depending on the CSR settings. CSR also controls the number of address bytes displayed, indicated by the PST marker value preceding the DDATA nibble that begins the data output. This encoding also indicates that the SYNC_PC command has been processed.
0x06	Reserved
0x07	Begin execution of return from exception (RTE) instruction.
0x08–0x0B	Indicates the number of data bytes to be loaded into the PST trace buffer. The capturing of peripheral bus data references is controlled by CSR[DDC]. 0x08 Begin 1-byte data transfer on DDATA 0x09 Begin 2-byte data transfer on DDATA 0x0A Reserved 0x0B Begin 4-byte data transfer on DDATA
0x0C–0x0F	Indicates the number of address bytes to be loaded into the PST trace buffer. The capturing of branch target addresses is controlled by CSR[BTB]. 0x0C Reserved 0x0D Begin 2-byte address transfer on DDATA (Displayed address is shifted right 1: ADDR[16:1]) 0x0E Begin 3-byte address transfer on DDATA (Displayed address is shifted right 1: ADDR[23:1]) 0x0F Reserved
0x10–0x11	Reserved
0x12	Completed execution of 2 sequential instructions

Table continues on the next page...

Table 41-41. CF1 Debug Processor Status Encodings (continued)

PST[4:0]	Definition
0x13	Completed execution of 3 sequential instructions
0x14	Completed execution of 4 sequential instructions
0x15	Completed execution of 5 sequential instructions
0x16	Completed execution of 6 sequential instructions
0x17	Completed execution of 7 sequential instructions
0x18	Completed execution of 8 sequential instructions
0x19	Completed execution of 9 sequential instructions
0x1A	Completed execution of 10 sequential instructions
0x1B	This value signals there has been a change in the breakpoint trigger state machine. It appears as a single marker for each state change and is immediately followed by a DDATA value signaling the new breakpoint trigger state encoding. The DDATA breakpoint trigger state value is defined as (0x20 + 2 × CSR[BSTAT]): 0x20 No breakpoints enabled 0x22 Waiting for a level-1 breakpoint 0x24 Level-1 breakpoint triggered 0x2A Waiting for a level-2 breakpoint 0x2C Level-2 breakpoint triggered
0x1C	Exception processing. This value signals the processor has encountered an exception condition. Although this is a multi-cycle mode, there are only two PST = 0x1C values recorded before the mode value is suppressed.
0x1D	Emulator mode exception processing. This value signals the processor has encountered a debug interrupt or a properly-configured trace exception. Although this is a multi-cycle mode, there are only two PST = 0x1D values recorded before the mode value is suppressed.
0x1E	Processor is stopped. This value signals the processor has executed a STOP instruction. Although this is a multi-cycle mode because the ColdFire processor remains stopped until an interrupt or reset occurs, there are only two PST = 0x1E values recorded before the mode value is suppressed.
0x1F	Processor is halted. This value signals the processor has been halted. Although this is a multi-cycle mode because the ColdFire processor remains halted until a BDM go command is received or reset occurs, there are only two PST = 0x1F values recorded before the mode value is suppressed.

41.4.3.1 Begin Execution of Taken Branch (PST = 0x05)

The PST is 0x05 when a taken branch is executed. For some opcodes, a branch target address may be loaded into the trace buffer (PSTB) depending on the CSR settings. CSR also controls the number of address bytes loaded that is indicated by the PST marker value immediately preceding the DDATA entry in the PSTB that begins the address entries.

Multiple byte DDATA values are displayed in least-to-most-significant order. The processor captures only those target addresses associated with taken branches that use a variant addressing mode (RTE and RTS instructions, JMP and JSR instructions using address register indirect or indexed addressing modes, and all exception vectors).

The simplest example of a branch instruction using a variant address is the compiled code for a C language case statement. Typically, the evaluation of this statement uses the variable of an expression as an index into a table of offsets, where each offset points to a unique case within the structure. For such change-of-flow operations, the ColdFire processor loads the PSTB as follows:

1. Load PST=0x05 to identify that a taken branch is executed.
2. Optionally load the marker for the target address capture. Encodings 0x0D or 0x0E identify the number of bytes loaded into the PSTB.
3. The new target address is optionally available in the PSTB. The number of bytes of the target address loaded is configurable (2 or 3 bytes, where the encoding is 0x0D and 0x0E, respectively).

Another example of a variant branch instruction is a JMP (A0) instruction. The following figure shows the PSTB entries that indicate a JMP (A0) execution, assuming CSR[BTB] was programmed to display the lower two bytes of an address.

PST/DDATA Values	Description
0x05	Taken Branch
0x0D	2-byte Address Marker
{10, Address[4:1]}	Address >> 1
{10, Address[8:5]}	
{10, Address[12:9]}	
{10, Address[16:13]}	

Figure 41-11. Example JMP Instruction Output in PSTB

The PST of 0x05 indicates a taken branch and the marker value 0x0D indicates a 2-byte address. Therefore, the following entries display the lower two bytes of address register A0, right-shifted by 1, in least-to-most-significant nibble order. The next PST entry after the JMP instruction completes depends on the target instruction. See PST Trace Buffer (PSTB) Entry Format, for entry descriptions explaining the 2-bit prefix before each address nibble.

41.4.3.2 PST Trace Buffer (PSTB) Entry Format

As PST and DDATA values are captured and loaded in the trace buffer, each entry is six bits in size therefore, the type of the entry can easily be determined when post-processing the PSTB.

	5	4	3	2	1	0
PSTB[PST]	0	PST[4:0]				
Data PSTB[DDATA]	1	$\overline{R}$/W	Data[3:0]			
Address PSTB[DDATA]	1	0	Address[3:0][1]			
Reset:	—	—	—	—	—	—

[1] Depending on which nibble is displayed (as determined by SDR[9:8]), Address [3:0] s (least-to-most-significant nibble order) displays four bits of the real CPU address [16:1] or [24:1].

Figure 41-12. V1 PST/DDATA Trace Buffer Entry Format

41.4.3.3 PST/DDATA Example

In this section, an example showing the behavior of the PST/DDATA functionality is detailed. Consider the following interrupt service routine that counts the interrupt, negates the IRQ, performs a software IACK, and then exits. This example is presented here because it exercises a considerable set of the PST/DDATA capabilities.

```
                        _isr:
01074: 46fc 2700          mov.w   &0x2700,%sr         # disable interrupts
01078: 2f08               mov.l   %a0,-(%sp)          # save a0
0107a: 2f00               mov.l   %d0,-(%sp)          # save d0
0107c: 302f 0008          mov.w   (8,%sp),%d0         # load format/vector word
01080: e488               lsr.l   &2,%d0              # align vector number
01082: 0280 0000 00ff     andi.l  &0xff,%d0           # isolate vector number
01088: 207c 0080 1400     mov.l   &int_count,%a0      # base of interrupt counters

                        _isr_entry1:
0108e: 52b0 0c00          addq.l  &1,(0,%a0,%d0.l*4) # count the interrupt
01092: 11c0 a021          mov.b   %d0,IGCR0+1.w       # negate the irq
01096: 1038 a020          mov.b   IGCR0.w,%d0         # force the write to complete
0109a: 4e71               nop                         # synchronize the pipelines
0109c: 71b8 ffe0          mvz.b   SWIACK.w,%d0        # software iack: pending irq?
010a0: 0c80 0000 0041     cmpi.l  %d0,&0x41           # level 7 or none pending?
010a6: 6f08               ble.b   _isr_exit           # yes, then exit
010a8: 52b9 0080 145c     addq.l  &1,swiack_count     # increment the swiack count
010ae: 60de               bra.b   _isr_entry1         # continue at entry1

                        _isr_exit:
010b0: 201f               mov.l   (%sp)+,%d0          # restore d0
010b2: 205f               mov.l   (%sp)+,%a0          # restore a0
010b4: 4e73               rte                         # exit
```

This ISR executes mostly as straight-line code: there is a single conditional branch @ PC = 0x10A6, which is taken in this example. The following description includes the PST and DDATA values generated as this code snippet executes. In this example, the CSR setting enables only the display of 2-byte branch addresses. Operand data captures are not enabled. The sequence begins with an interrupt exception:

```
interrupt exception occurs @ pc = 5432 while in user mode
                                                  # pst   = 1c, 1c, 05, 0d
                                                  # ddata = 2a, 23, 28, 20
```

```
                                                        #          trg_addr = 083a << 1
                                                        #          trg_addr = 1074
                      _isr:
01074: 46fc 2700        mov.w   &0x2700,%sr             # pst    = 01
01078: 2f08             mov.l   %a0,-(%sp)              # pst    = 01
0107a: 2f00             mov.l   %d0,-(%sp)              # pst    = 01
0107c: 302f 0008        mov.w   (8,%sp),%d0             # pst    = 01
01080: e488             lsr.l   &2,%d0                  # pst    = 01
01082: 0280 0000 00ff   andi.l  &0xff,%d0               # pst    = 01
01088: 207c 0080 1400   mov.l   &int_count,%a0          # pst    = 01
0108e: 52b0 0c00        addq.l  &1,(0,%a0,%d0.l*4)      # pst    = 01
01092: 11c0 a021        mov.b   %d0,IGCR0+1.w           # pst    = 01, 08c
                                                        # ddata = 30, 30
                                                        #          wdata.b = 0x00
01096: 1038 a020        mov.b   IGCR0.w,%d0             # pst    = 01, 08
                                                        # ddata = 28, 21
                                                        #          rdata.b = 0x18
0109a: 4e71             nop                             # pst    = 01
0109c: 71b8 ffe0        mvz.b   SWIACK.w,%d0            # pst    = 01, 08
                                                        # ddata = 20, 20
                                                        #          rdata.b = 0x00
010a0: 0c80 0000 0041   cmpi.l  %d0,&0x41               # pst    = 01
010a6: 6f08             ble.b   _isr_exit               # pst    = 05 (taken branch)
010b0: 201f             mov.l   (%sp)+,%d0              # pst    = 01
010b2: 205f             mov.l   (%sp)+,%a0              # pst    = 01
010b4: 4e73             rte                             # pst    = 07, 03, 05, 0d
                                                        # ddata = 29, 21, 2a, 22
                                                        #          trg_addr = 2a19 << 1
                                                        #          trg_addr = 5432
```

As the PSTs are compressed, the resulting stream of 6-bit hexadecimal entries is loaded into consecutive locations in the PST trace buffer:

```
PSTB[*]=  1c, 1c, 05, 0d,       // interrupt exception
          2a, 23, 28, 20,       // branch target addr = 1074
          1a,                   // 10 sequential insts
          13,                   // 3 sequential insts
          05, 12,               // taken_branch + 2 sequential
          07, 03, 05, 0d,       // rte, entry into user mode
          29, 21, 2a, 22        // branch target addr = 5432
```

Architectural studies on the compression algorithm determined an appropriate size for the PST trace buffer. Using a suite of ten MCU benchmarks, a 64-entry PSTB was found to capture an average window of time of 520 processor cycles with program trace using 2-byte addresses enabled.

41.4.3.4 Processor Status, Debug Data Definition

This section specifies the ColdFire processor and debug module's generation of the processor status (PST) and debug data (DDATA) output on an instruction basis. In general, the PST/DDATA output for an instruction is defined as follows:

PST = 0x01, {PST = 0x0[89B], DDATA = operand}

where the {...} definition is optional operand information defined by the setting of the CSR, and [...] indicates the presence of one value from the list.

The CSR provides capabilities to display operands based on reference type (read, write, or both). A PST value {0x08, 0x09, or 0x0B} identifies the size and presence of valid data to follow in the PST trace buffer (PSTB) {1, 2, or 4 bytes, respectively}. Additionally, CSR[DDC] specifies whether operand data capture is enabled and what size. Also, for certain change-of-flow instructions, CSR[BTB] provides the capability to display the target instruction address in the PSTB (2 or 3 bytes) using a PST value of 0x0D or 0x0E, respectively.

41.4.3.4.1 User Instruction Set

The following table shows the PST/DDATA specification for user-mode instructions. Rn represents any {Dn, An} register. In this definition, the y suffix generally denotes the source, and x denotes the destination operand. For a given instruction, the optional operand data is displayed only for those effective addresses referencing memory. The DD nomenclature refers to the DDATA outputs.

Table 41-42. PST/DDATA Specification for User-Mode Instructions

Instruction	Operand Syntax	PST/DDATA
add.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
add.l	Dy,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
adda.l	<ea>y,Ax	PST = 0x01, {PST = 0x0B, DD = source operand}
addi.l	#<data>,Dx	PST = 0x01
addq.l	#<data>,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
addx.l	Dy,Dx	PST = 0x01
and.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
and.l	Dy,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
andi.l	#<data>,Dx	PST = 0x01
asl.l	{Dy,#<data>},Dx	PST = 0x01
asr.l	{Dy,#<data>},Dx	PST = 0x01
bcc.{b,w,l}		if taken, then PST = 0x05, else PST = 0x01
bchg.{b,l}	#<data>,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
bchg.{b,l}	Dy,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
bclr.{b,l}	#<data>,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
bclr.{b,l}	Dy,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
bitrev.l	Dx	PST = 0x01
bra.{b,w,l}		PST = 0x05
bset.{b,l}	#<data>,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
bset.{b,l}	Dy,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
bsr.{b,w,l}		PST = 0x05, {PST = 0x0B, DD = destination operand}
btst.{b,l}	#<data>,<ea>x	PST = 0x01, {PST = 0x08, DD = source operand}
btst.{b,l}	Dy,<ea>x	PST = 0x01, {PST = 0x08, DD = source operand}

Table continues on the next page...

Table 41-42. PST/DDATA Specification for User-Mode Instructions (continued)

Instruction	Operand Syntax	PST/DDATA
byterev.l	Dx	PST = 0x01
clr.b	<ea>x	PST = 0x01, {PST = 0x08, DD = destination operand}
clr.l	<ea>x	PST = 0x01, {PST = 0x0B, DD = destination operand}
clr.w	<ea>x	PST = 0x01, {PST = 0x09, DD = destination operand}
cmp.b	<ea>y,Dx	PST = 0x01, {0x08, source operand}
cmp.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
cmp.w	<ea>y,Dx	PST = 0x01, {0x09, source operand}
cmpa.l	<ea>y,Ax	PST = 0x01, {PST = 0x0B, DD = source operand}
cmpa.w	<ea>y,Ax	PST = 0x01, {0x09, source operand}
cmpi.b	#<data>,Dx	PST = 0x01
cmpi.l	#<data>,Dx	PST = 0x01
cmpi.w	#<data>,Dx	PST = 0x01
divs.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
divs.w	<ea>y,Dx	PST = 0x01, {PST = 0x09, DD = source operand}
divu.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
divu.w	<ea>y,Dx	PST = 0x01, {PST = 0x09, DD = source operand}
eor.l	Dy,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
eori.l	#<data>,Dx	PST = 0x01
ext.l	Dx	PST = 0x01
ext.w	Dx	PST = 0x01
extb.l	Dx	PST = 0x01
illegal		PST = 0x01[1]
jmp	<ea>y	PST = 0x05, {PST = 0x0[DE], DD = target address}[2]
jsr	<ea>y	PST = 0x05, {PST = 0x0[DE], DD = target address}, {PST = 0x0B, DD = destination operand}[2]
lea.l	<ea>y,Ax	PST = 0x01
link.w	Ay,#<displacement>	PST = 0x01, {PST = 0x0B, DD = destination operand}
lsl.l	{Dy,#<data>},Dx	PST = 0x01
lsr.l	{Dy,#<data>},Dx	PST = 0x01
mov3q.l	#<data>,<ea>x	PST = 0x01, {PST = 0x0B,DD = destination operand}
move.b	<ea>y,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
move.l	<ea>y,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
move.w	<ea>y,<ea>x	PST = 0x01, {PST = 0x09, DD = source}, {PST = 0x09, DD = destination}
move.w	CCR,Dx	PST = 0x01
move.w	{Dy,#<data>},CCR	PST = 0x01
movea.l	<ea>y,Ax	PST = 0x01, {PST = 0x0B, DD = source}
movea.w	<ea>y,Ax	PST = 0x01, {PST = 0x09, DD = source}
movem.l	#list,<ea>x	PST = 0x01, {PST = 0x0B, DD = destination},...
movem.l	<ea>y,#list	PST = 0x01, {PST = 0x0B, DD = source},...

Table continues on the next page...

Table 41-42. PST/DDATA Specification for User-Mode Instructions (continued)

Instruction	Operand Syntax	PST/DDATA
moveq.l	#<data>,Dx	PST = 0x01
muls.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
muls.w	<ea>y,Dx	PST = 0x01, {PST = 0x09, DD = source operand}
mulu.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
mulu.w	<ea>y,Dx	PST = 0x01, {PST = 0x09, DD = source operand}
mvs.b	<ea>y,Dx	PST = 0x01, {0x08, source operand}
mvs.w	<ea>y,Dx	PST = 0x01, {0x09, source operand}
mvz.b	<ea>y,Dx	PST = 0x01, {0x08, source operand}
mvz.w	<ea>y,Dx	PST = 0x01, {0x09, source operand}
neg.l	Dx	PST = 0x01
negx.l	Dx	PST = 0x01
nop		PST = 0x01
not.l	Dx	PST = 0x01
or.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
or.l	Dy,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
ori.l	#<data>,Dx	PST = 0x01
pea.l	<ea>y	PST = 0x01, {PST = 0x0B, DD = destination operand}
pulse		PST = 0x04
rems.l	<ea>y,Dw:Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
remu.l	<ea>y,Dw:Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
rts		PST = 0x01, {PST = 0x0B, DD = source operand}, PST = 0x05, {PST = 0x0[DE], DD = target address}
sats.l	Dx	PST = 0x01
scc.b	Dx	PST = 0x01
sub.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
sub.l	Dy,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
suba.l	<ea>y,Ax	PST = 0x01, {PST = 0x0B, DD = source operand}
subi.l	#<data>,Dx	PST = 0x01
subq.l	#<data>,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
subx.l	Dy,Dx	PST = 0x01
swap.w	Dx	PST = 0x01
tas.b	<ea>x	PST = 0x01, {0x08, source}, {0x08, destination}
tpf		PST = 0x01
tpf.l	#<data>	PST = 0x01
tpf.w	#<data>	PST = 0x01
trap	#<data>	PST = 0x01[1]
tst.b	<ea>x	PST = 0x01, {PST = 0x08, DD = source operand}
tst.l	<ea>y	PST = 0x01, {PST = 0x0B, DD = source operand}
tst.w	<ea>y	PST = 0x01, {PST = 0x09, DD = source operand}

Table continues on the next page...

Table 41-42. PST/DDATA Specification for User-Mode Instructions (continued)

Instruction	Operand Syntax	PST/DDATA
unlk	Ax	PST = 0x01, {PST = 0x0B, DD = destination operand}
wddata.b	<ea>y	PST = 0x04, {PST = 0x08, DD = source operand}
wddata.l	<ea>y	PST = 0x04, {PST = 0x0B, DD = source operand}
wddata.w	<ea>y	PST = 0x04, {PST = 0x09, DD = source operand}

1. During normal exception processing, the PSTB is loaded with two successive 0x1C entries indicating the exception processing state. The exception stack write operands, as well as the vector read and target address of the exception handler may also be displayed.

```
Exception Processing:
      PST = 0x1C, 0x1C,
     {PST = 0x0B,DD = destination},                 // stack frame
     {PST = 0x0B,DD = destination},                 // stack frame
     {PST = 0x0B,DD = source},                      // vector read
      PST = 0x05,{PST = 0x0[DE],DD = target}     // handler PC
```

   A similar set of PST/DD values is generated in response to an emulator mode excetion. For these events (caused by a debug interrupt or properly-enabled trace exception), the initial PST values are 0x1D, 0x1D and the remaining sequence is equivalent to normal exception processing. The PST/DDATA specification for the reset exception is shown below:

```
Exception Processing:
     PST = 0x1C, 0x1C,
     PST = 0x05,{PST = 0x0[DE],DD = target}     // initial PC
```

   The initial references at address 0 and 4 are never captured nor displayed because these accesses are treated as instruction fetches. For all types of exception processing, the PST = 0x1C (or 0x1D) value is driven for two trace buffer entries.
2. For JMP and JSR instructions, the optional target instruction address is displayed only for those effective address fields defining variant addressing modes. This includes the following <ea>x values: (An), (d16,An), (d8,An,Xi), (d8,PC,Xi).

Table 41-43 shows the PST/DDATA specification for the MAC instructions if the optional MAC unit is present.

Table 41-43. PST/DDATA Values for Multiply-Accumulate Instructions

Instruction	Operand Syntax	PST/DDATA
mac.l	Ry,Rx	PST = 0x1
mac.l	Ry,Rx,<ea>y,Rw	PST = 0x1, {PST = 0xB, DD = source operand}
mac.w	Ry,Rx	PST = 0x1
mac.w	Ry,Rx,ea,Rw	PST = 0x1, {PST = 0xB, DD = source operand}
move.l	{Ry,#<data>},ACC	PST = 0x1
move.l	{Ry,#<data>},MACSR	PST = 0x1
move.l	{Ry,#<data>},MASK	PST = 0x1
move.l	ACC,Rx	PST = 0x1
move.l	MACSR,CCR	PST = 0x1
move.l	MACSR,Rx	PST = 0x1
move.l	MASK,Rx	PST = 0x1
msac.l	Ry,Rx	PST = 0x1

Table continues on the next page...

Table 41-43. PST/DDATA Values for Multiply-Accumulate Instructions (continued)

Instruction	Operand Syntax	PST/DDATA
msac.l	Ry,Rx,<ea>y,Rw	PST = 0x1, {PST = 0xB, DD = source}, {PST = 0xB, DD = destination}
msac.w	Ry,Rx	PST = 0x1
msac.w	Ry,Rx,<ea>y,Rw	PST = 0x1, {PST = 0xB, DD = source}, {PST = 0xB, DD = destination}

41.4.3.4.2 Supervisor Instruction Set

The supervisor instruction set has complete access to the user mode instructions plus the opcodes shown below. The PST/DDATA specification for these opcodes is shown in the following table.

Table 41-44. PST/DDATA Specification for Supervisor-Mode Instructions

Instruction	Operand Syntax	PST/DDATA
halt		PST = 0x1F, PST = 0x1F
move.l	Ay,USP	PST = 0x01
move.l	USP,Ax	PST = 0x01
move.w	SR,Dx	PST = 0x01
move.w	{Dy,#<data>},SR	PST = 0x01, {PST = 0x03}
movec.l	Ry,Rc	PST = 0x01
rte		PST = 0x07, {PST = 0x0B, DD = source operand}, {PST = 0x03}, {PST = 0x0B, DD = source operand}, PST = 0x05, {PST = 0x0[DE], DD = target address}
stldsr.w	#imm	PST = 0x01, {PST = 0x0B, DD = destination operand, PST = 0x03}
stop	#<data>	PST = 0x1E, PST = 0x1E
wdebug.l	<ea>y	PST = 0x01, {PST = 0x0B, DD = source, PST = 0x0B, DD = source}

The move-to-SR, STLDSR, and RTE instructions include an optional PST = 0x3 value, indicating an entry into user mode.

Similar to the exception processing mode, the stopped state (PST = 0x1E) and the halted state (PST = 0x1F) display this status for two entries when the ColdFire processor enters the given mode.

41.4.4 Freescale-Recommended BDM Pinout

Typically, a relatively simple interface pod is used to translate commands from a host computer into commands for the custom serial interface to the single-wire background debug system. Depending on the development tool vendor, this interface pod may use a standard RS-232 serial port, a parallel printer port, or some other type of communications such as a universal serial bus (USB) to communicate between the host PC and the pod. The pod typically connects to the target system with ground, the BKGD pin, RESET, and sometimes V_{DD}. An open-drain connection to reset allows the host to force a target system reset, useful to regain control of a lost target system or to control startup of a target system before the on-chip nonvolatile memory has been programmed. Sometimes V_{DD} can be used to allow the pod to use power from the target system to avoid the need for a separate power supply. However, if the pod is powered separately, it can be connected to a running target system without forcing a target system reset or otherwise disturbing the running application program.

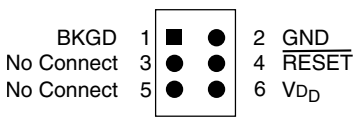


Figure 41-13. Recommended BDM Connector

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/salestermsandconditions.

Freescale, the Freescale logo, AltiVec, C-5, CodeTest, CodeWarrior, ColdFire, C-Ware, Energy Efficient Solutions logo, Kinetis, mobileGT, PowerQUICC, Processor Expert, QorIQ, Qorivva, StarCore, Symphony, and VortiQa are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Airfast, BeeKit, BeeStack, ColdFire+, CoreNet, Flexis, MagniV, MXC, Platform in a Package, QorIQ Qonverge, QUICC Engine, Ready Play, SafeAssure, SMARTMOS, TurboLink, Vybrid, and Xtrinsic are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.
© 2013 Freescale Semiconductor, Inc.

Document Number: MCF51QW256RM
Rev.1
01/2013

freescale™
semiconductor